UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

SISTEMA PARA CARGA DE DADOS EM DATA WAREHOUSE

DIEGO PAULIN

BLUMENAU
2009

2009/2-04

DIEGO PAULIN

SISTEMA PARA CARGA DE DADOS EM DATA WAREHOUSE

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação— Bacharelado.

Prof. Wilson Pedro Carli, Mestre - Orientador

BLUMENAU
2009

2009/2-04

SISTEMA PARA CARGA DE DADOS EM DATA WAREHOUSE

Por

DIEGO PAULIN

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Wilson Pedro Carli, Mestre – Orientador, FURB
Membro:	Prof. Adilson Vahldick, Mestre - FURB
Membro:	Prof. Cláudio Ratke, Mestre – FURB

Blumenau, 14 de dezembro de 2009.

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste. Aos meus pais, que sempre me apoiaram e encorajaram nos estudos.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, o pilar que me sustentou e me trouxe onde estou.

Aos meus amigos, pelo incentivo e força.

Ao meu orientador, Wilson Pedro Carli pela ajuda durante o desenvolvimento deste trabalho.

"Seja a mudança que você quer ver no mundo."

Mahatma Gandhi

RESUMO

Neste trabalho foi desenvolvida uma ferramenta para a automatização do processo de carga de dados em um *Data Warehouse*, construída em ambiente *web*, usando tecnologias Asp.Net, C Sharp (C#), Oracle, NHibernate, Aquarium Framework, Coolite para construção de um desktop *online*. A ferramenta abrange os processos de extração, limpeza, transformação e carga de dados, partindo de sistemas legados para auxiliar a unidade de desenvolvimento da FEESC no processo de carga no *Data Warehouse* de seus parceiros.

Palavras-chave: *Data Warehouse*. Banco de dados. ETL. *Framework* web

ABSTRACT

In this work was develop a tool to automate the process of loading data into a Data Warehouse, built in web environment, using technologies like Asp.Net, C Sharp (C #), Oracle, NHibernate, Aquarium Framework and Coolite to build a desktop online. The tool covers the processes of extraction, cleaning, transformation and load data from legacy systems to help drive development of FEESC in the process of loading data to the *Data Warehouse*'s partners.

Key-words: *Data Warehouse*. Data base. ETL. *Web framework*

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

BI – *Business Intelligence*

DM – *Data Mart*

DW – *Data Warehouse*

ETL – *Extract Transform Load*

FEESC – Fundação de Ensino e Engenharia de Santa Catarina

MVC – *Model View Controller*

OLAP – *Online Analytical Processing*

OLTP – *Online Transaction Processing*

SQL – *Structured Query Language*

TI – Tecnologia da Informação

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Conforme Inmon, Welch e Glassey (1999), um número cada vez maior de empresas e corporações que usam aplicações de processos de transação *online, Online Transaction Processing* (OLTP) está incorporando soluções para análise da grande quantidade de dados. Tornar esta grande quantidade de dados em informações que possam ser consumidas deixou de ser um artigo de luxo e passou a ser uma necessidade para manter as empresas vivas no mercado. Com a ajuda de indicadores críticos, como financeiros, estatísticos, que proporcionem informações em tempo real, os tomadores de decisões das empresas podem aproveitar melhor o seu tempo para extrair informações relevantes, ao invés de perder tempo para organizá-las.

Conforme Brackett (1996), quando bem implementados, os sistemas analíticos se tornam uma poderosa ferramenta para apoio a tomada de decisão, possibilitando análise de dados em vários níveis e granularidade, proporcionando a descoberta de novas oportunidades. O grande desafio em questão é justamente a implementação desses sistemas, pelo fato de serem complexos e altamente dependentes da regra de negócio no qual serão fundamentados.

O *Data Warehouse* (DW) surge como uma solução para implementação de sistemas analíticos. Sua estrutura de armazenamento de dados em forma de medidas e dimensões torna a leitura dos dados mais simples, facilitando a extração de informações úteis de sistemas operacionais/transacionais.

Segundo Inmon, Terdeman e Imhoff (2001), os *Data Warehouses* são como uma caixa cheia de peças de lego, em que se pode construir uma infinidade de possibilidades. Os *Data Marts* seriam os tipos de construções que se pode fazer com as peças de Lego, como por exemplo, tem-se carros, pontes, robôs, dentre outros.

Surge então a necessidade de dividir banco de dados operacionais dos bancos de dados analíticos. Segundo Inmon (1997) isso deve ocorrer por vários motivos, dentre eles pode se citar a diferença física de disposição dos dados, a comunidade de usuários que usam dados operacionais é diferente da comunidade que necessitam de dados analíticos assim como as características de processamento entre estes tipos de dados se diferem.

Com esta separação dos dados operacionais dos dados analíticos surgiram os programas de extração de dados. Estes programas de extração são responsáveis por fazerem a coleta de dados operacionais que são úteis para um DW.

O programa que realiza a *interface* de integração e transformação dos dados de um

sistema legado para o DW é conceitualmente simples, como mencionado em Inmon, Terdeman e Imhoff (2001), porém de natureza extremamente crítica. Definir quais os dados que farão parte do DW e como transformá-los em informações úteis é uma tarefa tão complexa a ponto de necessitar de técnicas próprias de gerenciamento, conhecimento tecnológico e da regra de negócio da aplicação em questão.

Um DW é a base para um sistema de *Business Intelligence* (BI), um termo de gerenciamento de negócios que se refere a aplicações e tecnologias empregadas para coletar, fornecer acesso e analisar dados e informações sobre as operações das empresas (BARBIERI, 2001).

Desta forma, a Fundação de Ensino e Engenharia de Santa Catarina (FEESC), em sua unidade de desenvolvimento de softwares têm aplicações BI alimentadas por DW, que necessitam constantemente de carga de dados para manter os dados de seus sistemas atualizados. É com base nestes programas de extração de dados de ambientes transacionais para ambientes analíticos que este trabalho está focado.

Foi desenvolvido o sistema denominado Freedom, uma solução para o processo de carga de dados no DW, usando os princípios da *Extract, Transform and Load* (ETL), que significa extração, transformação e carga dos dados, processo esse imprescindível para construção e funcionamento de sistemas analíticos desenvolvidos pela FEESC.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de uma aplicação para dar carga de dados em um DW.

Os objetivos específicos do trabalho são:

a) desenvolver uma aplicação web para execução da carga como objetivo de reduzir o tempo gasto no processo de carga de um DW;
b) automatizar o processamento dos cubos de decisão usando a ferramenta da Microsoft SQL Server Analysis Services (SSAS) para o banco de dados Oracle.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado na forma de capítulos. No capítulo um tem–se a introdução das funcionalidades da aplicação, abordando os principais objetivos a serem alcançados com a sua construção e utilização.

No segundo capítulo são apresentados os assuntos relacionados à fundamentação teórica usada na construção desta aplicação.

Já no terceiro capítulo é apresentada a forma com que foi construída a aplicação, partindo dos requisitos do sistema, suas especificações e a implementação, funcionalidades e os processos a serem adotados para o seu uso.

Por fim, no quarto capítulo tem-se as conclusões obtidas com o desenvolvimento da aplicação, bem como sugestões para futuras implementações visando o seu aprimoramento.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica necessária para a compreensão do trabalho e sua relevância, além de apresentar trabalhos correlatos.

O primeiro tópico abordado pelo capítulo apresenta os conceitos referentes ao tema *Data Warehouse*, o conceito de maior relevância para o trabalho. Em seguida é abordado o conceito de OLAP, sua definição e principais características como ferramenta de apoio ao processo de tomada de decisão.

Na sequência é apresentado o processo ETL, que foi implementado pela ferramenta desenvolvida neste trabalho. No quarto tópico é apresentado o conceito de metadados, sua relevância e importância dentro do *Data Warehouse.* Já no quinto tópico são relacionados os trabalhos correlatos, como ferramentas existentes no mercado e outros trabalhos de conclusão de curso.

2.1 DATA WAREHOUSE

Conforme conceito expresso em Inmon (1997), um DW é um banco de dados orientado por assunto, altamente integrado, que mantém informações históricas de seus registros, não volátil e que está organizado de forma que beneficie no processo de apoio à tomada de decisão.

Trata-se de um banco de dados alimentado continuamente, porém diferentes da forma com que os banco de dados transacionais são alimentados. Enquanto os bancos de dados transacionais processam centenas, ou até mesmo milhares de transações por dia, onde cada transação contém uma pequena parte do dado, um DW frequentemente processará uma única transação. Porém esta transação irá conter centenas ou até mesmo milhares de registros. Em um DW este processo massivo de transações tem o nome de carga de dados (KIMBALL, 1996).

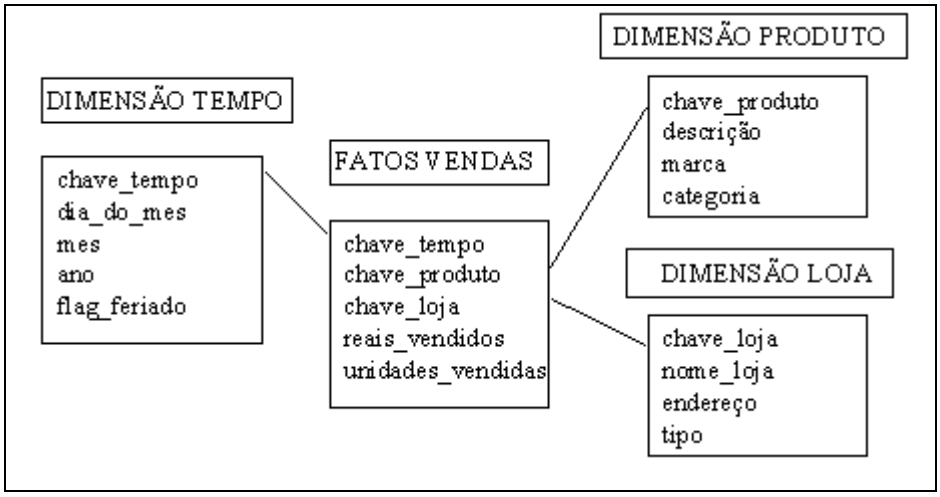
Os componentes básicos de um DW são as tabelas de fatos, as medidas e as dimensões. As tabelas de fato armazenam os dados do negócio propriamente dito. Nesta tabela estarão informações decorrentes do negócio que não são conhecidas previamente. Os componentes da tabela de fato são as medidas e as dimensões.

As medidas são os valores relacionados aos fatos. Eles são dados numéricos que ilustram os acontecimentos dentro de uma organização, desconhecidos até que aconteçam. As medidas podem representar quantidades, valores, podendo estes ser acumulados ou não, índices, dentre outras informações quantitativas de uma organização (JACOBSON, 2007).

Por fim, tem-se as dimensões, que guardam as informações relacionadas aos atributos do negócio. Os atributos do negócio são as descrições das características envolvidas em uma determinada regra de negócio. São dados conhecidos pelas organizações. As dimensões são usadas na realização de pesquisas nas tabelas de fato, restringindo a buscas por informações e dando diferentes perspectivas dos fatos e seus valores (JACOBSON, 2007).

Uma das dimensões mais importante de um DW é a dimensão tempo, presente em todos os DW. Fatos isolados que acontecem ao mesmo tempo podem ter relações que dificilmente são visualizadas. É o fator principal de restrições de pesquisas em um DW. Os dados desta dimensão são representados em forma de datas, anos, meses do ano, semanas do ano, dias da semana, horas do dia. O nível de detalhes da dimensão tempo está diretamente relacionada à regra de negócio que será implementada no DW (JACOBSON, 2007).

Dentre as formas de se arquitetar um DW a mais conhecida é o esquema estrela. No esquema estrela existe a tabela de fatos como sendo a tabela central da estrutura. Esta tabela de fato contém todas as medidas relacionadas aos fatos, ou seja, os valores numéricos dos fatos. Além de conter as dimensões, a tabela de fatos também possui um identificador único (chave estrangeira) para cada uma das dimensões relacionadas ao fato. Na figura 1 tem-se um exemplo do modelo estrela de DW.



Fonte: SHAMMAS (2009).

Figura 1 – Esquema estrela

2.2 ONLINE ANALYTICAL PROCESSING

O termo OLAP é definido como processamento analítico *online*, que caracteriza um conjunto de tecnologias de software destinadas a usuário que participam de decisões a nível tático e estratégico de uma corporação, proporcionando acesso aos dados de forma dinâmica, permitindo a análise destes dados em diferentes perspectivas, visando refletir as dimensões reais de uma determinada regra de negócio (INMON, 1997).

Conforme Inmon (1997, p. 175), o OLAP é um método importante na arquitetura de um DW, através do qual os dados podem ser transformados em informações. À medida que o volume de dados a serem manipulados em um DW cresce, também aumenta a necessidade de uso de tecnologias mais sofisticadas para a sua manutenção.

Para Inmon (1997, p. 176), o OLAP é uma extensão natural de um DW, pois desde a concepção da sua arquitetura ele se torna ideal para a execução do processamento analítico *online*. O nível estruturado em forma de departamentos dentro de um DW, ou seja, as subdivisões de áreas de negócio implementados são chamadas de níveis OLAP, atualmente conhecidos como *Data Marts* (DM).

Existem uma série de regras a serem seguidas para implementação da tecnologia OLAP em um DW. A seguir, tem-se a lista de algumas dessas regras:

a) visão conceitual multidimensional: dados modelados em uma estrutura multidimensional;
b) transparência: funcionalidades OLAP devem estar transparentes para o usuário final;
c) acessibilidade: acesso a várias fontes de dados;
d) arquitetura cliente servidor: operar em arquitetura cliente servidor;
e) flexibilidade na geração de relatórios: fácil manipulação das informações para a geração de relatórios analíticos;
f) dimensionalidade genérica: as dimensões devem ser iguais quanto a sua estrutura, e devem estar relacionadas a um fato;
g) operação dimensional cruzada irrestrita: capacidade de cruzar dados de diferentes estruturas livremente, a fim de encontrar novas informações..

2.3 EXTRACT TRANSFORM AND LOAD (ETL)

Considerada a fase mais crítica de um DW, o processo *Extract Transform and Load* (ETL), cuja tradução significa, extração, transformação e carga, se trata do processo de extração de dados de sistemas ou ambientes legados, transformação destes dados, realizando as operações necessárias para adequá-los a regra de negócio do DW e a carga dos mesmos em um DW.

Geralmente são construídas aplicações que realizam este processo seguindo uma série de regras para garantir que a carga de dados seja realizada com sucesso (KIMBALL, 1996). Estas aplicações possuem um conceito relativamente simples, tirar os dados de um banco de dados e colocar em outro. Porém a sua natureza é extremamente crítica, pois são estas ferramentas quem irão determinar os dados a serem manipulados pelo DW, conforme mencionado em Inmon, Terdeman e Imhoff (2001).

O conceito de “tirar de um lado e colocar no outro”, realmente parece ser uma tarefa simples, mas a forma com que isto será feito é extremamente complexa, chegando ao ponto de exigir uma equipe inteira de TI para execução dos processos de carga de dados. Toda esta complexidade se deve ao fato de trabalhar com vários níveis de dados que podem estar organizadas de diversas formas diferentes, realizar operações nestes dados e migrá-los para uma outra estrutura (DW) com um conceito de armazenamento completamente diferente de onde estes dados originalmente estavam.

O processo de extração de dados deve ser capaz de acessar as informações de várias fontes diferentes de dados. Estas fontes podem ser os bancos de dados de diversos sistemas computacionais de uma organização, documentos digitalizados, planilhas de ferramentas como o Microsoft Excel e a própria internet. A diversidade a ser atingida pelo processo de extração de dados vai depender diretamente da necessidade de informações que deve conter o DW.

Conforme Brackett (1996), a transformação é um processo formal de transformar dados de várias fontes em recursos de dados em uma arquitetura comum, ou seja, dados integrados.

Ainda em Brackett (1996), é ressaltada a relevância deste processo, onde destaca: “Transformar dados não é uma tarefa trivial”. É um processo formal que exige planejamento para assegurar que uma organização possa sobreviver com seus dados diferentes enquanto desenvolve recursos de dados integrados.

A transformação de dados é um dos processos mais importantes a ser definido pelo profissional de DW, e também um dos mais entediantes, conforme Inmon, Welch e Glassey (1999), devido a sua *interface* complicada e a série de exigências que deve levar em conta, como:

a) adequar-se a novas tecnologias de DW;
b) acesso eficiente a ambientes legados;
c) reformatação de estrutura de dados interna;
d) seleção a partir de várias origens de dados;
e) adição de um elemento de tempo.

Uma vez que os dados possuem origens diferentes, nem sempre os dados que dizem respeito à mesma área de negócio estão formatados da mesma maneira. Um exemplo clássico desta situação é a forma com que diferentes sistemas tratam as datas, ora com quatro dígitos, ora com dois dígitos. Estas situações demandam uma etapa de integração dos dados. É também na fase de transformação que os dados são adequados a realidade do DW.

A última etapa do processo ETL consiste em inserir no DW os dados que foram extraídos, integrados e transformados, assumindo a arquitetura dimensional. Os dados devem ser migrados em forma de fatos, contendo suas medidas e dimensões.

2.4 METADADOS

O metadado é outro componente de suma importância para um DW. O metadado em um DW tem o mesmo princípio de utilização que um sumário ou um índice em um livro. Através do metadado é possível identificar a localização dos dados em um DW, sua origem, seu significado e as transformações que ocorreram no processo ETL (GONÇALVES, 2003).

Também conhecido como dicionário de dados, os metadados podem assumir duas formas distintas de utilização. A primeira forma é o metadado corporativo, cuja finalidade tem o foco no usuário final, visando fornecer informações a nível de negócio. A segunda forma é o metadado técnico, com enfoque na equipe de desenvolvimento do DW, contendo informações técnicas como *scripts* usados durante as fases de extração e transformação dos dados, dentre outras informações úteis à equipe técnica do DW.

Conforme Singn (2001), os metadados podem ser usados como um diretório que contribui para a localização dos dados dentro de um DW, como um guia de mapeamento dos

dados carregados e sumarizados.

2.5 TRABALHOS CORRELATOS

Uma ferramenta que realiza o processo ETL de forma muito eficiente e de grande aceitação no mercado é o Kettle, desenvolvido na linguagem de programação Java pelo grupo Pentaho, conhecido por seus produtos de código aberto na área de BI (PENTAHO, 2009). O Kettle é uma ferramenta de código aberto que implementa o processo ETL. Tratando-se de uma ferramenta já consolidada no mercado, possui uma infinidade de adeptos. É constituído por quatro ferramentas:

a) Spoon: uma ferramenta gráfica para modelar o fluxo de transformação dos dados;
b) Pan: que executa as transformações mapeadas pelo Spoon;
c) Chef: modelador de tarefas;
d) Kitchen: que executa as tarefas do Chef.

O trabalho realizado pelo professor Evaristo Baptista, propõe alternativas de migração de sistemas transacionais para o ambiente DW, dando relevância para o processo de carga de dados para um DW, onde se fundamenta a importância do desenvolvimento da aplicação proposta (BAPTISTA, 1998).

No trabalho de conclusão de curso de Henrique J. Strube desenvolvido na Universidade Regional de Blumenau (FURB), o mesmo propõe o estudo de um caso real de migração de banco de dados de sistemas transacionais para Data Warehouse, onde demonstra com clareza as etapas que envolvem o processo e exalta a importância do DW em ambientes corporativos como um diferencial competitivo (STRUBE, 2001).

Quanto à relevância e a importância do desenvolvimento de aplicações BI para manter uma empresa no competitivo mercado globalizado, fornecendo informações necessárias a todos os níveis da empresa, em suporte aos objetivos estratégicos, o trabalho de Giseli Sanzon, também desenvolvido na FURB, pode ser citado (SANZON, 2006).

3 DESENVOLVIMENTO

Neste capítulo são apresentados os aspectos técnicos referentes ao desenvolvimento do trabalho. No tópico inicial é descrita as funcionalidades da aplicação e os requisitos a serem atendidos. Os tópicos seguintes são dedicados a especificação da ferramenta, contendo diagramas *Unified Modeling Language* (UML) com o objetivo de definir os processos realizados pelo software. Na sequência são detalhadas uma a uma as funcionalidades presentes na aplicação desenvolvida neste trabalho.

3.1 LEVANTAMENTO DE INFORMAÇÕES

O sistema desenvolvido neste trabalho tem por objetivo automatizar o processo de carga de dados em DW, implantados pela FEESC, através de um sistema *web*, denominado Freedom. O sistema contempla as seguintes funcionalidades:

a) extração de dados: que resolverá questões como a freqüência da extração, conteúdo que será carregado, procedimentos com dados que foram alterados desde a última extração, uso de recursos para realizar a extração;
b) transformação dos dados: agregação, fusão, ordenação dos dados e operações pertinentes a regra de negócio será o foco no processo de transformação dos dados. Este processo poderá ser executado em várias etapas, dependendo da regra de negócio implementada;
c) carga de dados: é o último passo do processo ETL . Nesta etapa os dados serão armazenados no DW, na forma de dimensões e tabelas de fato, automatizando o processo antes realizado por uma série de *scripts* de inserção no banco de dados;
d) processamento do cubo de decisão: etapa na qual os dados serão preparados para serem manipulados por uma aplicação BI. Esse processamento acontecerá de forma automática, assim que uma carga de dados for realizada no DW.

A implantação desta aplicação terá impacto direto no desempenho e padronização do processo de carga de dados nos DW's implantados pela FEESC, poupando mão de obra de programador e analista durante todo o processo.

No quadro 1 apresenta-se os requisitos funcionais utilizados pelo sistema e sua

rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Requisitos Funcionais	Caso de Uso
RF01: O sistema deverá possibilitar o cadastro do DW que sofrerá a carga de dados.	UC01
RF02: O sistema deverá permitir ao analista o cadastro dos *Data Mart* que compõem o DW.	UC02
RF03: O sistema deverá permitir ao analista cadastrar as tabelas de fato dos DM.	UC03
RF04: O sistema deverá permitir ao analista cadastrar as medidas das tabelas de fato.	UC04
RF05: O sistema deverá permitir ao analista cadastrar as dimensões que serão usadas pelo DM.	UC05
RF06: O sistema possibilitará ao analista a seleção da fonte de dados para a carga do DW.	UC06
RF07: O sistema deverá permitir armazenagem homogênea dos dados temporários.	UC07
RF08: O sistema permitirá ao analista transformar os dados a serem carregados no DW.	UC08
RF09: O sistema possibilitará a execução do processo de carga de dados e o agendamento do mesmo.	UC09
RF10: O sistema efetuará limpeza nos dados temporários.	UC10
RF11: O sistema deverá efetuar o processamento dos cubos de decisão.	UC11
RF12: O sistema deverá fazer a leitura da estrutura de dados do sistema legado cujos dados serão extraídos.	UC12
RF13: O sistema permitirá ao analista alterar as informações adquiridas através da leitura do metadado do sistema legado.	UC13
RF14: O sistema deverá permitir ao analista mapear os dados a serem extraídos do sistema legado.	UC14
RF15: O sistema permitirá ao analista criar regras de extração dos dados do sistema legado.	UC15
RF16: O sistema possibilitará ao analista o cadastro do modelo lógico dos dados que serão inseridos no DW.	UC16
RF17: O sistema permitirá ao analista cadastrar as unidades lógicas que irão	UC17

compor os modelos lógicos.	
RF18: O sistema deverá permitir ao analista cadastrar as regras de transformação dos dados.	UC18
RF19: O sistema permitirá que o analista cadastre regras que conversão e integração dos dados	UC19
RF20: O sistema deverá permitir ao analista o cadastro de usuários do sistema.	UC20
RF21: O sistema deverá permitir ao analista o cadastro dos tipos de usuários.	UC21
RF22: O sistema deverá permitir ao analista cadastrar regras de extração de dados.	UC22

Quadro 1 – Requisitos funcionais

O Quadro 2 lista-se os requisitos não funcionais previstos para o sistema.

Requisitos Não Funcionais
RNF01: O sistema deverá ser desenvolvido na linguagem C# usando *framework* .NET 2.0
RNF02: O sistema deverá suportar o banco de dados Oracle 10G
RNF03: O sistema deverá possuir *interface* web.
RNF04: O sistema deverá suportar o navegador Internet Explorer 7.

Quadro 2 – Requisitos não funcionais

3.2 ESPECIFICAÇÃO

Para a especificação do sistema utilizou-se a notação UML, sendo os diagramas gerados através da ferramenta Enterprise Architect.

3.2.1 Diagrama de caso de uso

Na figura 2 apresenta-se o diagrama de caso de uso dos cadastros que compõem o sistema de carga de dados desenvolvido neste trabalho. Através deste diagrama é possível visualizar todas as operações que poderão ser feitas pelo analista dentro da aplicação.

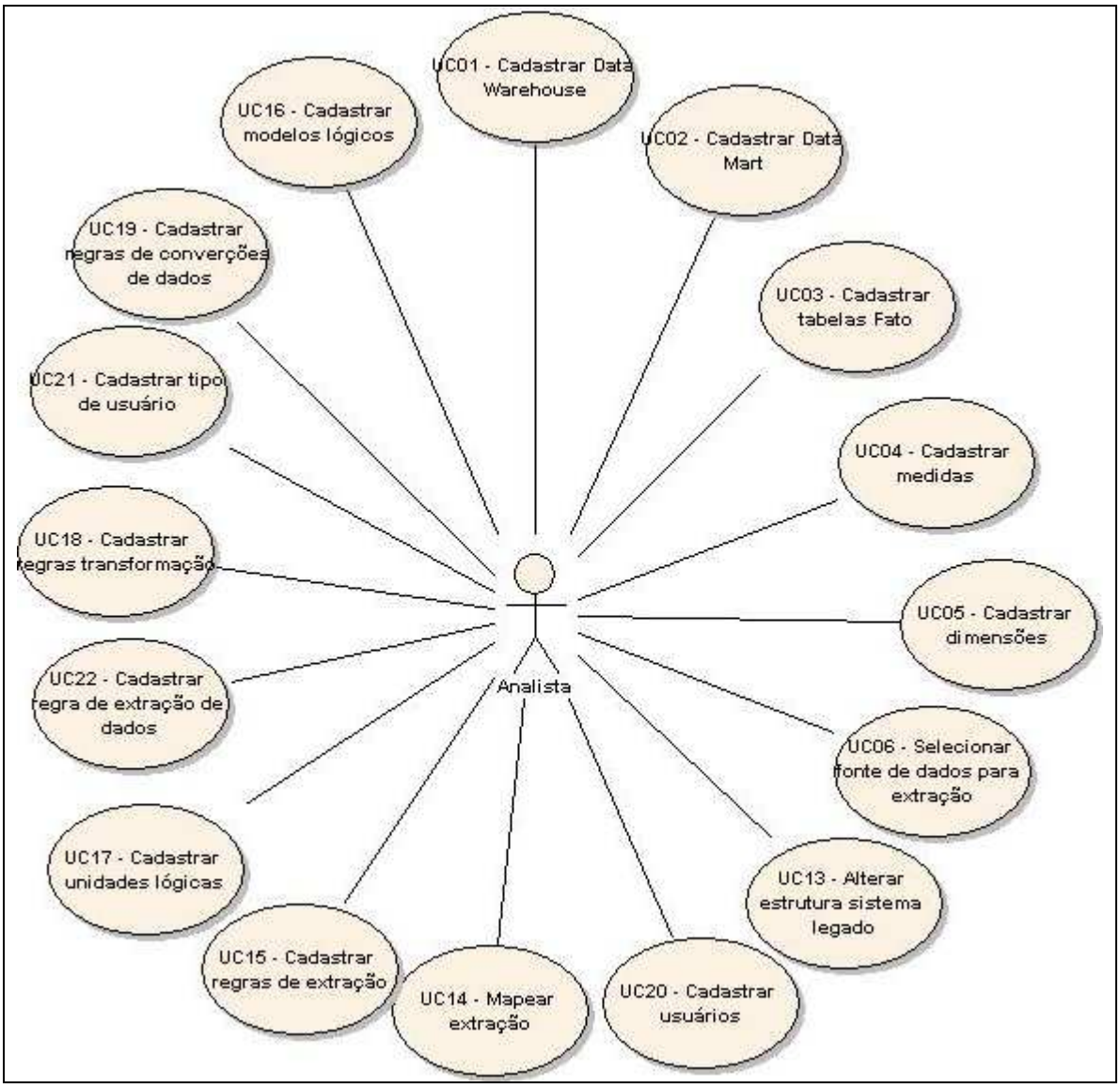


Figura 2 – Diagrama de caso de uso dos cadastros do sistema

Na figura 3 tem-se o diagrama de casos de uso, representando todas as funcionalidades que serão executadas pela própria aplicação de acordo com as configurações previamente cadastradas pelo analista.

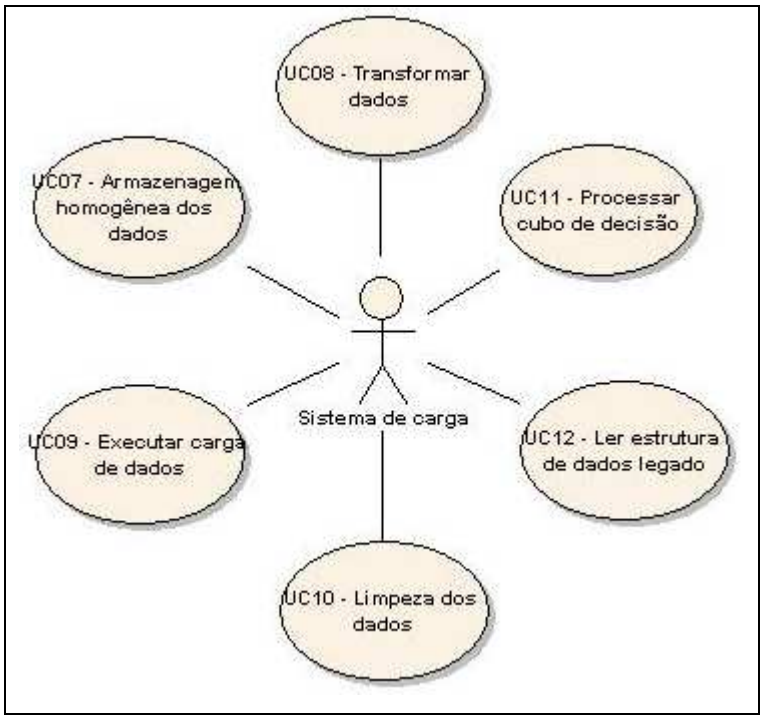


Figura 3 – Caso de uso das tarefas executadas pela aplicação

3.2.2 Modelagem dos dados

A seguir tem-se a figura 4, contendo o modelo de entidades relacionais contendo todas as entidades de banco de dados criadas para construção da aplicação.

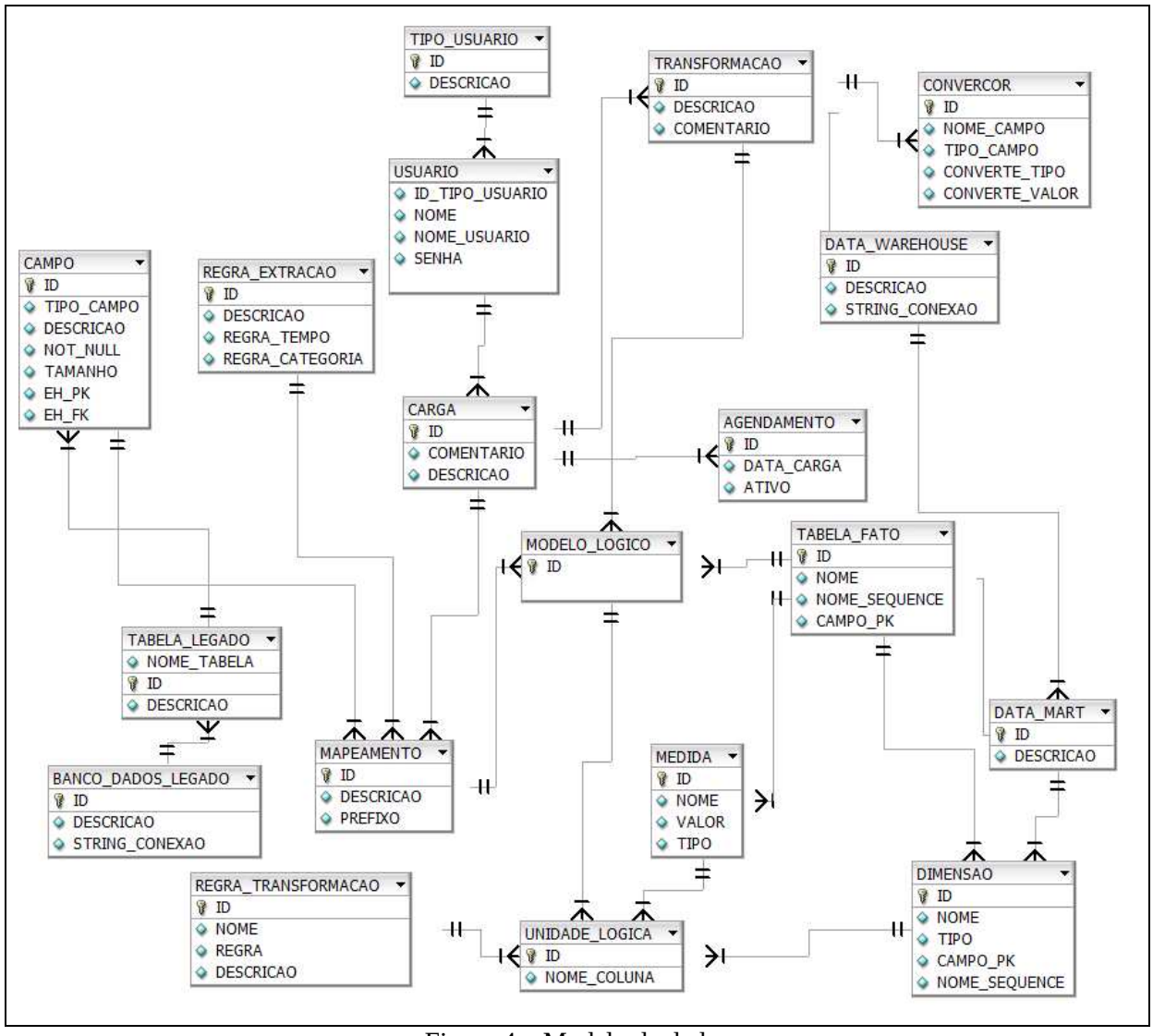


Figura 4 – Modelo de dados

A seguir tem-se a descrição das principais entidades do modelo de dados.

a) DATA_WAREHOUSE: entidade que armazena dados sobre o DW que receberá a carga de dados;

b) BANCO_DADOS_LEGADO: entidade que armazena informações necessárias para realizar conexões em bancos de dados que sofrerão o processo de extração de dados;

c) MAPEAMENTO: entidade responsável por mapear as informações que serão parte do processo de extração de dados;

d) CARGA: entidade principal do sistema de carga de dados em um DW. Contém as informações do DW que receberá a carga e das transformações que serão executadas durante a extração dos dados;

e) TABELA_FATO: nesta entidade são armazenadas informações sobre a estrutura do DW, contendo as medidas e dimensões que fazem parte da sua estrutura;

f) MODELO_LOGICO: entidade que contém dados específicos do processo de transformação dos dados durante o processo de carga. Contém informações da origem dos dados, as regras de transformações que serão aplicadas e o destino destes dados dentro do DW;
g) TRANSFORMACAO: entidade que contém as definições das transformações que o processo de carga sofrerá;
h) REGRA_TRANSFORMACAO: entidade contém as operações que serão executadas durante a fase de transformação dos dados;
i) CAMPO: contém informações dos campos que compõem uma entidade de um sistema de banco de dados;
j) DATA_MART: contém as definições dos DM que compõem um DW;

3.2.3 Diagrama de atividades

Para a compreensão do funcionamento e do fluxo de processos que serão executados dentro da aplicação desenvolvida neste trabalho foram criados diagramas de atividades. Através destes diagramas é possível modelar o comportamento da aplicação e o seu fluxo lógico de funcionamento. Na figura 5, tem-se o diagrama de atividades para iniciar o processo de carga de dados em um DW. Neste diagrama é apresentado o fluxo de atividades a serem executadas pelo analista e pelo sistema para mapear a extração de dados de um sistema legado.

O processo tem início com o cadastro da carga de dados, onde o analista irá informar qual o nome deste processo de carga e comentários relevantes a este processo. Em seguida o analista deverá cadastrar de quais fontes de dados serão extraídos os dados que serão usados no processo de extração. Será feita uma validação desta conexão informando se a origem dos dados existe. Caso não seja encontrado o banco de dados informado, o analista deverá corrigir o cadastro desta conexão com o sistema legado. Caso o banco de dados exista o sistema irá gravar a estrutura de entidades existente no banco de dados legado. Na próxima etapa o analista deverá cadastrar as regras de extração do banco de dados. Para isso usará os dados obtidos com a leitura das entidades do sistema legado realizados no processo anterior.

Definidas as regras de extração o analista deverá associá-las com os campos que serão extraídos das entidades lidas do sistema legado, realizando o cadastro de mapeamento da extração de dados. Este mapeamento será associado a um processo de carga de dados. Com o

cadastro dos mapeamentos de extração, o analista poderá executar o processo de extração de forma independente dos outros processos da ETL, para fins de testes e validações das suas regras. Caso opte por realizar o processo de extração neste momento, o sistema irá executar os *scripts* de extração de dados e realizando o armazenamento das informações extraídas nas estruturas homogêneas do Freedom. A seguir tem-se a figura 4, que demonstra o processo de extração de dados de um DW.

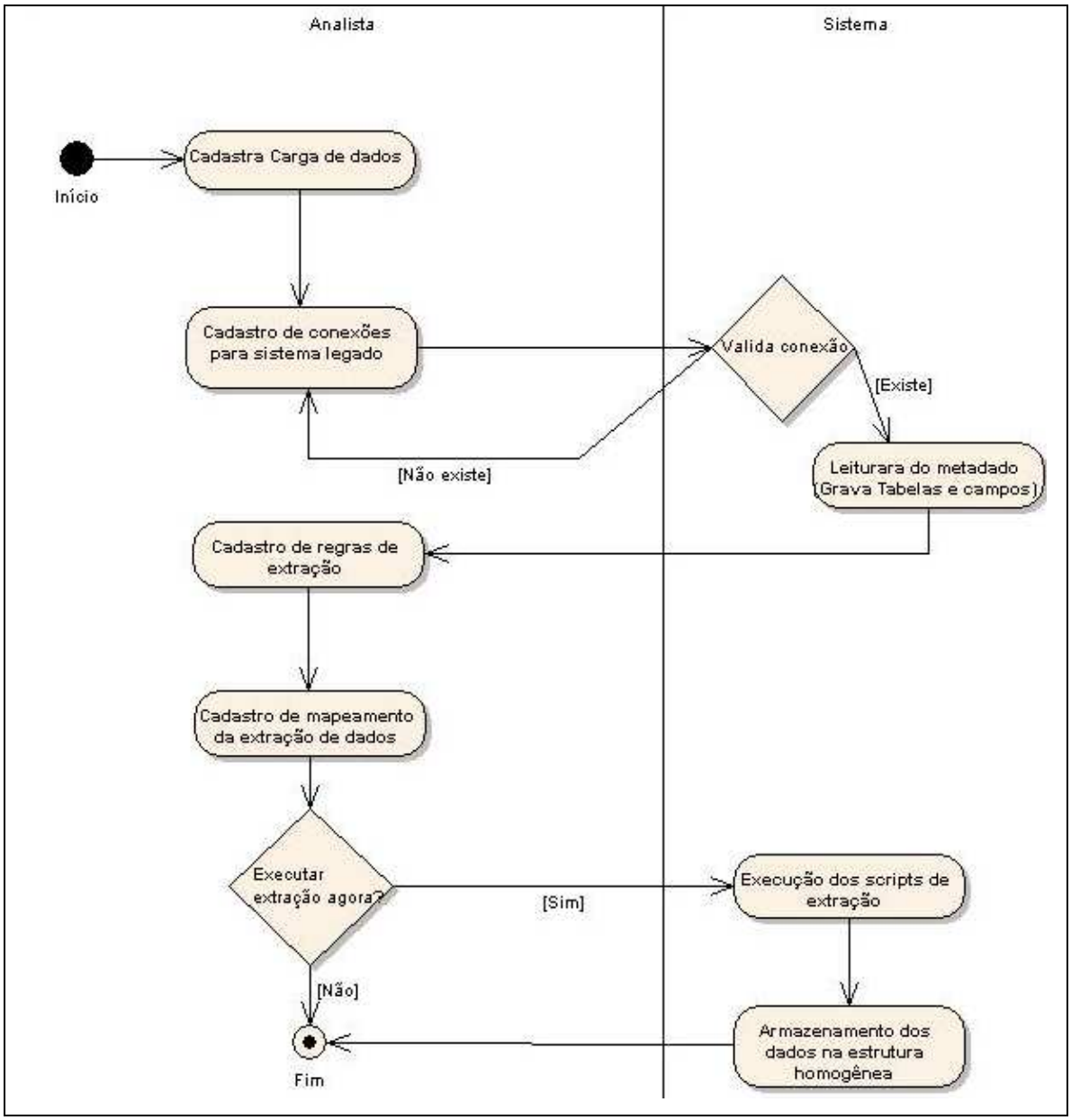


Figura 5 – Diagrama de atividades de mapeamento da extração de dados

Na figura 6, tem-se o diagrama de atividades está ilustrando o fluxo de atividades a serem realizadas pelo analista para mapear o processo de transformação dos dados previamente armazenados na estrutura homogênea de dados.

Este processo tem início com o cadastro do processo de transformação de dados que será realizado no processo de carga.

No processo seguinte, o analista irá definir as regras de conversão e integração dos dados armazenados na estrutura homogênea. O analista pode optar executar o processo de conversão e integração dos dados de forma independente dos outros processos, para fins de testes e validações das regras cadastradas. Caso opte por realizar a conversão de dados, o sistema irá executar os *scripts* necessários para realizar a integração dos dados.

Em seguida, o analista deverá cadastrar o modelo lógico de transformação dos dados, onde serão definidas as configurações principais do processo de transformação. A próxima etapa é o cadastro das regras de transformação dos dados, onde o analista irá realizar operações que irão modificar os dados extraídos para inseri-los no DW.

Para finalizar o processo de mapeamento da transformação de dados, o analista deverá associar as regras de transformação de dados a um processo de transformação previamente cadastrado, realizando o cadastro das unidades lógicas, que contém as definições das etapas da transformação dos dados. A figura 6 contém o diagrama de atividades do processo de transformação dos dados.

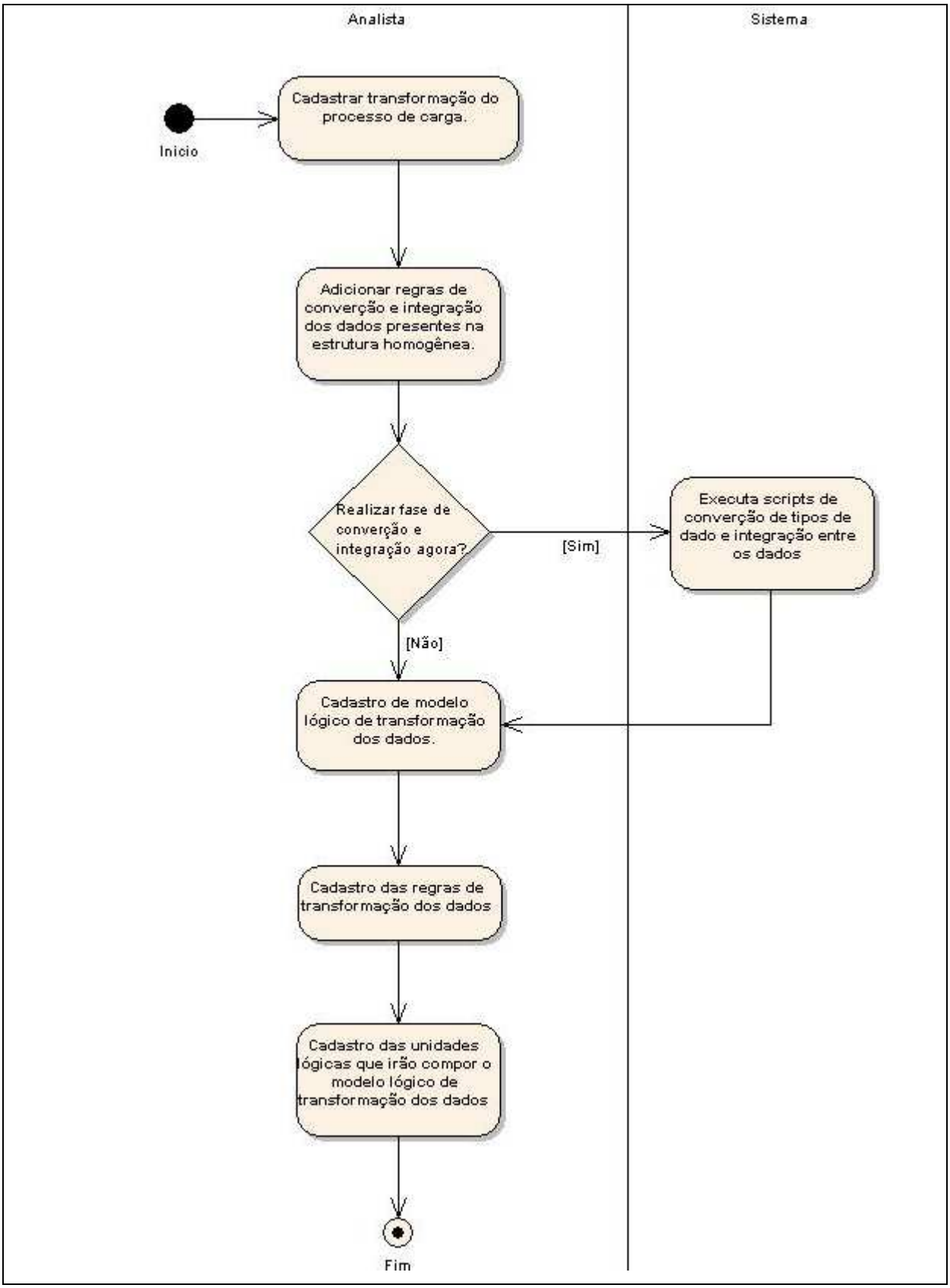


Figura 6 – Diagrama de atividades do mapeamento do processo de transformação dos dados

Na figura 7 tem-se o diagrama de atividades, tem-se o fluxo de atividades realizadas durante a etapa de carga de dados.

Este processo é realizado pelo sistema, que utiliza os dados previamente cadastrados pelo analista para montar os *scripts* de extração, transformação e carga de dados em um DW.

O processo tem início com o sistema realizando a leitura das configurações do processo de carga de dados. O sistema verifica se há registros pertinentes a este processo de carga na estrutura homogênea. Caso não haja dados, o sistema irá identificar que existe a necessidade de executar os *scripts* de extração de dados. No caso de já existirem dados, o sistema faz a verificação das configurações da carga em execução onde o analista define se os dados previamente armazenados neste processo devem ser apagados ou devem ser reutilizados.

Verificando a necessidade do processo de extração, o sistema executa os *scripts* para extrair os dados do sistema legado e os armazena na estrutura homogênea.

Ao concluir a extração de dados, o sistema irá verificar a necessidade de executar os *scripts* de conversão e integração dos dados. No próximo passo sistema executa os *scripts* de transformação, armazenando os dados em uma estrutura temporária, para que em seguida sejam inseridos como registros na tabela de fato do DW.

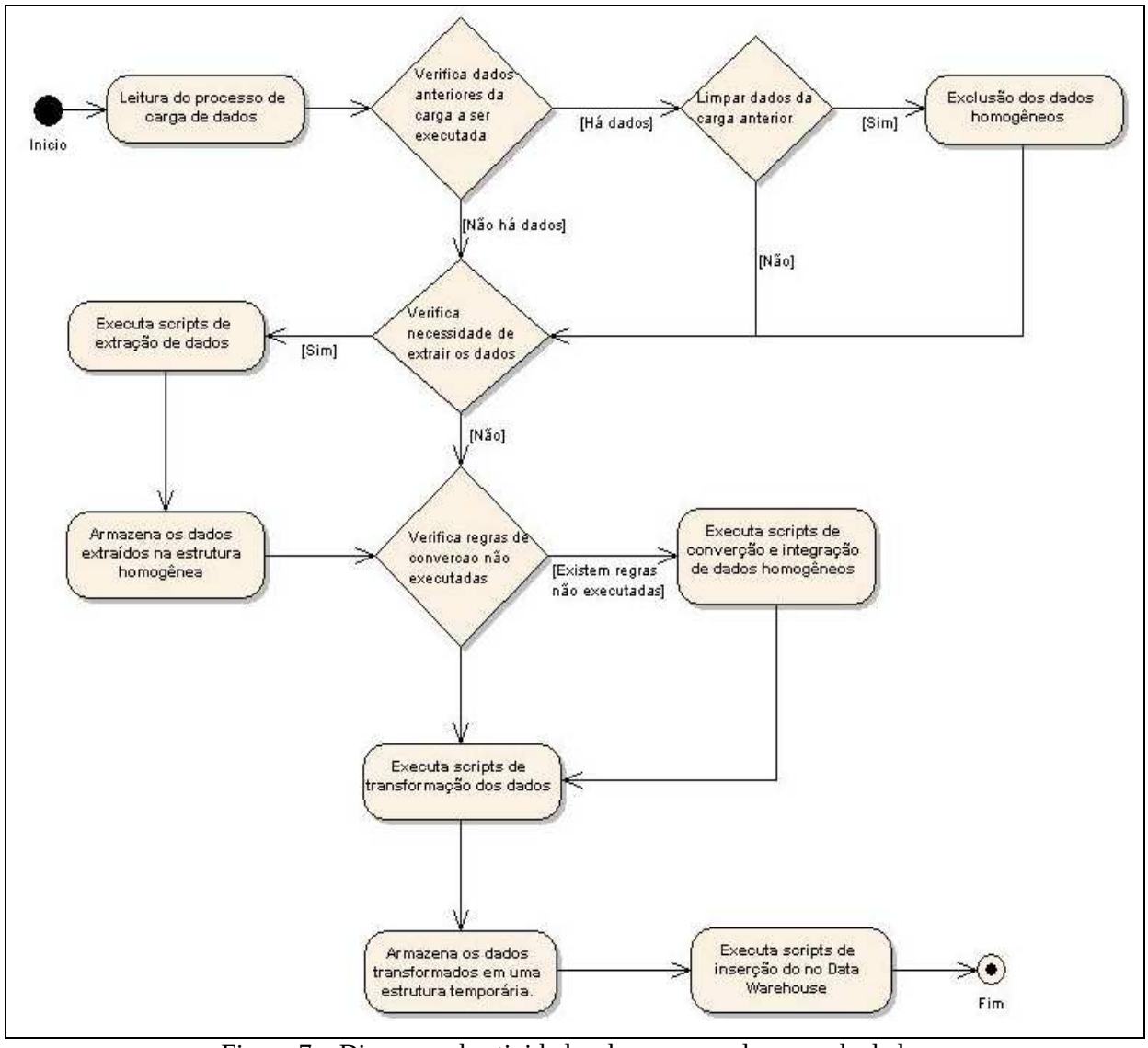


Figura 7 – Diagrama de atividades do processo de carga de dados

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para iniciar o desenvolvimento do sistema Freedom, foi desenvolvido um *framework*

para auxiliar no desenvolvimento das funcionalidades de uma aplicação web orientada a objetos. A maioria das horas trabalhadas no desenvolvimento desta aplicação foi dedicada à construção e manutenção do *framework.* Este *framework* web foi desenvolvido na linguagem de programação C#, usando o ambiente de desenvolvimento Microsoft Visual Studio 2008 em conjunto com o Framework .Net 3.5.

A seguir serão apresentadas todas as tecnologias utilizadas para o desenvolvimento do *framework* que auxiliou na geração do sistema Freedom. Este *framework* foi construído com base em um modelo genérico, e pode ser usado para colaborar na construção de outras aplicações web.

3.3.1.1 Construção de aplicações web em camadas

A construção de aplicações em camadas traz uma série de vantagens, sendo que a principal é a reutilização do código. Este tipo de implementação faz com que programadores se preocupem com os problemas de forma isolada, resolvendo uma questão de cada vez. Tratar os problemas de forma isolada garante códigos cada vez mais genéricos, aumentando consideravelmente a sua reutilização (ROB; HANSELMAN; HAACK, 2009).

O *framework* desenvolvido para implementação da aplicação desenvolvida neste trabalho foi dividido em três camadas. Estas camadas tem embasamento teórico no livro Asp.Net MVC 1.0, que propõe uma série de camadas para o desenvolvimento de aplicações.

A primeira camada de acesso aos dados tem o nome de *Data Access Layer* (DAL). Nesta camada estão implementadas todas as classes e métodos de acesso a um banco de dados. A base para o seu desenvolvimento está na utilização de um *framework* específico para mapeamento relacional de objetos, o NHibernate. Todas as transações e pesquisas que serão executadas no banco de dados da aplicação usarão objetos instanciados pelas classes da biblioteca DAL.

Na sequência, foi desenvolvida a camada de regras de negócio, a *Business Logic Layer* (BLL), ou camada lógica de negócios. Esta camada consegue enxergar todas as classes da camada de acesso a dados (DAL), e é responsável pela implementação das regras de negócio da aplicação. Esta biblioteca possui uma classe genérica na qual as demais classes que pertençam a BLL deverão ser filhas (conceito de herança).

Na classe genérica da BLL estão todos os métodos para recuperação de informações de um objeto. Cada classe criada na BLL que herde da classe genérica é fortemente “tipada”. É

necessário declarar que tipo de objeto (CDL) será o retorno das pesquisas no banco de dados.

A *Class Definition Layer* (CDL), ou camada de definição de classes, é a camada responsável por manter objetos que irão definir a regra de negócio a ser implementada. São as classes “POJO” do Java, classes estas que contém as definições de um objeto, métodos para recuperar suas propriedades e métodos para alterar as suas propriedades. Possuem membros privados e propriedades públicas para acesso externo. Esta biblioteca possui uma classe genérica na qual todas as classes de persistência devem herdar. Esta classe implementa a propriedade de identificador único do banco de dados, sendo a maior referência para o mapeamento do objeto relacional.

Por fim tem-se a camada de visualização, que tem por objetivo facilitar a construção de formulário no ambiente web. Para geração das telas da aplicação são usadas três tecnologias: o Asp.Net, para construção de páginas específicas; o Aquarium Framework para a geração de formulários padrões; e por fim o Coolite, que se trata de um *toolkit* com uma série de componentes Ajax para deixar a aplicação web mais próxima de uma aplicação *desktop*, além de proporcionar o desenvolvimento de *interfaces* ricas.

3.3.1.2 NHibernate

Trata-se de uma ferramenta de mapeamento objeto/relacional. É uma versão para a plataforma .Net Framework do *framework* Hibernate, para Java. A grande funcionalidade proporcionada pelo NHibernate é a capacidade de transformar em objetos os dados da estrutura lógica que estão armazenadas em um banco de dados (SCHENKER, 2009) .

Isso possibilita a redução de código necessário para realizar acesso a banco de dados, seja nas consultas ou nas transações. Outro fator importante para a redução de código é a redução do uso de *Structured Query Language* (SQL) no código da aplicação. Ao invés disso, o NHibernate possui uma linguagem própria muito semelhante ao SQL, o *Hibernate Query Language* (HQL), acelerando a velocidade do desenvolvimento de métodos de acesso a banco (SCHENKER, 2009).

O NHibernate da suporte a uma persistência de dados transparente, ou seja, as classes não precisam seguir nenhum modelo de restrição, como implementar *interfaces* ou herdar de classes específicas. Isso contribui para que toda a regra de negócio da aplicação se concentre na camada de negócio implementada na tecnologia Microsoft .Net. Outra grande vantagem do uso do NHibernate é pelo fato de ser compatível com quase todos os bancos de dados

existentes hoje no mercado, facilitando a migração de tecnologia quando e se for necessário (SCHENKER, 2009).

3.3.1.3 Aquarium Framework

Conforme consta no endereço eletrônico da empresa Code OnTime, o Aquarium Framework é um gerador de formulários web para aplicações desenvolvidas com a tecnologia Microsoft.Net. Esta ferramenta está disponível para *download* no site "www.codeontime.com", possuindo duas versões, uma gratuita e outra paga. A versão usada para construção do *framework* de desenvolvimento da aplicação deste trabalho é a versão gratuita (CODE ONTIME, 2009).

O principal benefício trazido com o uso da ferramenta Aquarium é a velocidade no desenvolvimento de aplicações. Esta ferramenta possui um gerador de formulários, criando páginas que contemplam todas as funcionalidades básicas que um formulário de uma aplicação web deve possuir a inserção, a alteração e a exclusão de registros (CODE ONTIME, 2009).

Os formulários são gerados com base na estrutura do banco de dados da aplicação, gerando um formulário por tabela. Os formulários são arquivos *eXtensible Markup Language* (XML), totalmente configuráveis. Estes documentos são interpretados pelo núcleo do Aquarium em tempo de execução, que usa componentes Ajax e do Microsoft Framework .Net 3.5, para montar as páginas na web (CODE ONTIME, 2009).

3.3.1.4 Coolite

Trata-se de um conjunto de ferramentas para auxiliar no desenvolvimento de aplicações web ricas. Baseado no *framework* Ext, escrito completamente em na linguagem Javascript, o Coolite tem como principal objetivo tornar a *interface* de sites da internet desenvolvidos com a tecnologia Microsoft .Net ricas e fáceis de serem usadas pelos seus usuários (COOLITE INK, 2009).

Esta ferramenta pode ser encontrada no endereço "www.coolite.com". Possui uma versão paga e uma versão comunitária. A versão usada para o desenvolvimento do *framework* usado na construção da aplicação deste trabalho é a versão comunitária.

3.3.1.5 SQL Server Analysis Services

Conforme Jacobson e Stacia (2007), SQL Server Analysis Services (SSAS) é uma ferramenta desenvolvida pela empresa Microsoft, que implementa o processamento analítico *online* (OLAP), utilizada para integração de dados relacionais. Através do SSAS é possível criar soluções BI.

Usando a ferramenta SQL Server Managemet Studio 2005 (SSMS), desenvolvida pela empresa Microsoft é possível a conexão com os serviços fornecidos pela instalação do SSAS, possibilitando a criação dos cubos de decisão, mapeamento do DW e definição das medidas e dimensões. Esta ferramenta possui o *Cube Browser*, que possibilita a visualização dos dados dimensionais e operações pertinentes ao OLAP. Esta ferramenta também disponibiliza o processamento dos cubos de decisão, ou seja, leitura das alterações ocorridas no DW para que seja possível sua manipulação por ferramentas OLAP que se conectem ao SSAS (JACOBSON, 2007).

3.3.2 Operacionalidade da implementação

Nesta seção será apresentada a sequência de telas e operações a serem realizadas pelo analista para executar de forma correta a carga de dados em um DW. Também são apresentados trechos de código fonte de algumas das principais funcionalidades do sistema.

3.3.2.1 Acesso ao sistema

Para acessar o sistema de carga de dados será necessário que o usuário esteja conectado a internet. Ao acessar o endereço do servidor no qual estará implantando, a tela de autenticação de usuário será apresentada pelo sistema conforme a figura 8.

Figura 8 – Tela de autenticação de usuário

Para ter acesso às funcionalidades do sistema o usuário deverá informar o seu nome de acesso e senha previamente cadastrados na aplicação. Após informar os dados solicitados pelo sistema o analista deverá clicar no botão “Login”. Ao clicar neste botão o sistema irá verificar se a pessoa que está tentando acessar o sistema possui permissão. Caso ela não tenha acesso ao sistema, será disparada a mensagem “Usuário ou senha incorreto”. A mesma mensagem será apresentada em casos de erro no nome do usuário ou na senha informada. No caso de existir os dados informados, o sistema irá redirecionar o usuário à página principal da aplicação.

3.3.2.2 *Menu* principal da aplicação

A página principal da aplicação contém uma barra de tarefas na parte inferior da página, com o botão “Iniciar”, e uma área que se estende até o final da página. Ao clicar no botão “Iniciar” serão apresentadas todas as funcionalidades da aplicação. Para maior comodidade do usuário, as principais funcionalidades possuem um atalho na página principal do sistema. Na figura 9 apresenta-se a tela principal.

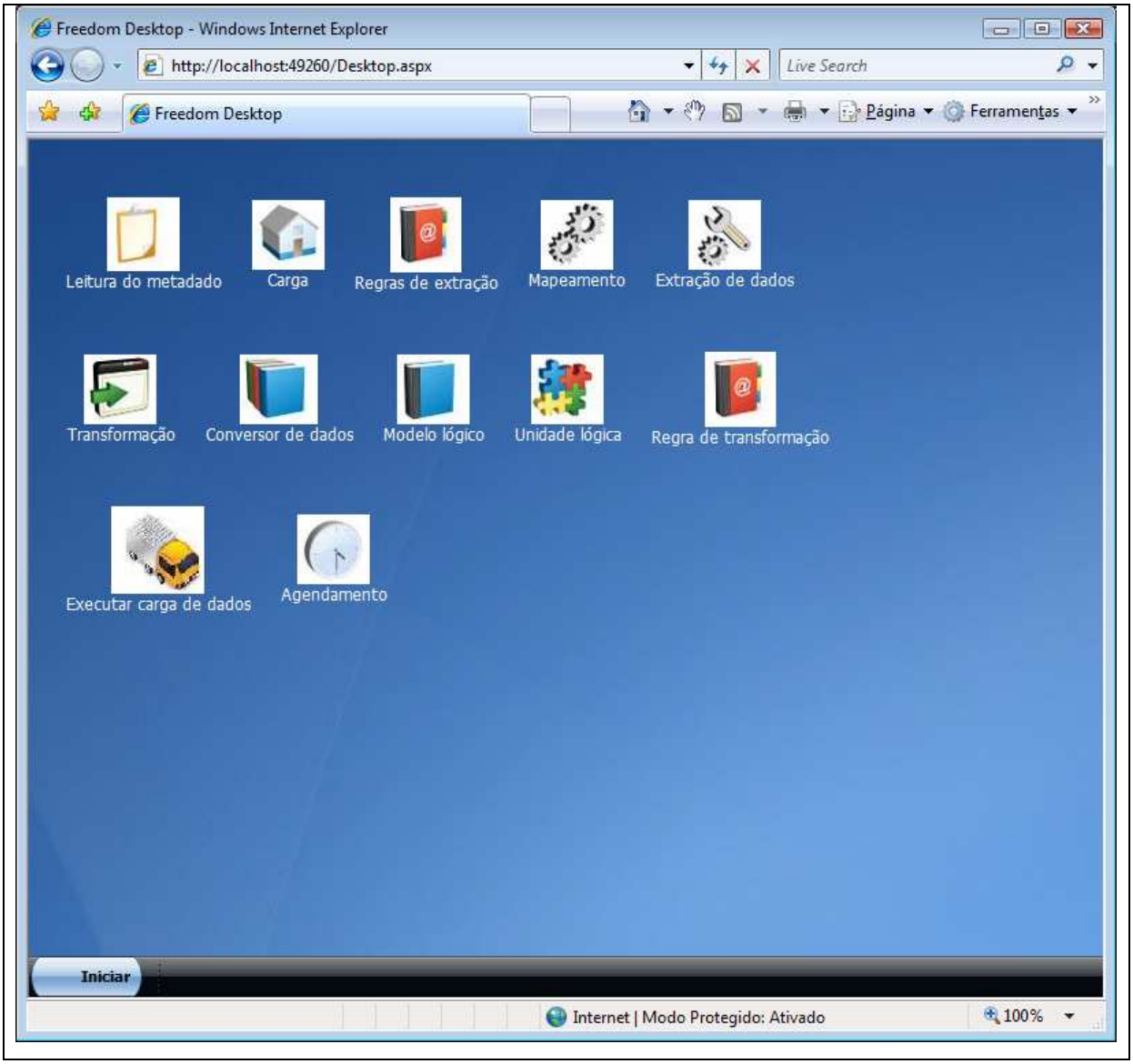


Figura 9 – Tela principal da aplicação em formato Desktop

Cada funcionalidade da aplicação possui uma tela própria. Estas telas serão apresentadas para o usuário em forma de janelas suspensas no ambiente da página principal, conforme a figura 10. Estas janelas poderão ser redimensionadas, minimizadas e armazenadas no painel inferior da página principal, podendo ser restauradas com um único clique em seu nome.

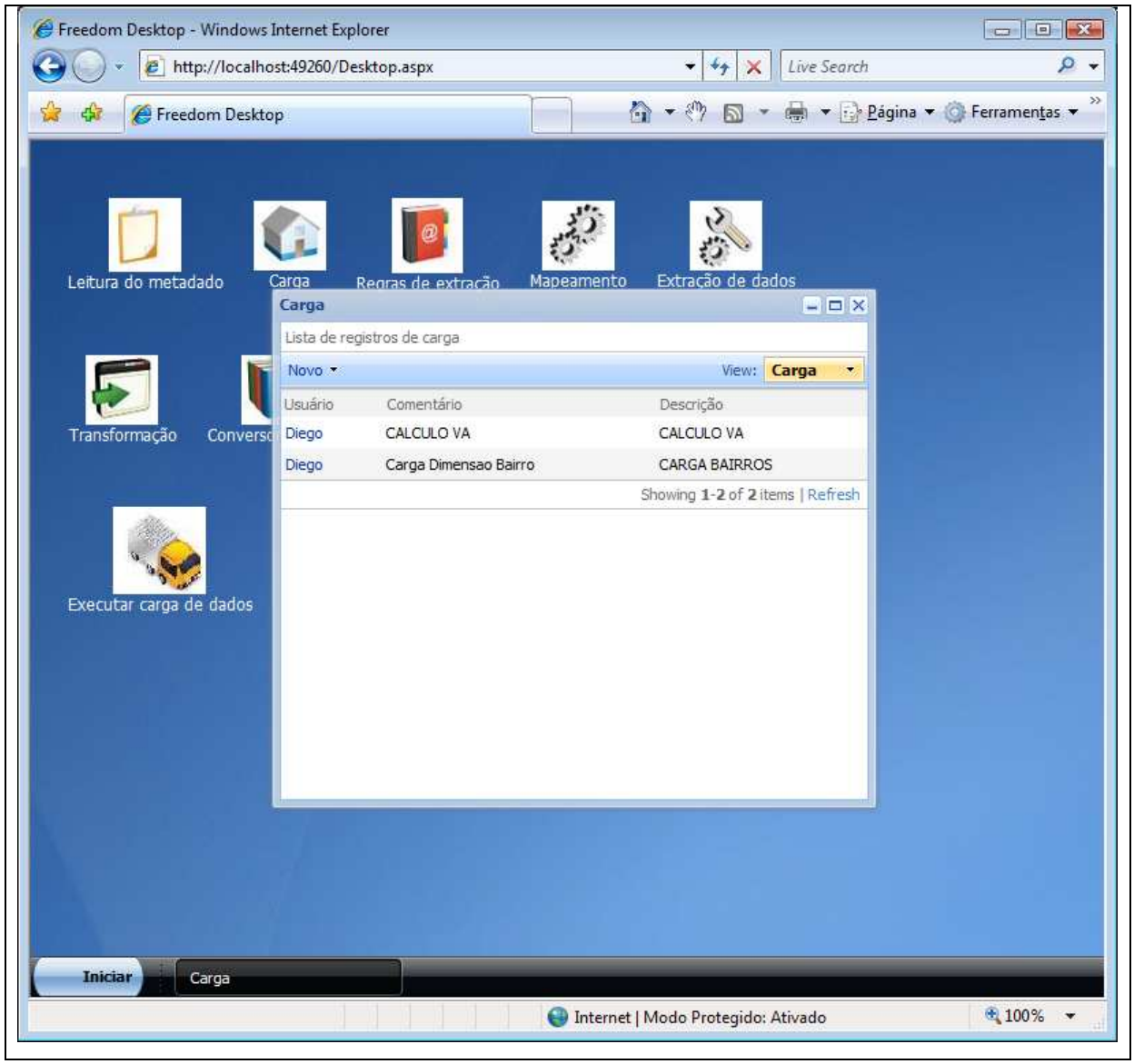


Figura 10 – Janelas suspensas na página principal

3.3.2.3 Cadastro de usuários e tipos de usuários

Para que outros analistas ou profissionais que trabalham com o DW possam ter acesso ao sistema, será necessário o cadastro de usuário. Para isso, basta acessar o *menu* de cadastros no sistema, clicando no botão “Iniciar” da página principal da aplicação, selecionar o *menu* “Cadastros”, selecionar o sub-*menu* “Usuário” e em seguida clica na opção “Usuário”, conforme apresenta-se na figura 11.

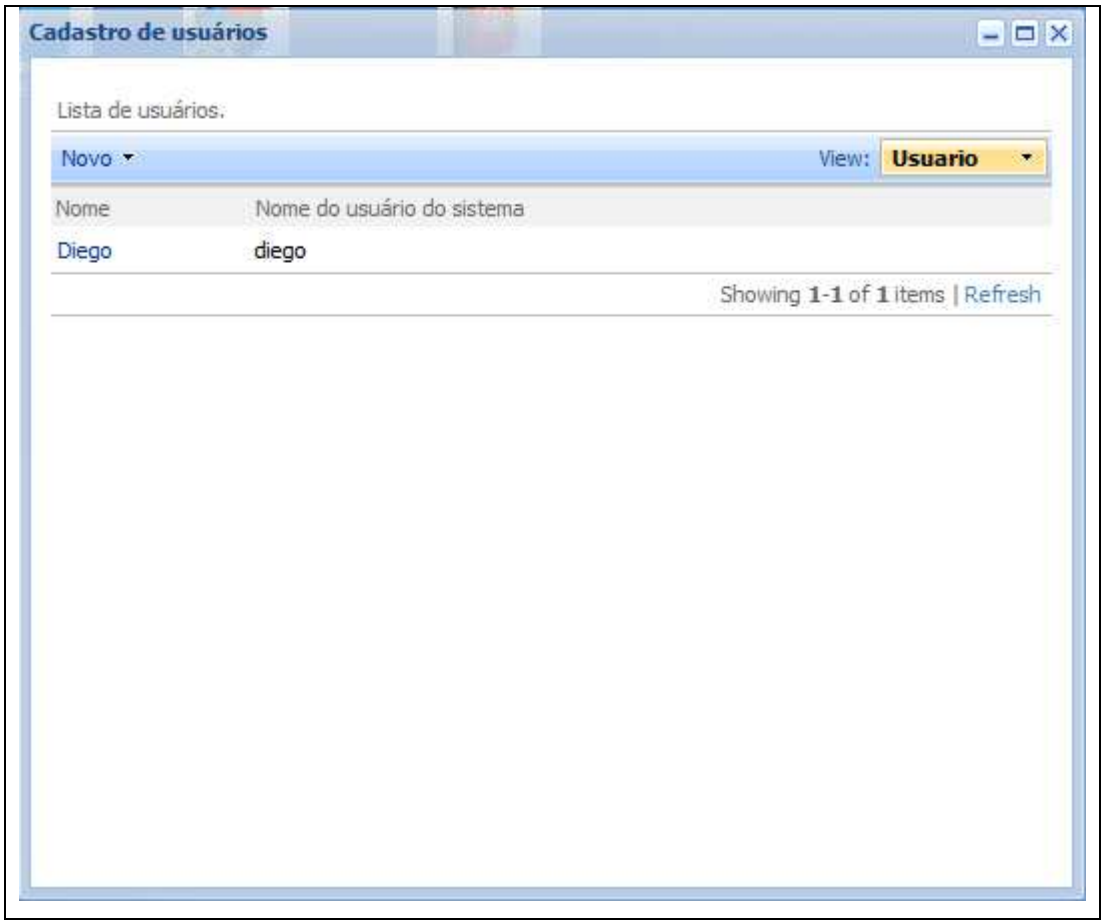


Figura 11 – Tela de cadastro de usuários do sistema.

Na janela será apresentada uma lista de todos os usuários previamente cadastrados na aplicação. Para inserir um novo usuário, clicar no botão “Novo”, na barra de tarefas na parte superior da página e selecionar a opção “Novo usuário”.

Um formulário será apresentado para seleção do tipo de usuário que será inserido, bastando selecionar um tipo previamente cadastrado. Caso o tipo de usuário desejado para o cadastro não esteja na lista, será necessário cadastrar o tipo de usuário. Será necessário informar o nome completo do usuário, um nome de acesso ao sistema (nome de usuário) e a senha de acesso. Preenchido todos estes campos, o botão “Ok” deverá ser acionado para que os dados sejam salvos, conforme pode-se visualizar na figura 12.

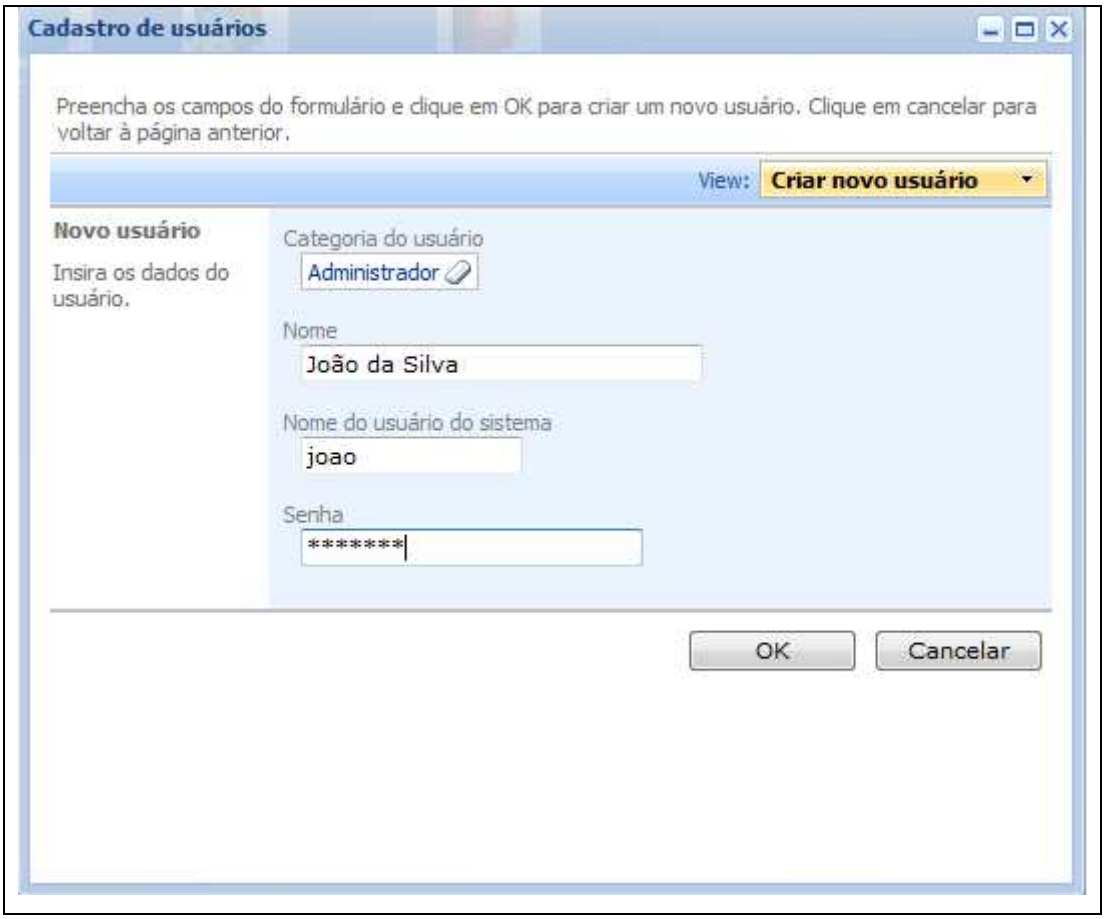


Figura 12 - Formulário para cadastro de um novo usuário

Param cadastrar um novo tipo de usuário devem-se seguir os mesmos passos usados para chegar até o cadastro de usuários, mas ao invés de selecionar a opção “Usuário”, selecionar a opção “Tipo de usuário”. Seguir os mesmo procedimentos usados para o cadastro de usuário.

3.3.2.4 Cadastro do modelo de carga de dados

Para iniciar o processo é necessário o cadastro da carga de dados propriamente dita. Para acessar a tela de cadastro da carga de dados, ir ao *menu* “Iniciar”, selecionar a opção “Cadastros”, opção Carga e clicar no item de *menu* “Carga de dados”. Será apresentada uma tela contendo todas as cargas de dados previamente cadastradas. Para inserir um novo processo de carga de dados, clicar em “Novo”, selecionar a opção “Nova carga de dados”. O sistema então será redirecionado para o formulário de cadastro de carga de dados. A figura 13 apresenta o formulário de cadastro da carga de dados.

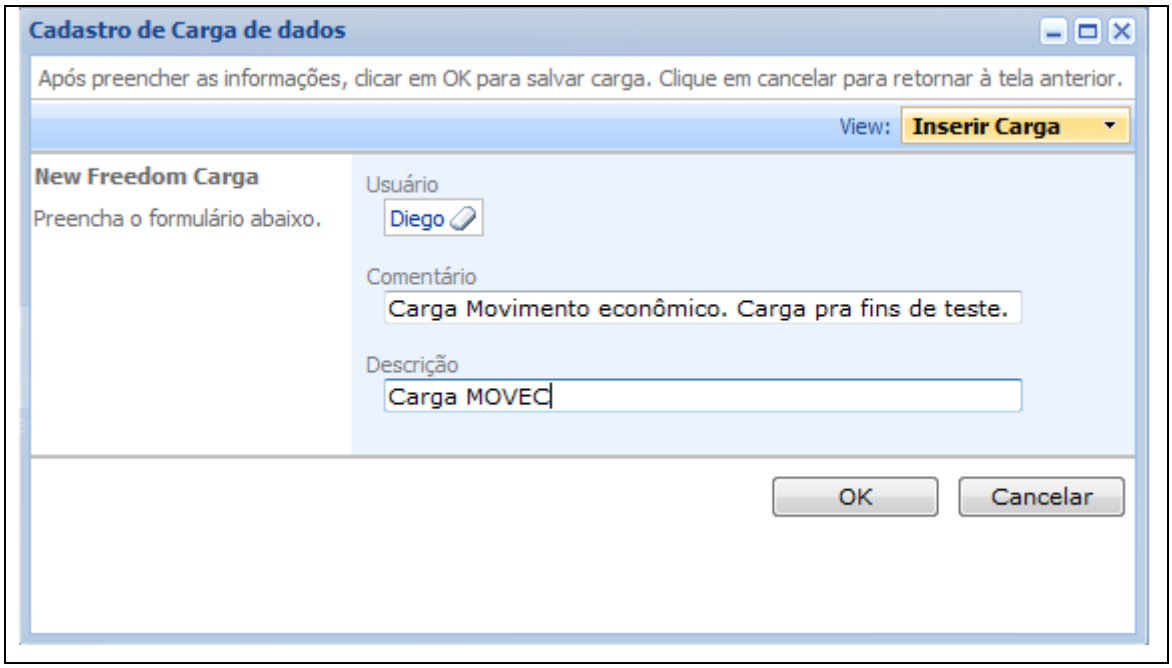


Figura 13 – Cadastro de carga de dados

Trata-se de um cadastro simples, onde será necessário informar apenas a descrição da carga de dados, sendo que este nome será usado para identificar o processo ao longo das demais funcionalidades da aplicação e as observações importantes referentes ao processo de carga. A partir daqui o usuário do sistema passa a alimentar o metadados do DW.

A classe que representa os processos de carga de dados do DW é muito simples, contendo as propriedades necessárias para recuperar os dados inseridos no banco de dados, conforme descrito no passo acima, e duas coleções de informações. Uma coleção de mapeamentos de processo de extração e uma coleção de transformações que os dados que serão carregados sofrerão ao longo do processo da carga. A seguir tem-se o quadro 3, contendo a definição da classe que representa a carga de dados.

```
public class CargaCDL : ObjectCDL
    {
        public override string ToString()
        {
            return _descricao;
        }

        private UsuarioCDL _usuario;
        private string _descricao;
        private string _comentario;
        private IList<MapeamentoCDL> _mapeamentos;
        private IList<TransformacaoCDL> _transformacoes;//so terá uma

        public virtual IList<TransformacaoCDL> Transformacoes
        {
            get { return _transformacoes; }
            set { _transformacoes = value; }
        }

        public virtual IList<MapeamentoCDL> Mapeamentos
        {
            get { return _mapeamentos; }
            set { _mapeamentos = value; }
        }

        public virtual string Comentario
        {
            get { return _comentario; }
            set { _comentario = value; }
        }

        public virtual String Descricao
        {
            get { return _descricao; }
            set { _descricao = value; }
        }

        public virtual UsuarioCDL Usuario
        {
            get { return _usuario; }
            set { _usuario = value; }
        }

    }
```

Quadro 3 – Classe que representa o processo de carga de dados em um DW

3.3.2.5 Cadastro de conexões com sistemas legado

Será através de conexões com outros bancos de dados que o sistema Freedom irá executar o processo de extração dos dados. Para cadastrar o acesso a outros bancos, acionar o *menu* “Iniciar”, “Cadastros”, “Banco de dados legado” e clicar em Conexões. Uma tela com

as conexões existentes será apresentada. Para cadastrar uma nova conexão, clicar em “Novo”, “Nova conexão”. A figura 14 apresenta o formulário de cadastro de banco de dados.

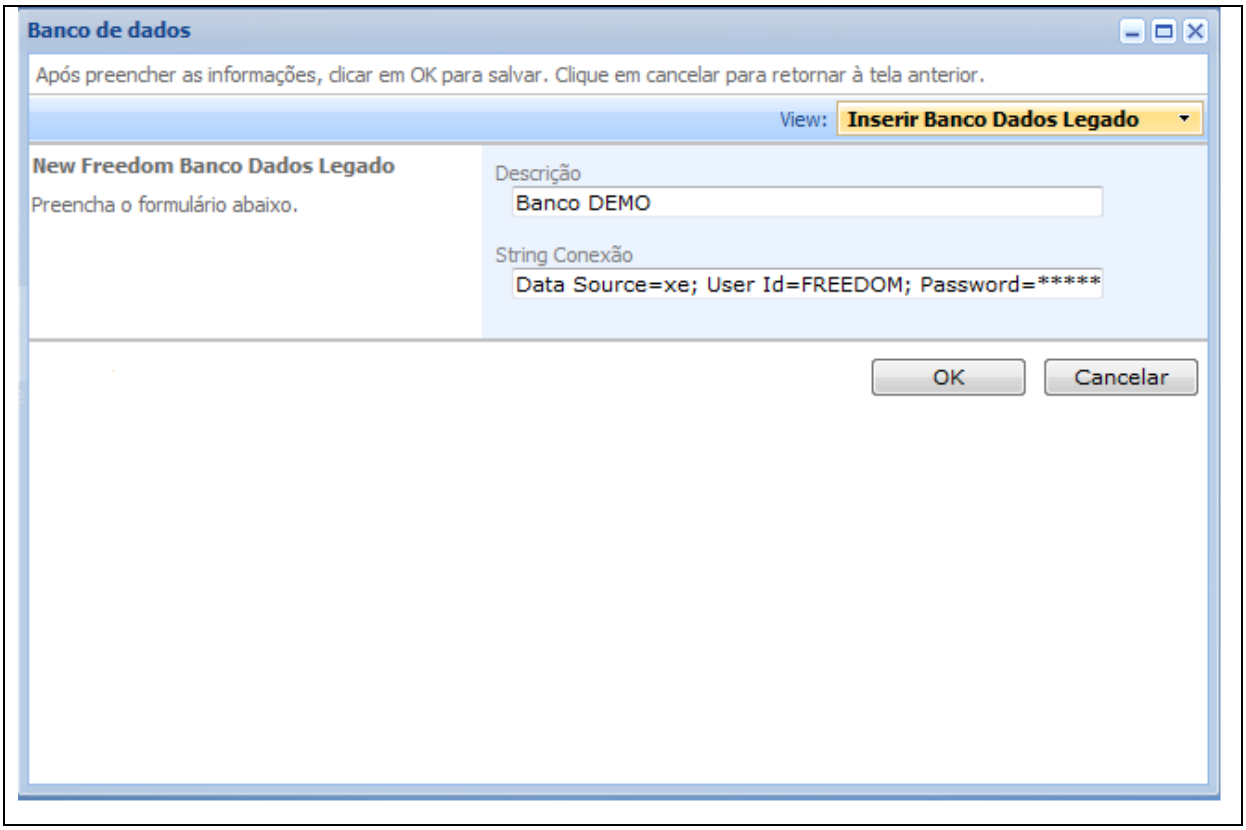


Figura 14 – Criando conexão com banco de dados legado

A informação mais importante neste formulário é a string de conexão com o banco de dados Oracle, lembrando que nesta versão de sistema, o Freedom se conectará somente a banco de dados Oracle. Será com as informações da *string* de conexão que o sistema irá manter as conexões múltiplas com banco de dados.

3.3.2.6 Leitura da estrutura de dados do sistema legado

Para que o processo de carga de dados seja realizado com sucesso é de fundamental importância que as estruturas de dados dos sistemas legados, ou seja, sistema que vai fornecer os dados para o DW (sofrerá o processo de extração) seja conhecido.

Para realizar esta etapa, deve estar muito claro para o usuário do sistema quais os dados serão extraídos do sistema legado, transformados e incluídos no DW, portanto, tanto o conhecimento da estrutura de dados do ambiente legado quanto do ambiente do DW serão necessários para efetuar a extração.

Para fazer com que o Freedom conheça a estrutura do sistema legado é necessário que

seja feita a sua leitura. Para realizar este processo é necessário acessar a tela de leitura de metadados, clicando no ícone “Leitura metadado” na tela principal da aplicação ou acionando o *menu* “Iniciar”, e clicar no item de *menu* de mesmo nome. A seguir tem-se a figura 15, apresentando a tela de leitura do metadado do sistema legado.

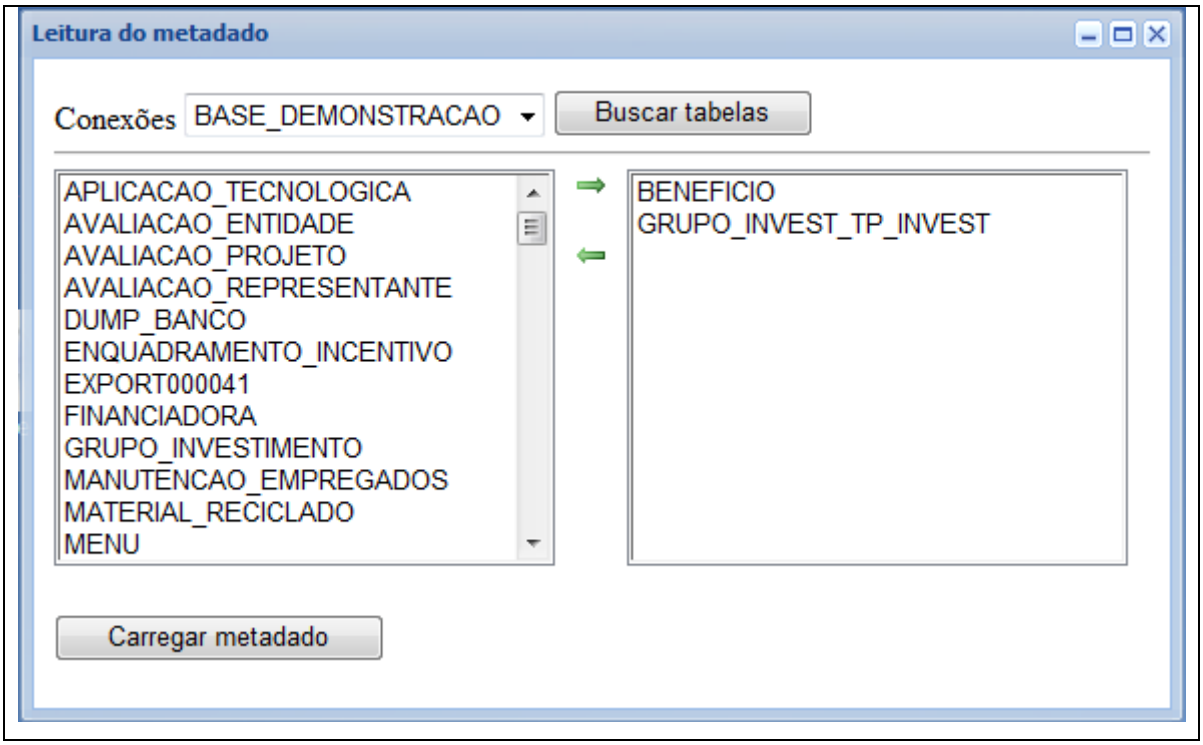


Figura 15 – Tela de leitura de metadados

No canto superior da tela de, tem-se uma caixa de seleção das conexões previamente cadastradas no sistema. Ao selecionar a conexão desejada, deve-se clicar no botão “Buscar tabelas” para que o sistema execute o algoritmo de leitura das tabelas existentes na aplicação.

No quadro 4, tem-se o código usado pelo sistema para realização da leitura das tabelas de um banco de dados.

```
        /// <summary>
        /// Cria as tabelas passadas como parâmetrow
        /// </summary>
        /// <param name="pListaTableOwner"></param>
        public List<TabelaCDL> GeraTabelasAplicacao(NameValueCollection
                               listaTableOwner, Boolean gerarDependencias)
        {
            //Cria as tabelas
            List<TabelaCDL> listaRetorno = new List<TabelaCDL>();
            foreach (String table in listaTableOwner.AllKeys)
            {
                AddTable(listaTableOwner[table], table, listaRetorno,
                  gerarDependencias);
                //Atribue tabelaLookup nas colunas que são chave
                  estrangeira
                AttributeForeignKeys(listaRetorno);
            }

            return listaRetorno;
```

```
        }
        private TabelaCDL AddTable(string ownerName, string tableName,
                        List<TabelaCDL> listaTabelas, bool
gerarDependencias)
        {
            TabelaCDL sisTabela = BuscaTabelaLista(tableName,
            listaTabelas);

            if (sisTabela == null)
            {
                DataTable DTTabelas = GetDataTable("tables", new string[]
                  { ownerName, tableName });
                //Instancia a lista de tabelas
                sisTabela = new TabelaCDL();
                sisTabela.Campos = new List<CampoCDL>();

                foreach (DataRow row in DTTabelas.Rows)
                {
                    sisTabela.Nome = Convert.ToString(row["TABLE_NAME"]);
                    sisTabela.Owner = Convert.ToString(row["OWNER"]);
                }

                //Carrega demais campos da tabela
                List<CampoCDL> camposTabela = GetTableFields(sisTabela,
                  gerarDependencias, listaTabelas);
                foreach (CampoCDL campo in camposTabela)
                {
                    campo.Tabela = sisTabela;
                    sisTabela.Campos.Add(campo);
                }
                listaTabelas.Add(sisTabela);
            }

            return sisTabela;
        }
```

Quadro 4 – Código fonte para leitura dos metadados das tabelas

Ao ser feita a leitura das tabelas, o sistema irá armazenar os nomes das tabelas encontradas em uma caixa de seleção, situada na parte esquerda da tela. O usuário fará a seleção das tabelas que desejar clicando sobre o nome da tabela, e pressionando o botão em forma de seta, apontando para a direita. Isso fará com que o nome da tabela selecionada apareça na caixa de seleção da esquerda, áreas destinadas às tabelas que farão parte do cadastro de tabelas de sistema legado do Freedom. Para selecionar mais de uma tabela por vez, basta deixar pressionando a tecla “Ctrl” do teclado e clicar nas tabelas que serão importadas para o sistema. Caso seja necessário remover alguma tabela selecionada, basta clicar no nome da tabela da caixa de seleção da direita, e clicar no botão que possui uma seta apontando para a esquerda.

Após selecionar todas as tabelas, deve-se clicar no botão “Carregar metadados”, para que o sistema execute o algoritmo completo de leitura da estrutura do banco de dados. As

partes mais relevantes do código de leitura da estrutura do banco de dados estão presentes no quadro 5.

```
private List<CampoCDL> GetTableFields(TabelaCDL sisTabela, bool
                  gerarDependencias, List<TabelaCDL> listaTabelas)
        {
            List<CampoCDL> TableFields = new List<CampoCDL>();

            //Carregando a chave primária
            CampoCDL campoPK = GetPrimaryKey(sisTabela.Owner,
            sisTabela.Nome, sisTabela);
            if (campoPK != null)
                TableFields.Add(campoPK);

            DataTable DTCampos = GetDataTable("Columns", new string[] {
                                    sisTabela.Owner, sisTabela.Nome });
            DataTable DTForeignKeysColumns =
            GetDataTable("ForeignKeyColumns", new string[]
            { sisTabela.Owner, sisTabela.Nome });
            foreach (DataRow campo in DTCampos.Rows)
            {
                if (BuscaCampoLista(campo["COLUMN_NAME"].ToString(),
                                                TableFields) == null)
                {
                    CampoCDL sisCampo = new CampoCDL();
                    sisCampo.Nome = campo["COLUMN_NAME"].ToString();
                    sisCampo.Ordem = int.Parse(campo["ID"].ToString());
                    sisCampo.PermiteNulo =
                    ConverteBool(Convert.ToChar(campo["NULLABLE"]));
                    sisCampo.Tipo =
                    ConverteTipoDado(Convert.ToString(
                    campo["DATATYPE"]), Convert.ToInt32(campo["LENGTH"]));
                    sisCampo.Tabela = sisTabela;

                    foreach (DataRow foreignKeyColumn in
                        DTForeignKeysColumns.Rows)
                    {
                        //Procura campo que é chave estrangeira
                        if (foreignKeyColumn["COLUMN_NAME"].
                                    Equals(campo["COLUMN_NAME"]))
                        {
                            sisCampo.ChaveEstrangeira = true;
                            sisCampo.NomeConstraint = Convert.ToString(
                            foreignKeyColumn["CONSTRAINT_NAME"]);
                            break;
                        }
                        sisCampo.ChaveEstrangeira = false;
                    }

                    if (sisCampo.ChaveEstrangeira)
                    {
                        DataTable DTForeignKeys =
                        GetDataTable("ForeignKeys",
                        new string[] { sisTabela.Owner,
                        sisTabela.Nome, sisCampo.NomeConstraint });
                        foreach (DataRow foreignKeys in
                              DTForeignKeys.Rows)
                        {
                            sisCampo.TabelaLookupNome =
                            foreignKeys["PRIMARY_KEY_TABLE_NAME"].
                                                    ToString();
                            sisCampo.AutoRelacionamento =
```

```
                            (sisCampo.Tabela.Nome ==
                                  sisCampo.TabelaLookupNome);

                            //GerarDependencia é false para que sejá
                              gerado apenas do primeiro nível
                            if (gerarDependencias)
                                AddTable(foreignKeys["R_OWNER"]
                                .ToString(), sisCampo.TabelaLookupNome,
                                listaTabelas, false);
                        }
                    }
                    TableFields.Add(sisCampo);
                }
            }
            return TableFields;
        }
```

Quadro 5 – Código fonte do método de leitura da estrutura do banco de dados

3.3.2.7 Alteração do da estrutura do banco de dados importado

Após a importação da estrutura do banco de dados, é possível verificar todo o conteúdo importando e alterá-lo. Para isso, basta acessar o *menu* “Iniciar” da tela principal do sistema, opção “Cadastros”, ir em “Base de dados legado” e clicar na opção “Tabelas”. Será apresentada uma tela semelhante às telas de Usuário e Tipo de usuário.

Para a alteração das informações importadas, basta clicar na primeira coluna da listagem de tabelas apresentada pelo sistema. Nota-se que aparece o símbolo de uma seta apontando para baixo na primeira célula da linha selecionada. Ao clicar nesta seta, o sistema irá mostrar as opções “Selecionar”, “Editar” e “Apagar”. Cada uma destas opções se propõe a fazer o que o nome sugere, sendo assim, basta clicar na opção “Editar” para que seja possível a alteração de um dos valores que compõem a tabela. A figura 16 apresenta a tela de cadastro em estado de edição dos dados.

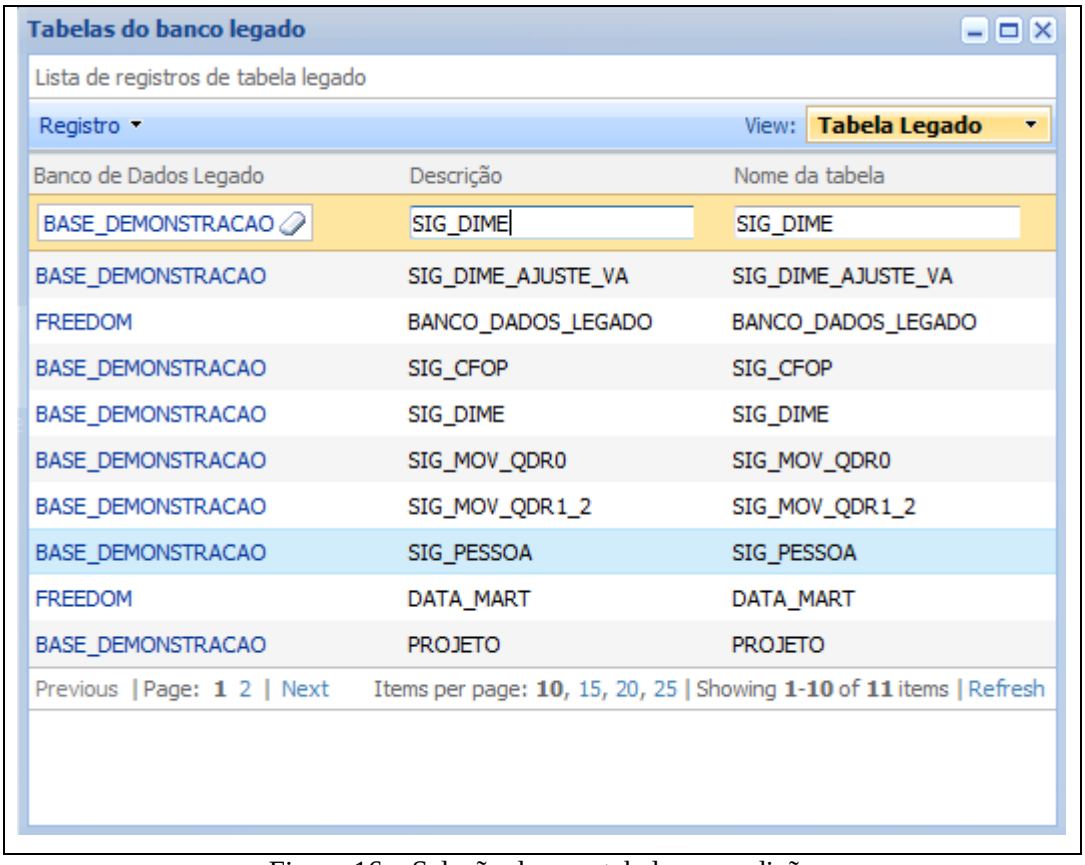


Figura 16 – Seleção de uma tabela para edição

Para a alteração dos campos das tabelas importadas, basta seguir os mesmos passos adotados para chegar até a página “Tabelas”, e selecionar a opção “Campos”. Esta tela contém todos os campos conhecidos pelo sistema Freedom. Para facilitar a o manuseio destas informações, basta usar as funcionalidades de filtro e ordenação presentes na listagem de tabelas fornecidas.

Para executar filtros na listagem de dados apresentada pelo sistema, basta que o usuário posicione o ponteiro do *mouse* no cabeçalho do campo que necessita ser filtrado, e clique na imagem de uma seta apontando para baixo, para que as opções de ordenação e filtro apareçam. A seguir tem-se a figura 17, que apresenta o menu de filtro e ordenação das listagens.

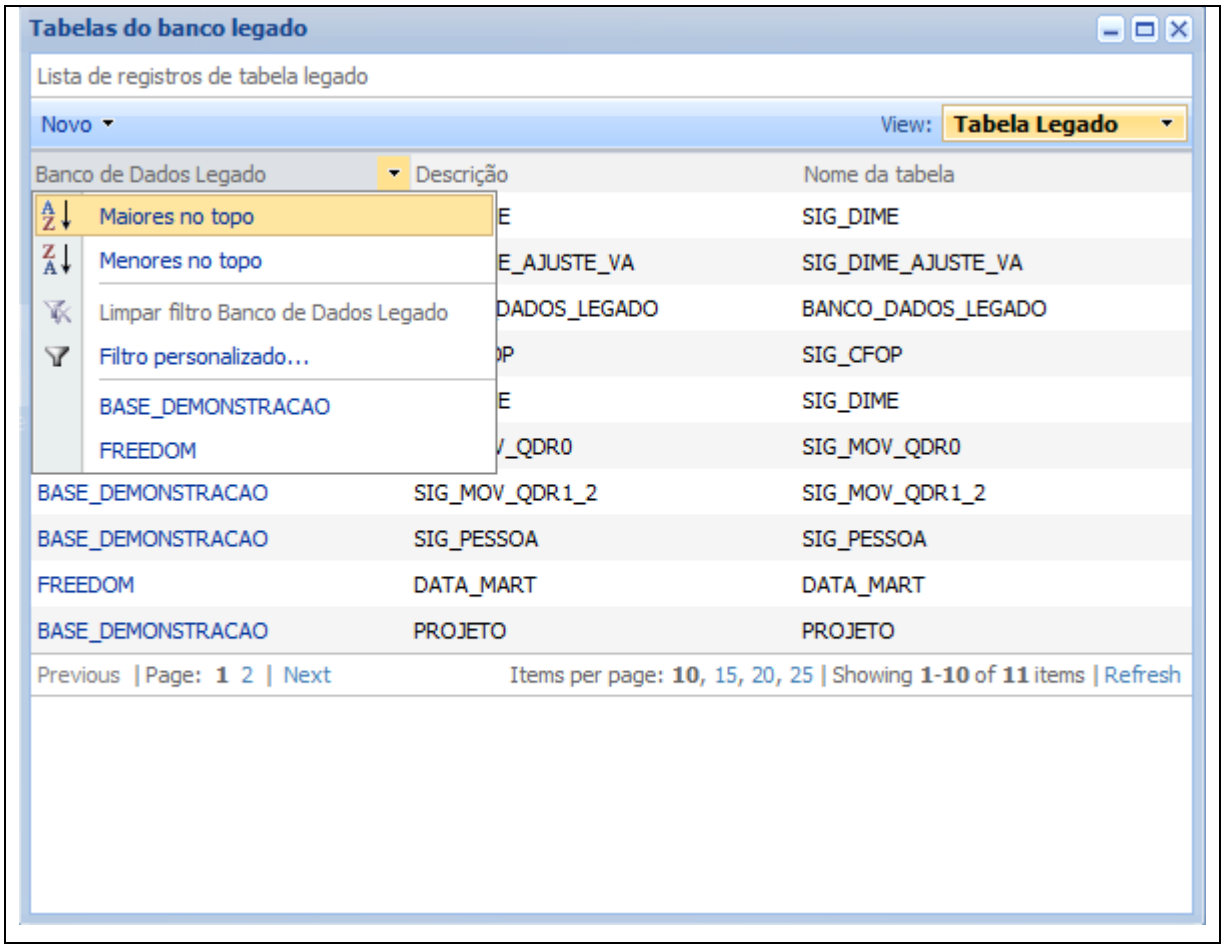


Figura 17 – Filtro e ordenação dos campos

O sistema trará as opções padrões de filtro, no entanto, o usuário poderá optar por customizar os filtros, clicando na opção “Filtro customizado”. A tela ilustrada na figura 18 será mostrada ao usuário, bastando que ele siga as instruções na tela para criar os seus próprios filtros.

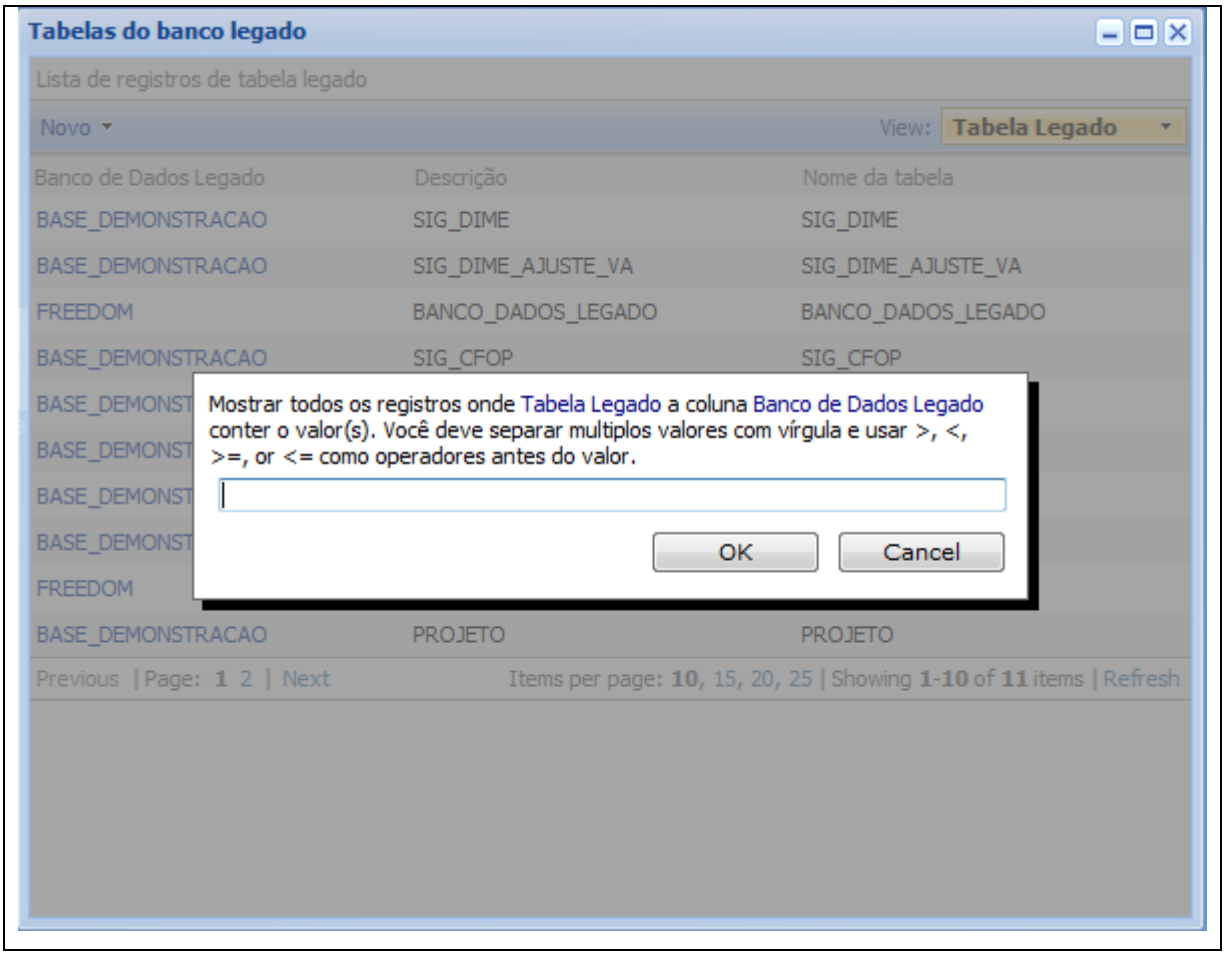


Figura 18– Customização do filtro

3.3.2.8 Cadastro de regras de extração dos dados

Uma vez que já são conhecidos do sistema as tabelas e seus respectivos campos que sofrerão o processo de extração de dados, o próximo passo a ser dados para realizar a fase de extração é a definição das regras de extração.

Para acessar a tela de cadastro de regras, basta clicar no ícone presente no *menu* principal do sistema cujo nome é “Regras de extração”, ou acessar o *menu* “Iniciar”, “Cadastros”, “Regras” e clicar em “Regras de extração”. Será apresentada a página para o cadastro das regras de extração. Para inserir uma nova regra, basta seguir os mesmo passos descritos no cadastro de usuários.

Nota-se o formulário de criação de novas regras de extração, no painel a esquerda do formulário contém uma série de instruções para a criação da regra. Ao chegar neste ponto, o usuário do sistema deve ter muito claro o significado dos dados que deseja extrair do sistema

legado, e como usará estes dados para realizar a carga dentro do DW. A regra que será construída está ligada diretamente a regra de negócio do sistema que sofrerá a extração, por isso se torna imprescindível o conhecimento do sistema legado para que seja efetuada com sucesso a extração dos dados.

O usuário deverá selecionar sempre o campo de nível mais alto em relação à regra de negócio, ou seja, deverá optar sempre por um campo que pertença à entidade pai para a aplicação da regra, cuja suas informações constem em entidades filhas, que tenham uma chave estrangeira apontando para ela. Este entendimento se faz necessário para o cadastro das regras, pois sem ele será difícil de manipular os dados da estrutura homogênea na qual os dados extraídos ficarão armazenados.

As regras de extração são compostas por três campos, uma descrição do que faz esta regra, as regras de tempo e as regra de categoria.

3.3.2.8.1 Formulando o tempo das regras de extração

Ao cadastrar uma regra, o usuário já deve ter em mente em qual campo do sistema legado esta regra irá operar. A regra de tempo cadastrada pelo usuário deve obedecer ao formato descrito na figura 19.

NOME DA TABELA:CAMPO LIGAÇÃO:NOME CAMPO:DATA INICIAL:DATA FINAL;

Figura 19 – Formato da regra de tempo

A seguir, tem-se a legenda dos campos que compõem a regra de extração de tempo:

a) nome da tabela: deve ser informado o nome de uma entidade filha que possui uma chave estrangeira para o entidade do campo que será aplicada a regra. Esta informação poderá ser omitida no caso do dado estar na mesma entidade do campo que a regra será aplicada;
b) campo de ligação: deve ser informado o campo que é chave estrangeira da entidade pai, para que seja possível fazer a ligação entre os dados e recuperar a informação correta. Esta informação poderá ser omitida no caso do dado estar na mesma entidade do campo que a regra será aplicada;
c) nome do campo: o nome do campo cujo dados serão extraídos;
d) data inicial: filtro especificando o período inicial que deve compreender a extração;
e) data final: filtro especificando o período final que deve compreender a extração.

Neste nível de definição das regras, intuitivamente o usuário já está preparando e dimensão tempo do DW, a dimensão mais importante e que deve estar presente em qualquer estrutura de DW. A seguir, são apresentadas as duas formas que podem assumir as regras de tempo:

a) ::nome_campo:Data_Inicial:Data_final: esta regra se aplica quando as informações referentes ao tempo de acontecimento do dados está na mesma entidade do campo no qual será aplicada a regra. Exemplo: “::Ano:2006:2008;”;
b) nome_tabela:campo_ligação:nome_campo:Data_Inicial:Data_final: esta regra se aplica quando as informações referentes ao tempo de acontecimento do dados não está na mesma entidade do campo no qual será aplicada a regra, sendo necessário informar em qual tabela esta a informação e qual o campo que une as duas tabelas. Exemplo: “Vendas:Id_Cliente:Ano:2006:2008;”.

Mais de uma regra de tempo poderá ser aplicada na mesma regra de extração, sendo que as regras serão separadas pelo símbolo ponto e vírgula (;). Seque o exemplo de várias regras de tempo associadas a mesma regra de extração de dados: “regra_tempo_1;regra_tempo_2;regra_tempo_3;”.

A execução de uma regra de extração onde foi configurada a regra de tempo irá resultar em uma estrutura homogênea, conforme figura 20.

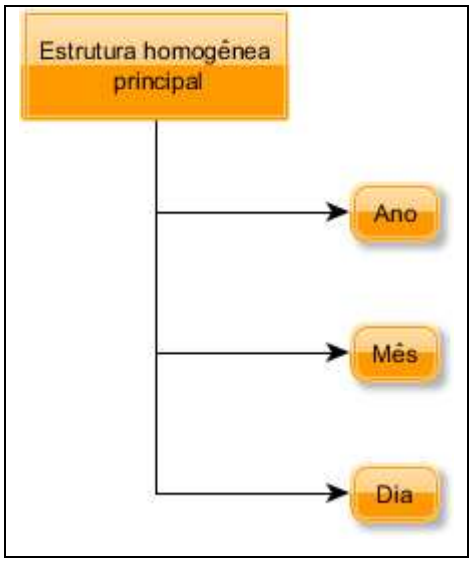


Figura 20 – Estrutura homogênea resultante da aplicação da regra

3.3.2.8.2 Formulando as categorias da regra de extração

As regras de categorias aplicadas à regra de extração seguem o mesmo conceito básico apresentado nas regras de tempo do tópico descrito acima. No entanto o seu objetivo é completamente diferente. Ao invés de se preocupar com dados relacionados ao período de acontecimento do que os dados representam, estas regras visam buscar informações que podem categorizar os dados, ou até mesmo informações que podem vir a serem medidas de uma tabela de fato. A figura 201ilustra o formado que deve obedecer as regras para extração das categorias.

NOME TABELA:CAMPO LIGAÇÃO:NOME CAMPO:OPERADOR:VALOR;

Figura 21 – Formato da regra de categoria

Conforme a figura 21, as regras de extração de dados necessitam seguir uma série de padrões, sendo eles:

a) nome da tabela: deve ser informado o nome de uma entidade filha que possui uma chave estrangeira para o entidade do campo que será aplicada a regra. Esta informação poderá ser omitida no caso do dado estar na mesma entidade do campo que a regra será aplicada;
b) campo de ligação: deve ser informado o campo que é chave estrangeira da entidade pai, para que seja possível fazer a ligação entre os dados e recuperar a informação correta. Esta informação poderá ser omitida no caso do dado estar na mesma entidade do campo que a regra será aplicada;
c) nome do campo: o nome do campo cujo dados serão extraídos;
d) operador: poderá ser qualquer operador conhecido em SQL, como “=”, “<>”, “like”;
e) valor: como sugere o nome, será o valor no qual os dados serão filtrados com base no operador selecionado.

A seguir, serão apresentadas as duas formas que podem assumir as regras de categoria:

a) ::nome_campo:operador:valor: esta regra se aplica quando as informações referentes ao tempo de acontecimento do dados está na mesma entidade do campo no qual será aplicada a regra. Exemplo: “::ValorVenda:>=:500;”;
b) nome_tabela:campo_ligação:nome_campo:operador:valor: esta regra se aplica quando as informações referentes ao tempo de acontecimento do dados não está na mesma entidade do campo no qual será aplicada a regra, sendo necessário informar

em qual tabela esta a informação e qual o campo que une as duas tabelas. Exemplo: “Vendas:Id_Cliente:Vlr_Multa:>:0;”.

Mais de uma regra de categoria poderá ser aplicada na mesma regra de extração, sendo que as regras serão separadas pelo símbolo ponto e vírgula (;). Exemplo: “regra_categoria_1;regra_categoria_2;regra_categoria_3;”.

A execução de uma regra de extração onde foi configurada a regra de categoria irá resultar na seguinte estrutura homogênea, representada pela figura 22, estrutura esta que será apresenta em um dos próximos tópicos deste capítulo.

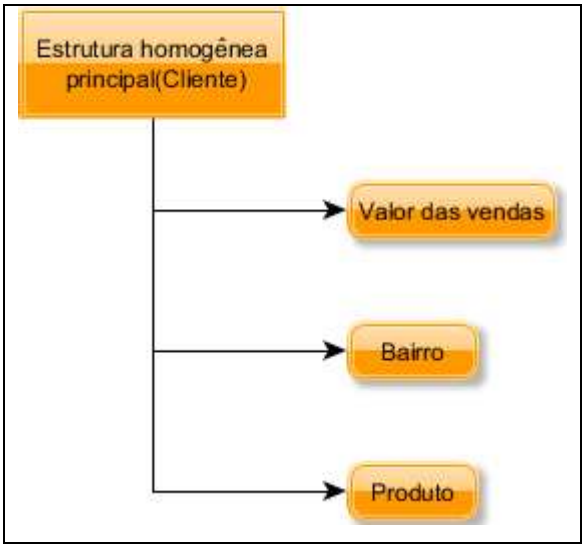


Figura 22 – Estrutura homogênea resultante da aplicação da regra.

Conforme descrito anteriormente, existe uma limitação quanto à extração de dados originárias de uma entidade filha. Por exemplo, há a necessidade de extrair dados referentes às vendas realizadas de um certo grupo isolado de clientes. Será necessário extrair dados dos valores das vendas de cada cliente e a cidade em que este cliente está.

Na grande maioria dos bancos de dados relacionais, as entidades estão normalizadas ao ponto de terem uma entidade destinada somente aos bairros. E como visto anteriormente, as regras de extração hoje estão limitadas a extrair dados somente de entidades que possuem uma chave estrangeira apontando para a entidade “pai”. A seguir tem-se a figura 23 representando o modelo de dados citado conforme o estudo de caso acima.

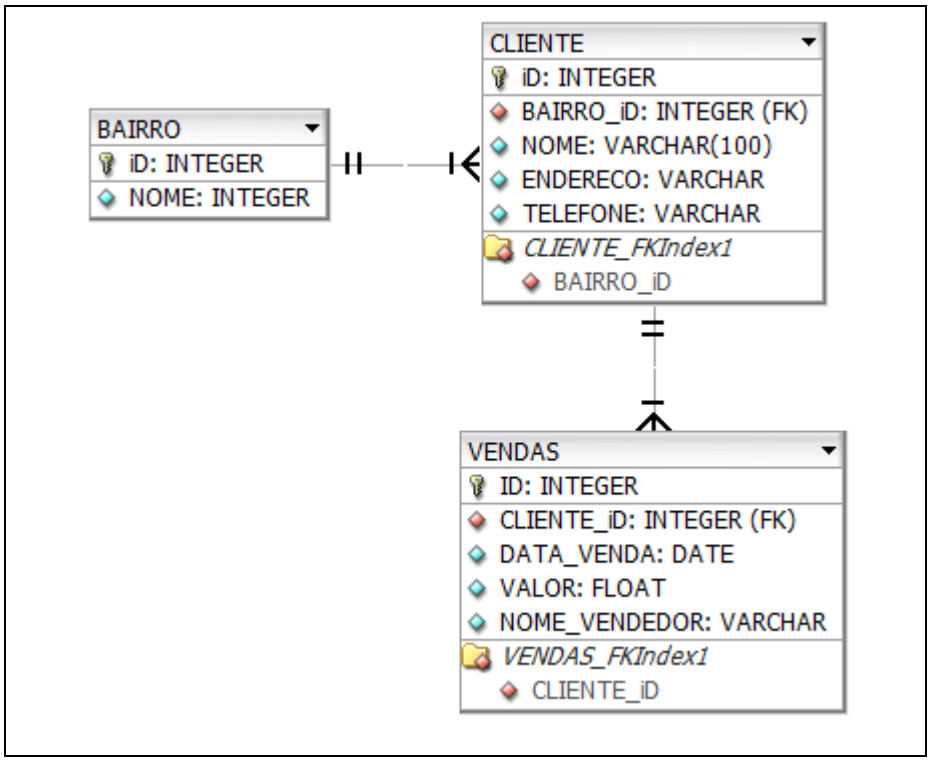


Figura 23 – Entidades normalizadas de um sistema legado

Para resolver estas situações no Freedom, torna-se necessária a extração isolada da entidade Bairro, armazenando valores como o nome do bairro e a chave primária do bairro. A chave primária é necessária para unir as informações posteriormente no processo de transformação dos dados, onde as informações serão cruzadas dentro da estrutura homogênea através do campo que no sistema legado representou a sua chave primária.

3.3.2.9 Mapeamento da extração dos de dados

Tendo em mãos as estruturas necessárias para extração de dados e definida a forma com que elas serão extraídas através das regras de extração, chega ao momento de mapear o processo. Para isso, o sistema Freedom proporciona duas *interfaces* distintas. Partindo da *interface* mais simples, o cadastro de mapeamentos, onde o usuário atribuirá a carga de dado os mapeamentos de extração. Para acessar o cadastro de mapeamentos, clicar no *menu* “Iniciar”, “Cadastros”, “Carga”, e clicar no item de *menu* “Mapeamento”. A mesma listagem conhecida das telas anteriores será apresentada, contendo todas as funcionalidades já descritas.

Ao inserir um novo mapeamento, o formulário de cadastro apresentado pelo sistema solicitará a qual carga será atribuído este mapeamento. Na sequência será necessário informar

qual campo pertencendo ao sistema legado será usado como base para aplicação das regras de extração. A seleção deste campo é uma das etapas críticas do processo de carga de dados, por isso é extremamente importante que o usuário tenha ciência do funcionamento das regras descritas nos tópicos anteriores. Por fim, o usuário poderá escrever a descrição do significado deste mapeamento, para fins de consistência do metadado do DW.

A outra *interface* que fornece a funcionalidade para cadastrar os mapeamentos do banco de dados pode ser acessada pelo *menu* da tela principal da aplicação “Extração de dados”. Esta tela além de proporcionar o mapeamento dos dados a serem extraído, executa o processo de extração de dados, no caso do usuário julgar necessário realizar a extração separadamente das etapas de transformação e carga. Esta tela também pode ser usada para testar as regras de extração de dados, proporcionando ao analista verificar a viabilidade da aplicação das regras que foram criadas. Na figura 24 é apresentado o primeiro passo da tela. Trata-se de um passo a passo para execução do processo de extração de dados.

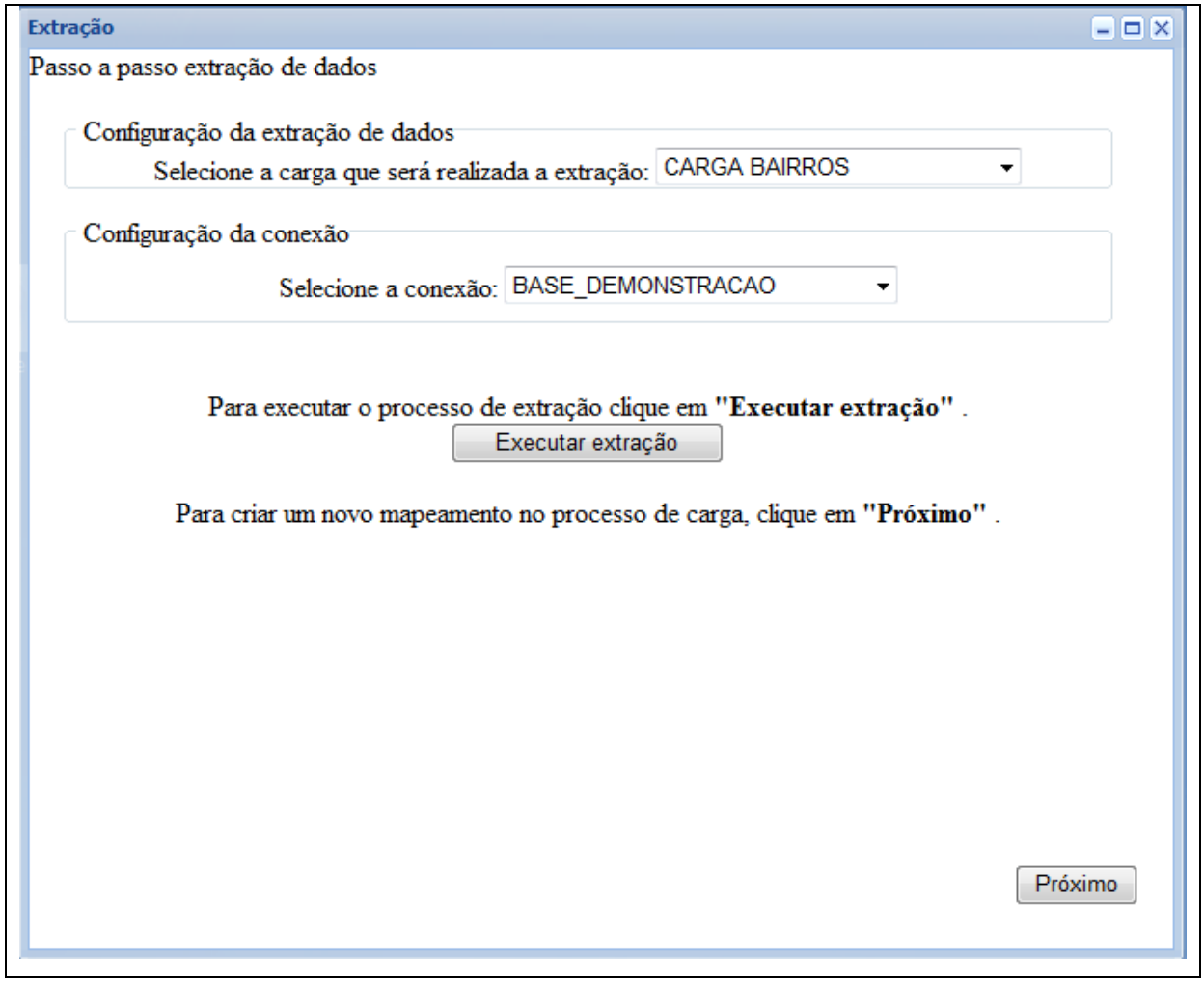


Figura 24 – Primeiro passo da tela de extração de dados

Esta etapa possui a seleção da carga cuja extração será realizada. Em seguida qual a conexão que será usada para buscar as tabelas que irão compor o mapeamento. Nada impede que o usuário retorne a este passo para inserir tabelas de uma conexão diferente da primeira selecionada.

Outros dois botões fazem parte desta tela. O primeiro botão se destina a processos de carga que já foram configurados, e o usuário deseja somente executar o processo de extração. Logo abaixo existe o botão próximo, que irá avançar para o segundo passo o processo de configuração da carga de dados. Este passo a passo deve ser executado caso o processo de carga ainda não esteja configurado ou seja necessária alguma alteração em sua estrutura.

No segundo passo será configurado o mapeamento do processo de carga de dados. O usuário irá selecionar as tabelas que serão necessárias selecionado o seu nome na caixa de seleção no canto superior esquerdo da tela, clicando em adicionar para mover tabela para as entidades selecionadas.

Ao inserir uma tabela, o usuário irá configurar os campos desta tabela, clicando no nome da tabela na caixa de seleção à esquerda. Os campos desta tabela são apresentados, conforme a figura 25.

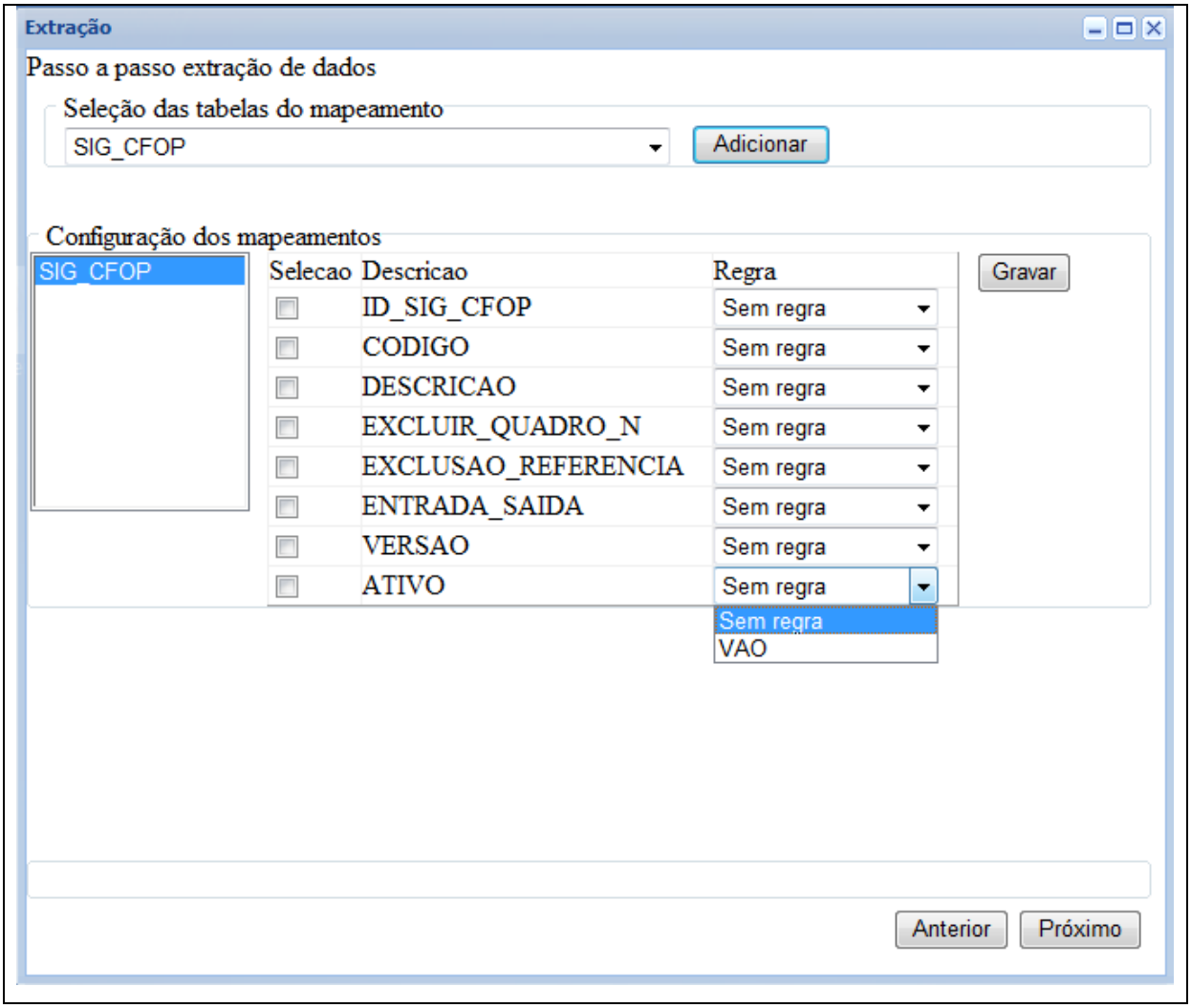


Figura 25 – Segundo passo da tela de extração de dados

No primeiro item de configuração dos campos da tabela há uma caixa de seleção. Ao clicar neste campo, ele será selecionado para fazer parte do processo de extração de dados do sistema legado. O segundo item corresponde ao nome do campo. E por fim, no terceiro item está a regra de extração a ser aplicada durante a fase de extração dos dados do sistema legado. Intuitivamente o usuário acaba de realizar o processo de mapeamento da extração de dados. Esta tela proporciona a criação múltipla de mapeamentos, diferente da tela de cadastro de mapeamentos onde é possível fazer somente um mapeamento por transação.

Ao clicar no botão próximo, o sistema será redirecionado ao terceiro passo do processo de configuração da carga de dados. Nesta etapa o usuário poderá visualizar todos os mapeamentos que foram inseridos por ele até o momento, e havendo a necessidade de retornar para ajustar alguma configuração, deverá clicar no botão “Anterior”.

Seguindo para a última etapa do processo de extração, o usuário será informado de todos os processos de mapeamento configurado na carga selecionada no primeiro passo desta funcionalidade. Ele poderá realizar a limpeza das estruturas homogêneas previamente

armazenadas por extrações anteriores desta carga. Ao clicar em concluir, o sistema irá realizar o processo de extração de dados, que será o tema do próximo tópico deste capítulo.

3.3.2.10 Algoritmo de extração de dados

Como visto nos tópicos anteriores, o processo de extração está estruturado sobre dados do sistema legado, regras de extração e o mapeamento dos mesmos. A seguir será apresentada a listagem do algoritmo de extração de dados, que foi implementado dentro da classe “CargaBLL”. Esta classe contém todos os algoritmos que serão usados para realização do processo de carga de dados. Pertence a camada BLL, sendo o núcleo da aplicação Freedom. A seguir tem-se o quadro 6, contendo a estrutura principal do código fonte do processo de extração.

```
public string ExtracaoDados(CargaCDL carga)
        {
            TimeSpan dtIni = DateTime.Now.TimeOfDay;
            TimeSpan tempoGasto = new TimeSpan();
            RefreshCargaDados(carga);
            string retorno = string.Empty;
            foreach (MapeamentoCDL map in carga.Mapeamentos)
            {
                string nomeCampo = map.Campo.Descricao;
                string from = string.Empty;
                string where = string.Empty;
                string mapDesc = "null";
                if (!string.IsNullOrEmpty(map.Descricao))
                    mapDesc = map.Descricao;

                string prefixo = map.Prefixo;
                if (string.IsNullOrEmpty(prefixo))
                    prefixo = string.Empty;
                String query = "SELECT " + map.NomeTabela + "." +
                       map.Campo.CampoPK + " as IdLegado, " +
                                        map.Id + " as Mapeamento, " +
                                        "'" + map.Descricao + "'" + "
                                            as Identificacao, " +
                                        "'" + nomeCampo + "'" + " as
                                        Nome, " +
                                        map.Campo.Descricao + " as
                                        valor, " +(Convert.ToInt32(
                                    map.Campo.Tipo)).
                                    ToString() + " as TipoCampo " +
                                     "FROM " + map.NomeTabela;

                if (map.RegraExtracao != null)
                    AplicaRegraExtracao(map, out from, out where);
                query += from;
                query += where;

                ControleConexao.ChangeConnection(
```

```
                        map.Campo.Tabela.BaseLegado.StringConexao);

                IQuery teste = this.CurrentSession.CreateSQLQuery(query)
                    .AddScalar("IdLegado", NHibernateUtil.Int32)
                    .AddScalar("Mapeamento",
                       NHibernateUtil.Entity(typeof(MapeamentoCDL)))
                    .AddScalar("Identificacao", NHibernateUtil.String)
                    .AddScalar("Nome", NHibernateUtil.String)
                    .AddScalar("Valor", NHibernateUtil.String)
                    .AddScalar("TipoCampo", NHibernateUtil.Int32)
                        .SetResultTransformer(
                              Transformers.AliasToBean(typeof(HomogCDL)));


                IList<HomogCDL> ret = teste.List<HomogCDL>();
                ControleConexao.ReturnFreedomConnection();


                retorno = GravaEstruturaHomog(ret);
            }
            tempoGasto = dtIni.Duration();
            retorno += " Tempo gasto: " + tempoGasto.ToString() + ".";
            return retorno;
        }
```

Quadro 6 – Código fonte do processo de extração de dados

Nos primeiros trechos do código são instanciadas as variáveis locais do método de extração, seguidas de uma estrutura de repetição que irá percorrer todos os mapeamentos contidos na carga de dados passada como parâmetro para o método. Na seqüência são instanciadas variáveis locais à estrutura de repetição, que serão usadas na construção do processo de extração. A parte mais importante deste trecho de código vem a seguir, que é a montagem dinâmica do *script* na linguagem SQL que dará origem às estruturas homogêneas extraídas do sistema legado mapeado.

Nesta etapa que entram em ação das regras de extração de dados que foram cadastradas pelo usuário. Elas servirão de apoio para a montagem do *script* SQL de construção das estruturas homogêneas. Neste ponto do código também entrará em ação as funcionalidades do NHibernate, que transformará o resultado da pesquisa dos *scripts* em objetos das classes que compões a estrutura homogênea proposta pelo Freedom. Após a criação das instâncias em memória das estruturas homogêneas, elas serão armazenadas fisicamente no banco de dados do sistema Freedom, para que na próxima etapa, a de transformação, elas sejam usadas.

No quadro 7 será apresentado o código fonte da interpretação das regras de extração.

```
private void AplicaRegraExtracao(MapeamentoCDL map, out string from, out
string where)
        {
            String clausulaFrom = string.Empty;
            String clausulaWhere = " WHERE 0 = 0 ";
            RegraExtracaoCDL regra = map.RegraExtracao;
```

```
            this.CurrentSession.Refresh(regra);
            if (regra != null)
            {
                if (regra.regras_extracao_categoria != null)
                {
                    foreach (string[] l in
                        regra.regras_extracao_categoria)
                    {
                        if (!string.IsNullOrEmpty(l[0]))
                            if (!clausulaFrom.Contains(l[0])) //verifica
                                    se a tabela já não foi declarada
                                clausulaFrom += ", " + l[0];
                        if (!string.IsNullOrEmpty(l[1]))
                        {
                            string aux = " and " + map.NomeTabela + "." +
                                    map.Campo.CampoPK +
                                    " = " + l[0] + "." + l[1];
                            if (!clausulaWhere.Contains(aux)) //verifica
                              se o join já foi feito.
                                clausulaWhere += aux;
                        }
                        if (!string.IsNullOrEmpty(l[2]))
                            if (!string.IsNullOrEmpty(l[3]))
                                if (!String.IsNullOrEmpty(l[0]))//Caso
                      l[0] for nulo, entao o campo pertence a propria tabela
                                {
                                    clausulaWhere += " and " + l[0] + "."
                                         + l[2] + " " + l[3] + " " + l[4];
                                }
                                else
                                    clausulaWhere += " and " +
                                         map.NomeTabela + "." + l[2] +
                                         " " + l[3] + " " + l[4];


                    }
                }

                if (regra.regras_extracao_tempo != null)
                {
                    foreach (string[] l in regra.regras_extracao_tempo)
                    {
                        if (!string.IsNullOrEmpty(l[0]))
                            if (!clausulaFrom.Contains(l[0])) //verifica
                                    se a tabela já não foi declarada
                                clausulaFrom += ", " + l[0];
                        if (!string.IsNullOrEmpty(l[1]))
                        {
                            string aux = " and " + map.NomeTabela + "." +
                              map.Campo.CampoPK + " = " +
                              l[0] + "." + l[1];
                            if (!clausulaWhere.Contains(aux)) //verifica
                              se o join já foi feito.
                                clausulaWhere += aux;
                        }
                        if (!string.IsNullOrEmpty(l[2]))
                            if (!string.IsNullOrEmpty(l[3]))
                                if (!string.IsNullOrEmpty(l[4]))
                                    if (!String.IsNullOrEmpty(l[0]))//Caso
                      l[0] for nulo, entao o campo pertence a propria tabela
                                    {
```

```
                                        clausulaWhere += " and " + l[0] +
                                          "." + l[2] + " between " +
                                          l[3] + " and " + l[4];
                                    }
                                    else
                                        clausulaWhere += " and " +
                                          map.NomeTabela + "." + l[2] +
                                    " between " + l[3] + " and " + l[4];

                        }
                    }
                }
                from = clausulaFrom;
                where = clausulaWhere;
            }
```

Quadro 7 – Código fonte de interpretação das regras

A seguir têm-se o quadro 8, que contém o código fonte do armazenamento dos dados migrados para a estrutura homogênea.

```
private string GravaEstruturaHomog(IList<HomogCDL> lista)
{
            ITransaction transacao =
            this.CurrentSession.BeginTransaction();
            transacao.Begin();
            try
            {
                HomogBLL homogBLL = new HomogBLL();
                int contador = 0;
                foreach (HomogCDL homog in lista)
                {
                    contador++;
                    homogBLL.Insert(homog);
                    GravaSubCategoriasHomog(homog);
                }
                transacao.Commit();

                return "Extração realizada com sucesso! ";
            }
            catch (Exception ex)
            {
                transacao.Rollback();
                return "Erro no processo de extração: " + ex.Message;
            }
              }
```

Quadro 8 – Código fonte da extração de dados reais do sistema legado

3.3.2.11 Estrutura homogênea de dados

Para o armazenamento de dados de diversos sistemas em um ambiente único e integrado, foi criada uma estrutura homogênea na qual todos os dados resultantes do processo de extração descrito no tópico anterior serão guardados. Esta estrutura é composta por três

níveis:

a) o nível principal: contém os dados do níveis mais altos da hierarquia da regra de negócios. Possui um campo chamado nome, destinado ao nome do campo do sistema legado do qual foi extraído e um campo valor, que armazena o valor deste campo no banco de dados que sofreu a extração;
b) o nível do tempo: destinado a armazenar informações referentes ao período de tempo em relacionado diretamente ao dado extraído. As informações contidas neste nível da estrutura serão usadas para popular a dimensão tempo;
c) o nível de categoria: contém as demais informações que estão relacionadas ao nível principal, atribuindo características a ele. Neste nível são armazenadas as informações que serão convertidas em medidas e dimensões do DW.

A seguir na figura 26, tem-se a estrutura de entidades que compões a estrutura homogênea de armazenamento de dados.

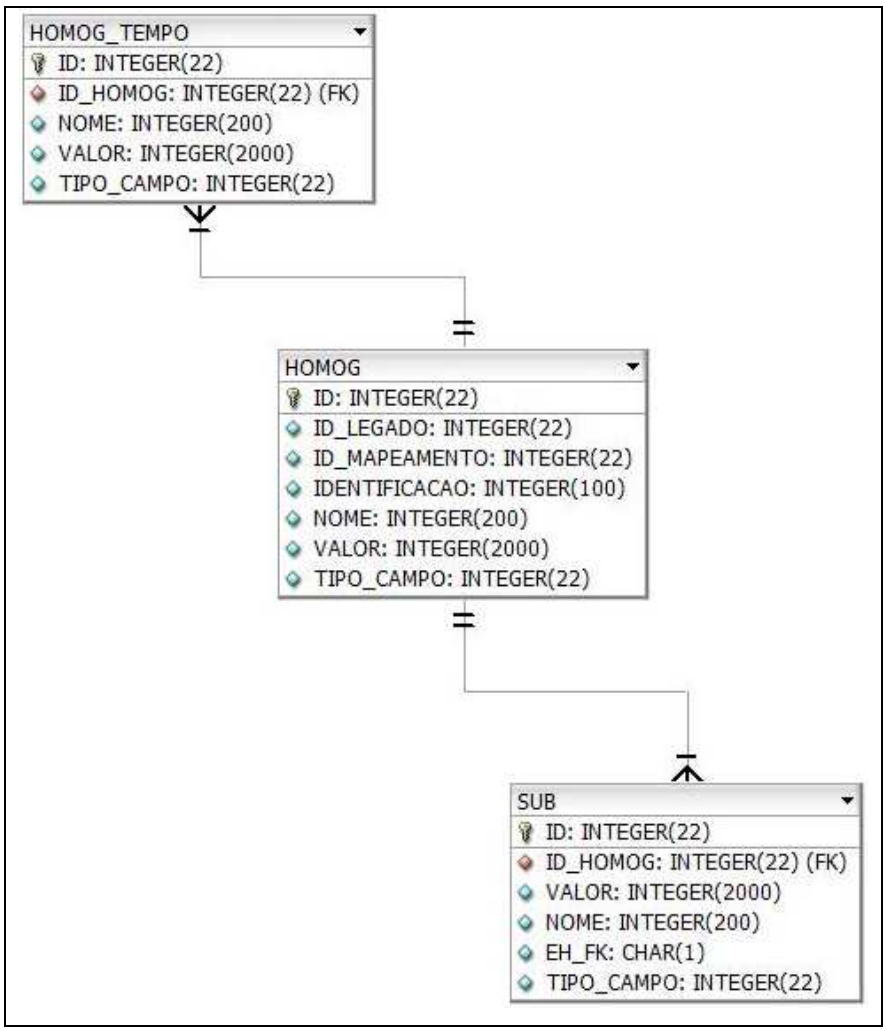


Figura 26 – Estrutura homogênea de armazenamento de dados

3.3.2.12 Processo de transformação

Este processo exige que o usuário tenha conhecimento de SQL. Como esta ferramenta está voltada para uso de profissionais que atuam em desenvolvimento de sistema e banco de dados, este processo será de fácil entendimento. É através da transformação dos dados armazenados nas estruturas temporárias que surgirão os registros a serem inseridos no DW.

Para realizar o processo de transformação de dados no Freedom, foram adotados alguns conceitos sugeridos em Inmon (1999), que propõe a definição de um modelo lógico de transformação dos dados. Este modelo contém as definições de todo o processo de transformação dos dados, sendo composto pelas informações da origem dos dados, as transformações que deverão ocorrer nestes dados e o destino final do dado transformado. Nos tópicos seguintes serão apresentados os modelos lógicos construídos no Freedom para realizar o processo de extração dos dados.

Para dar início ao processo de transformação na aplicação, é necessário cadastrar formalmente a existência deste processo. O usuário deverá selecionar o item “Transformação” na página principal da aplicação. O sistema irá exibir a listagem das transformações previamente cadastradas. Ao inserir um novo processo de transformação, o usuário deverá informar a qual processo de carga esta transformação atuará e em qual DW os dados serão armazenados. No campo descrição, será atribuído um nome a este processo de transformação e no campo “comentário” poderá ser descrito informações relevantes ao metadados do DW.

A seguir tem-se a figura 27, apresentando a tela de cadastro das regras de transformação.

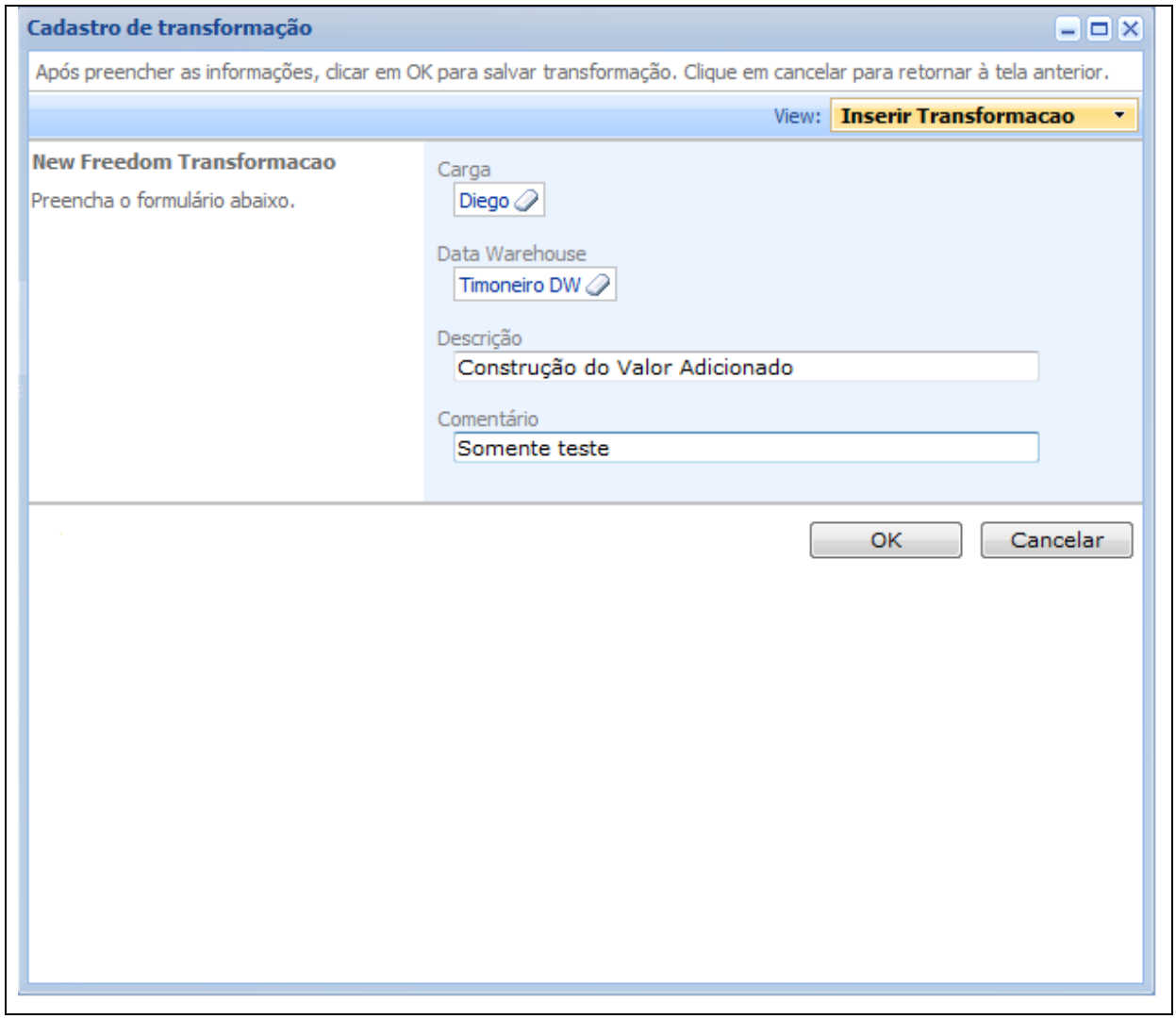


Figura 27 – Tela de cadastro de transformações

3.3.2.12.1 Modelo Lógico

O modelo lógico usado para realizar transformações de dados no sistema Freedom é implementado conforme o modelo de classes apresentado na figura 28.

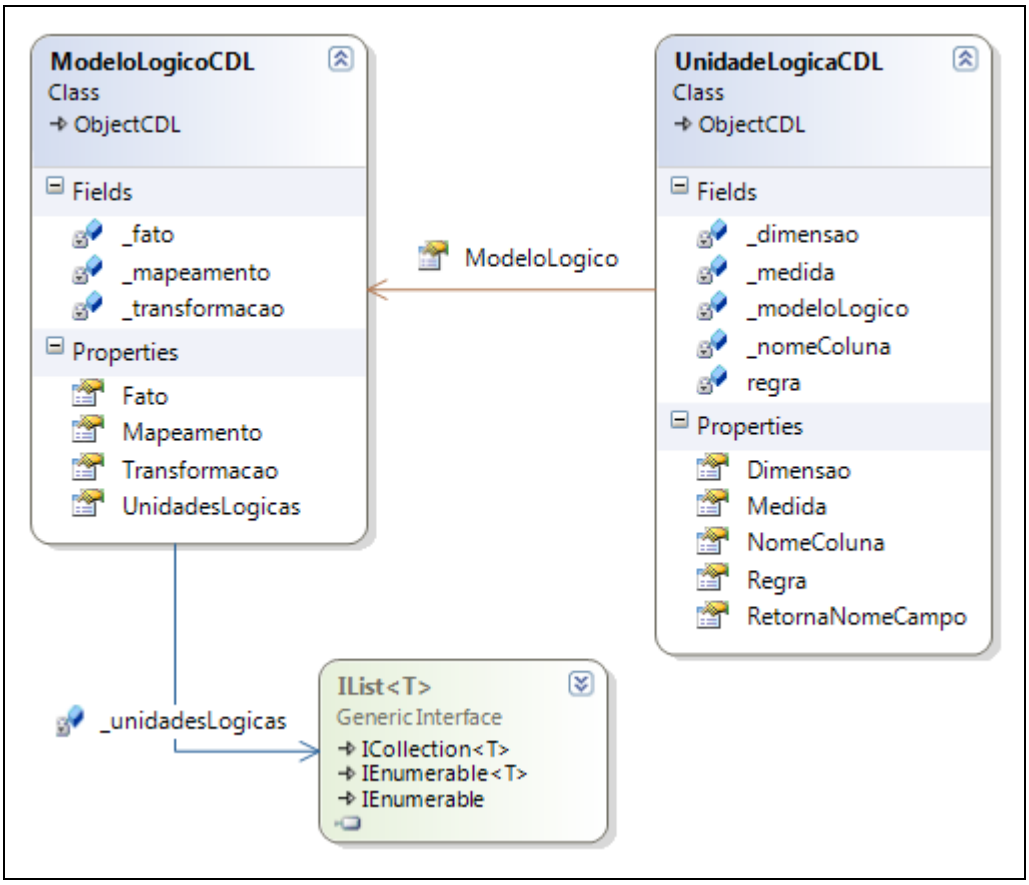


Figura 28 – Modelo de classes do modelo lógico de transformação

Na classe “ModeloLogicoCDL” estão as informações de origem, transformação e destino dos dados a serem transformados, contendo uma lista de unidades lógicas, representadas pela classe “UnidadeLogicaCDL”, que contém as definições de como o modelo lógico irá se comportar dentro da aplicação. São através das unidades lógicas que o processo de transformação dos dados será executado. Nesta classe estão às informações de quais regras de transformação serão usadas e onde os dados transformados serão salvos no DW, ou seja, onde a carga será efetivada dentro do DW, sendo tabelas de fato ou dimensão.

O cadastro dos modelos lógicos no Freedom será realizado através de cadastro na tela “Modelo Lógico”, podendo ser acessado através do seu ícone presente na tela principal do sistema, ou no *menu* “Iniciar”, “Cadastros”, “Carga”, selecionando o item de *menu* “Modelo Lógico”. Para inserir um novo modelo lógico o usuário deverá informar no primeiro campo a qual processo de transformação pertence o modelo, a qual tabela de fato se destina a carga e qual foi o mapeamento usado para extração de dados originou o dado a ser transformado. Esta última informação é opcional, relevante somente a consistência do metadado do DW. A seguir tem-se a figura 29, apresentando o formulário de cadastro do modelo lógico.

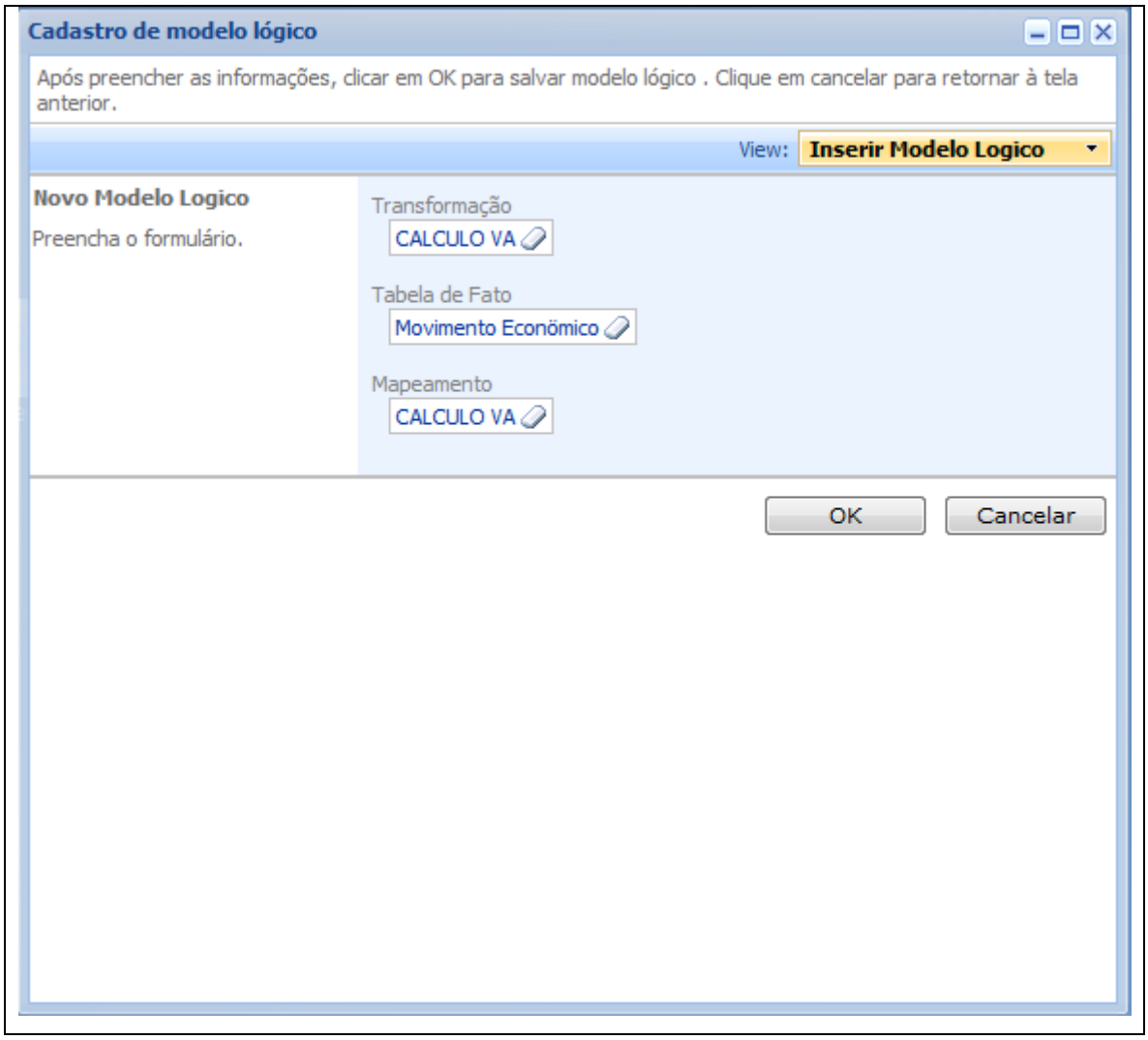


Figura 29 – Cadastro do modelo lógico

As unidades lógicas serão cadastradas da mesma forma que a unidade lógica, podendo ser acessadas diretamente na página inicial da aplicação através do ícone “Unidade Lógica”, ou através do item de *menu* presente no *menu* “Iniciar”, “Cadastros”, “Carga” e Unidade Lógica. Ao inserir uma nova unidade lógica, o usuário deverá informar a qual modelo lógico esta unidade pertence. Em seguida qual a regra de transformação será usada para transformar o dado previamente armazenado na estrutura homogênea cuja origem pertence ao processo de extração. Na seqüência o usuário deverá optar por inserir o dado em uma medida da tabela de fato, ou em uma dimensão. Caso seja um dado numérico, que representa um valor, o mesmo deverá ser selecionado em qual medida este dado será inserido. No caso deste dado representar uma informação que caracterize um fato (atributo), o usuário deverá informar qual dimensão será alimentada pela carga.

Ainda no caso da seleção da dimensão, o usuário deverá informar qual campo da tabela

dimensão pertence o dado transformado, preenchendo o campo “Nome Coluna” com o nome da coluna da tabela dimensão. A seguir tem-se a figura 30, que apresenta o formulário de cadastro da unidade lógica.

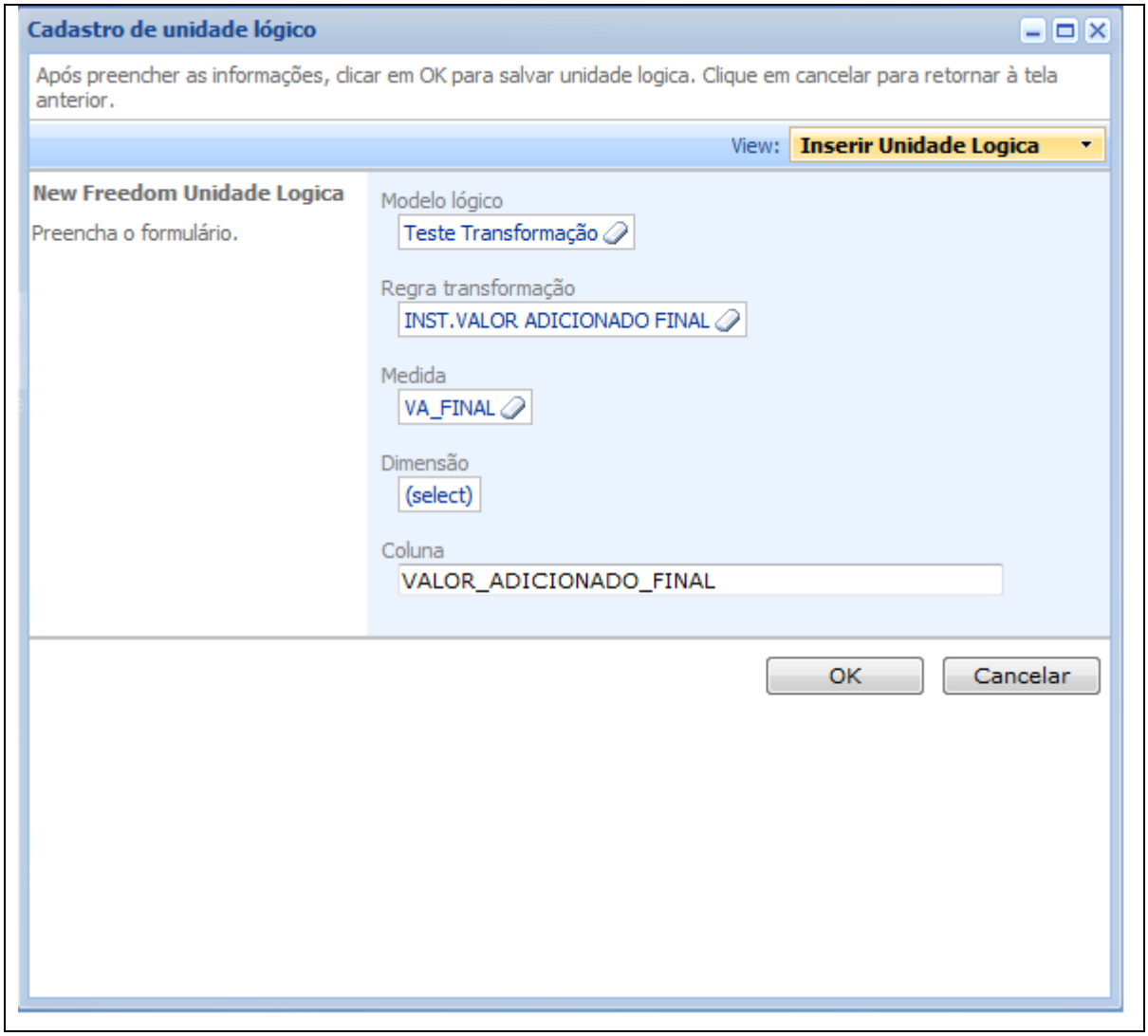


Figura 30 – Cadastro da unidade lógica

3.3.2.12.2 Regras de conversão e integração dos dados

As regras de conversão de dados estão associadas ao processo de transformação. Estas regras serão construídas com o objetivo de integrar os dados, quanto ao seu tipo e sua formatação. Este processo se torna imprescindível devido ao fato de que os dados são extraídos de vários sistemas e ambientes diferentes. Dados de igual significado podem assumir formatações diferentes conforme a forma com que cada sistema foi concebido ou que a própria regra de negócio exige. Este processo deve estar presente em qualquer processo de

transformação de cuja finalidade seja a carga em um DW, conforme Inmon (1999) e Brackett (1996).

A conversão de dados serão cadastradas no Freedom através da definição de regras de conversão. Para cadastrar uma regra de conversão no Freedom o usuário deverá acessar a tela “Conversores de dados”, que pode ser encontrada acessando o *menu* “Iniciar”, “Cadastros”, “Regras”. Na figura 31 é ilustrada a tela de cadastro dos conversores de dados.

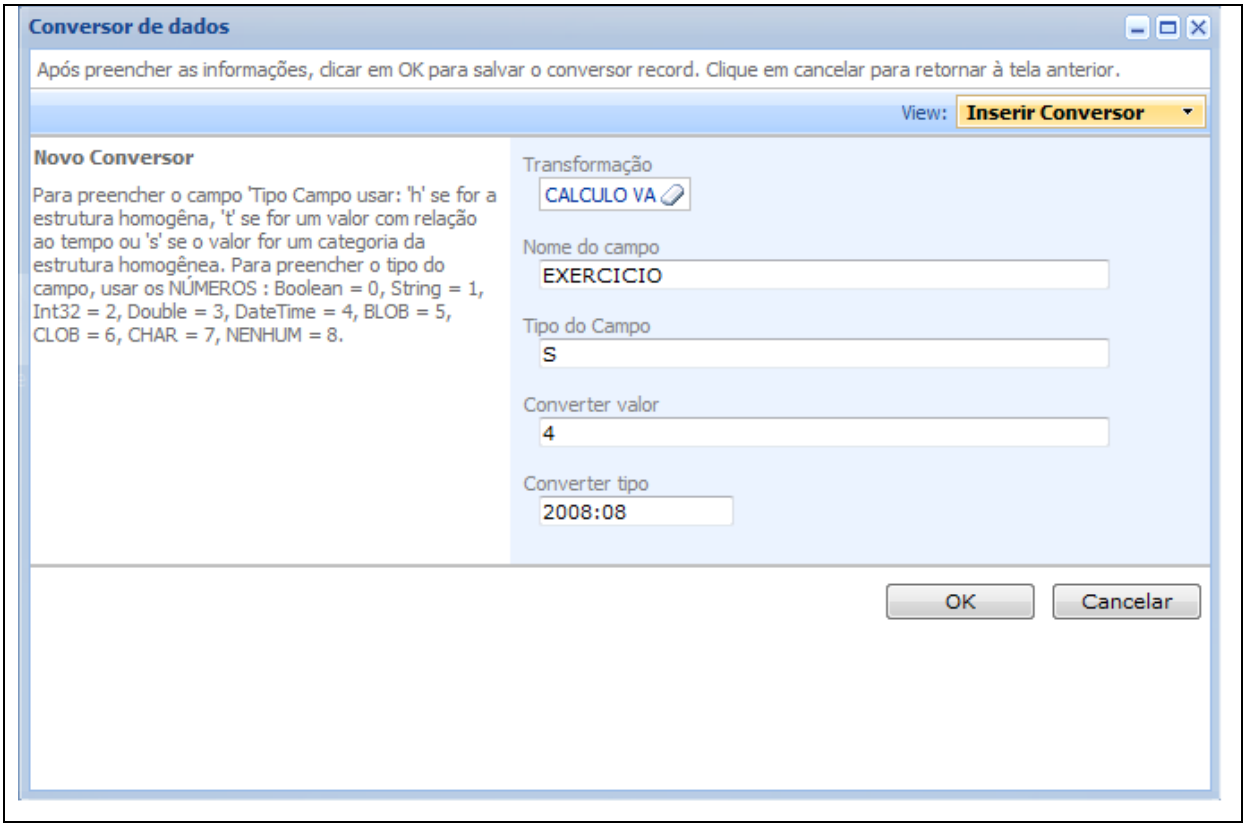


Figura 31 – Tela de cadastros dos conversores de dados

Ao inserir uma nova regra de conversão, o usuário deverá informar a qual processo de transformação pertencerá esta regra. Em seguida qual o campo da estrutura homogênea sofrerá o processo conversão. No campo “Nível estrutura homogênea”, o usuário irá informar a qual nível da estrutura pertence o campo, informando. Inserir “h” para a estrutura principal, “s“ para estrutura de categorias e “t” para estrutura do tempo.

Na seqüência o usuário irá informar a regra para conversão do dado. Esta regra deve assumir o seguinte padrão: valor do campo a ser formatado, o caráter dois pontos (:), novo valor assumido pelo dado. Exemplo: 2008:08. Neste exemplo, foi convertida a formatação da data de quatro dígitos para dois dígitos. Na seqüência, o tipo de dado que o campo selecionado será convertido, os tipos dos campos serão informados de forma numérica, conforme a numeração: 0 para dados booleanos, 1 para o tipo String, 2 para inteiros, 3 para dados com pontos flutuantes (*float*), 4 para datas, 5 para BLOB, 6 para campos CLOB, 7 para

caracteres (CHAR) e 8 caso não haja conversão. Estes dados servirão como base para construção dinâmica dos *scripts* que irão alterar os dados armazenados na estrutura homogênea. A seguir tem-se o código fonte que irá gerar o *script* de integração dos dados homogêneos. O quadro 9 contém o código fonte responsável por montar o *script* de conversão de dados.

```
public void ConverteDados(TransformacaoCDL trans)
{
            CurrentSession.Refresh(trans);
            OracleConnection conexao = new OracleConnection("Data
                  Source=xe; User Id=FREEDOM; Password=katana;");
            if (conexao.State == System.Data.ConnectionState.Closed)
                conexao.Open();

            foreach (ConvercorCDL con in trans.Convercores)
            {
                if (con.ConverteTipo != 8) //Sem converção
                {
                    String query = String.Empty;
                    if (con.TipoCampo == 'h')
                    {
                        query = @"UPDATE HOMOG H SET TIPO_CAMPO = " +
                                        con.ConverteTipo.ToString() +
                                    " where H.NOME = " + "'" +
                                          con.NomeCampo + "'";
                    }

                    if (con.TipoCampo == 's')
                    {
                        query = @"UPDATE SUB S SET TIPO_CAMPO = " +
                                    con.ConverteTipo.ToString() +
                                         " where S.NOME = " + "'" +
                                          con.NomeCampo + "'";
                    }

                    if (con.TipoCampo == 't')
                    {
                        query = @"UPDATE HOMOG T SET TIPO_CAMPO = " +
                                        con.ConverteTipo.ToString() +
                                        " where T.NOME = " + "'" +
                                        con.NomeCampo + "'";
                    }

                    OracleCommand cmd = new OracleCommand(query, conexao);
                    cmd.ExecuteNonQuery();
                }
                if (!String.IsNullOrEmpty(con.ConverteValor))
                    ConverteValor(con, conexao);
            }
            conexao.Close();
        }
```

Quadro 9 – Código fonte da montagem do *script* de conversão de dados

3.3.2.12.3 Cadastro das regras de transformação

As regras de transformação desempenham o papel mais importante de todo o processo de transformação dos dados. Fazem parte da unidade lógica de definição da transformação. As regras de transformação serão construídas usando linguagem SQL, que proporciona a construção de qualquer operação sobre os dados. Deixar o profissional de DW manipular as regras de extração usando SQL proporciona a uma infinidade de possibilidades de manipulação dos dados a serem transformados.

A base para criação das regras de extração está no nível principal da estrutura homogênea, que representa o nível mais alto em relação à regra de negócio. Neste é possível analisar a extrema importância do entendimento da estrutura e objetivo das tabelas de fato de um DW. Sem o conhecimento do real significado dos fatos, é impossível que o processo de carga como um todo seja bem sucedido, devido a complexidade de integração entre os dados e o seu significado. A base para a criação das regras de transformação está representada no quadro 10.

```
SELECT (REGRA_TRANSFORMACAO_1) AS CAMPO_TABELA_FATO_1
       (REGRA_TRANSFORMACAO_2) AS CAMPO_TABELA_FATO_2
       (REGRA_TRANSFORMACAO_3) AS CAMPO_TABELA_FATO_3
                  .
                  .
                  .
                  .
FROM HOMOG H
WHERE H.NOME = CAMPO_PRINCIPAL_ESTRUTURA_HOMOGENEA -- Definido na fase de
mapeamento da extração
```

Quadro 10 – Instrução SQL base das regras de transformação

Conforme observado na figura acima, mas regras de transformação farão parte da instrução "*select*" executada na estrutura homogênea principal, sendo que cada regra será um "*sub-select*" com ligação direta à estrutura principal. Os "*sub-selects*" deverão obedecer certos padrões de comportamento, para que o seu resultado possa ser qualificado como uma medida ou uma dimensão. Estes comportamentos são:

a) retornar um único valor, que irá representar o valor de uma medida ou dimensão na tabela de fato do DW;
b) para facilitar o entendimento nos "*sub-selects*", todas as clausulas presentes na instrução "*from*" deverão possuir alias que obedecem o seguinte padrão: sigla da estrutura homogênea + valor do campo nome da estrutura homogênea. Exemplo: T_ANO para a estrutura de tempo relacionada ao período, S_BAIRRO para a estrutura de tempo relacionada à sub-categoria;

c) estar diretamente relacionado com a estrutura principal, com uma operação de união (*join*) diretamente com a estrutura principal representada pelo apelido (*alias*) “H”. Exemplo: “where H.ID = S_BAIRRO.ID_HOMOG”;

d) para obtenção dos valores das estruturas deve-se usar sua coluna “VALOR”. Exemplo: T_ANO.VALOR, S_BAIRRO.VALOR;

e) quando necessário mais de uma estrutura para recuperar um valor, usar o apelido (alias) da estrutura no formato H_NOME_ESTRUTURA. Exemplo:

No quadro 11 tem-se um exemplo de SQL que foi montado para funcionar como regra de transformação.

```
SELECT T_ANO.VALOR
   FROM HOMOG_TEMPO T_ANO
WHERE H.ID = T_ANO.ID_HOMOG AND
        T_ANO.NOME = ‘ANO’
```

Quadro 11 – *Select* simples de transformação de dados

Conforme explicado anteriormente, este “*select”* será um “*sub-select”* dentro da estrutura principal de transformação. No quadro 10, nota-se que o resultado retorna um único valor relacionado ao tempo de um fato. Caso necessário alterar o valor principal de retorno deste “*select”* o usuário poderá aplicar qualquer função de formatação de valor presente no banco de dados da Oracle , como por exemplo, a função “*format”*, função “*substring*” dentre outras.

No quadro 12, será apresentado o exemplo de uma regra de transformação que realiza uma operação de subtração entre duas estruturas que estão no nível de sub-categoria.

```
(SELECT   S_VAF.valor
    FROM  SUB S_VAF
  WHERE   H.ID = S_VAF.id_homog AND
          S_VAF.NOME = 'VALOR_ADICIONADO_FINAL')
                       -
 (SELECT  S_VAO.valor
      FROM     SUB S_VAO
   WHERE  h.ID = S_VAO.id_homog AND
          S_VAO.NOME = 'VALOR_ADICIONADO_ORIGINAL')
```

Quadro 12 – Usando operações para transformação dos dados.

Devido à forma com que a estrutura homogênea de dados foi construída, nota-se que os atributos “Nome” da estrutura homogênea funcionam como o nome da tabela em uma instrução “*select”* em um banco de dados normal, assim como o atributo “Valor” da estrutura homogênea funciona como o nome da coluna do qual se deseja recuperar o valor.

A seguir tem-se o quadro que exemplifica a criação de uma regra de transformação que atue na estrutura homogênea gerada através da extração de um banco de dados normalizado, onde a estrutura principal possui uma chave estrangeira para uma entidade que contém informações relevantes a sua composição. Tem-se os dados principais armazenados em uma estrutura homogênea, como nome do cliente, valor vendido, período de vendas e a chave estrangeira que representa o bairro, que representa o valor de uma tabela normalizada no banco de dados do sistema legado. Em outra estrutura tem se as informações referentes aos bairros extraídos da tabela normalizada do sistema legado. Através da estrutura homogênea, que guarda a chave primaria do sistema legado, é possível fazer a estrutura principal reconhecer as características do bairro que estão presentes em uma outra estrutura. No quadro 13 tem-se um exemplo de como funciona o *script* de transformação de dados.

```
SELECT    S_BAIRRO_NOME.VALOR //nome do bairro de retorno

   FROM   SUB S_BAIRRO, //sub-categoria da estrutura principal
          HOMOG H_BAIRRO, // estrutura que contém as
informações do bairro
          SUB  H_BAIRRO_S_NOME// categoria da estrutura bairro
que contém o nome

WHERE     S_BAIRRO.ID_HOMOG = H.ID AND // ligação obrigatória
com a estrutura principal
          S_BAIRRO.NOME = 'FK_BAIRRO' AND
          H_BAIRRO.NOME = 'BAIRRO' AND
          H_BAIRRO_S_HOMOG.NOME = 'NOME_BAIRRO' AND

          //recuperando informações da estrutura bairro
          H_BAIRRO.ID = H_BAIRRO_S_NOME.ID_HOMOG AND
          S_BAIRRO.VALOR = H_BAIRRO.ID_LEGADO //chave
estrangeira do bairro da estrutura principal com a chave
primaria do bairro
```

Quadro 13 – Regra de transformação usando estruturas homogêneas diferentes para recuperar informações relevantes a uma mesma dimensão

Cabe ao usuário do sistema Freedom decidir como irá montar as suas estruturas homogêneas da melhor manipulá-las na etapa de transformação. As chaves de ligação entre diferentes estruturas podem ser campos em comum entre elas, como no caso de empresas, pode se usar a Inscrição Estadual ou o CNPJ como chave de ligação, desde que não haja duplicidade destas chaves dentro da estrutura. Armazená-las dentro da estrutura principal, ou nas suas categorias vai depender da necessidade do usuário e de como os dados estão estruturados no sistema legado.

A seguir, tem-se o quadro 14 com as linhas de código mais relevantes do processo de transformação de dados.

```
public string ExecutaTransformacao(CargaCDL carga)
        {

            //Variáveis de controle
            TransformacaoBLL tBLL = new TransformacaoBLL();
            string msgRetorno = string.Empty;
            bool pararProcesso = false;

            foreach (TransformacaoCDL trans in carga.Transformacoes)
            {
                RefreshTransformacaoDados(trans);
                if (pararProcesso)
                    break;

                //Executa o processo de converção dos dados.
                try
                {
                    //tBLL.ConverteDados(trans);
                }
                catch (Exception ex)
                {
                    msgRetorno = "Ocorreu o seguinte erro durante a
                        converção de dados: " + ex.Message;
                    return msgRetorno;
                }

                foreach (ModeloLogicoCDL ml in trans.ModelosLogicos)
                {
                    if (pararProcesso)
                        break;

                    string sqlTransformacao = "SELECT ";
                    this.CurrentSession.Refresh(ml);
                    foreach (UnidadeLogicaCDL ul in ml.UnidadesLogicas)
                    {
                        if ((ul.Dimensao != null) && (ul.Medida != null))
                        {
                            msgRetorno = "Uma únidade lógica deve possuir
                              uma medida ou uma dimensão, nunca ambas.";
                            pararProcesso = true;
                            break;
                        }

                        if (ul.Dimensao != null)
                        {
                            sqlTransformacao += CriaRegraTransformacao(ul)
                                    + " as " + ul.Dimensao.Nome + ", ";
                        }
                        if (ul.Medida != null)
                        {
                            sqlTransformacao += CriaRegraTransformacao(ul)
                                    + " as " + ul.Medida.Nome + ", ";
                        }
                    }
                    sqlTransformacao =
                    sqlTransformacao.Remove(sqlTransformacao.Length - 3);
                    sqlTransformacao += " FROM HOMOG H ";
                    sqlTransformacao += " WHERE H.NOME = " +
```

```
                        ml.Mapeamento.Campo.Descricao;

                    LimpaSQL(ref sqlTransformacao);
                    OracleConnection conexao = new OracleConnection("Data
                        Source=xe; User Id=FREEDOM; Password=*****");
                    OracleDataAdapter da = new
                        OracleDataAdapter(sqlTransformacao, conexao);
                    DataSet ds = new DataSet();
                    da.Fill(ds);

                    //corrige os nomes das colunas no DataSet
                    for (int i = 0; i <= ml.UnidadesLogicas.Count - 1;
                              i++)
                    {
                        ds.Tables[0].Columns[i].ColumnName =
                              ml.UnidadesLogicas[i].RetornaNomeCampo;
                    }
                    msgRetorno = ExecutaCargaDados(ml, ds);
                }
            }
            return msgRetorno;
        }
```

Quadro 14 – Código fonte do processo de transformação dos dados

3.3.2.13 Carga de dados no *Data Warehouse*

O processo de carga de dados consiste na inserção dos dados extraídos de um sistema legado, que sofreram processos de integração e transformação para dentro de um DW. A periodicidade deste processo está diretamente ligada a regra de negócio e as necessidades do proprietário do DW. Pode ser feita a carga diária do DW, tendo como fator crítico para o sucesso o período que a carga será executada. Como se trata de um processo demorado que exige recursos de processamento massivo de informações, é recomendável que ele seja feito em horários de pouquíssima ou nenhuma atividade que exija recuperação de dados do DW.

O processo de carga de dados realizado pelo Freedom é conseqüência direta do processo de transformação. Ao finalizar o processo de transformação, os *scripts* de inserção de carga são criados. Este *script* de inserção é montado de acordo com os modelos lógicos e suas respectivas unidades lógicas. Cada unidade lógica representa um campo que será inserido no DW. Serão realizadas cargas diretas na tabela de fato, nunca fará cargas exclusivas em tabelas de dimensão. Cada linha do "*script*" de inserção de dados irá representar um fato e trará consigo as suas medidas e dimensões.

As dimensões são tratadas de forma que as diversas dimensões que acompanham o registro do fato serão inseridas em suas respectivas tabelas caso elas ainda não existam. O

sistema faz um processo de verificação da existência da dimensão antes da sua inserção. Após a sua inserção, o sistema irá retornar para o *script* de carga o identificador único daquele registro, já que nas tabelas de fato, as dimensões são representadas por chaves estrangeiras. No caso da existência de uma dimensão durante a inserção na tabela de fato, o sistema irá retornar o identificador único que representa aquela dimensão para o *script* de carga de dados.

A seguir, tem-se o quadro 15 com o código fonte que realiza o processo de carga para o DW.

```
public string ExecutaCargaDados(ModeloLogicoCDL ml, DataSet ds)
{
            //carregando variáveis principais
            string log = string.Empty; //log de execuçao dos scripts
            string cabecalho = string.Empty; //Armazena o cabeçalho dos
                                                    inserts
            string InsertDw = string.Empty; //recebe o corpo do insert
            string msgRetorno = string.Empty;

            OracleConnection conDW = new OracleConnection(ml.Transformacao
                                        .DataWarehouse.StringConexao);
            if (conDW.State != ConnectionState.Open) //Abre a conexão com
                                                        o DW
                conDW.Open();
            OracleCommand cmd = new OracleCommand();
            cmd.Connection = conDW;
            cabecalho = MontaCabecalhoInsert(ml);//Monta o cabeçalho do
                                                    insert principal
            OracleTransaction transacao = conDW.BeginTransaction();
            foreach (DataTable table in ds.Tables)
            {
                foreach (DataRow row in table.Rows)
                {
                    InsertDw = cabecalho;
                    InsertDw += MontaInsert(ml, row, conDW) + ")";
                    cmd.CommandText = InsertDw;
                    log += InsertDw + ";";
                    try
                    {
                        cmd.ExecuteNonQuery();
                    }
                    catch (Exception ex){
                        msgRetorno = "Ecorreu o seguinte erro durante a
                                        carga de dados: " + ex.Message;
                        return msgRetorno;
                    }
                }

            }
            transacao.Commit();
            conDW.Close();
            return "Carga de dados realizada com sucesso!";
}
```

Quadro 15 – Código fonte responsável por realizar o processo de carga dos dados

3.3.2.14 Processamento do cubo

Realizado o processo de carga de dados no DW é necessário efetuar o processamento do cubo de decisão, para que os novos dados carregados no DW estejam disponíveis para os usuários de relatórios OLAP.

Este processamento consiste em carregar os novos dados do DW na estrutura do SSAS. É nesta etapa que as agregações dos dados planejados para o cubo de decisão acontecem. Processar o cubo envolve a leitura das tabelas que correspondem às dimensões para popular os níveis e membros dos dados presentes na tabela de fato. Somente após o processamento do cubo que ele poderá ser usado para realizar pesquisas OLAP.

Para realizar o processamento do cubo, o Freedom possui uma classe de configuração que contém as definições principais do cubo a ser processado, como dados para a conexão com o serviço SSAS, o nome da conexão e o nome do cubo a ser processado.

O algoritmo de processamento cria uma conexão com o serviço SSAS, enviando comando para execução do processamento completo da estrutura do cubo de decisão. No quadro 16 tem-se o algoritmo de processamento do cubo de decisão.

```
public void ProcessaCubo()
        {
            AdomdConnection cn = new AdomdConnection("Provider=MSOLAP;Data
                  Source=" + serverName + ";Initial Catalog=" +
                  databaseName);
            cn.Open();

            AdomdCommand cmd;

            cmd = cn.CreateCommand();

            cmd.CommandType = CommandType.Text;



            cmd.CommandText = "<Batch
                  xmlns=\"http://schemas.microsoft.com/
                  analysisservices/2003/engine\"><Parallel><Process>
                  <Object>"+"<DatabaseID>" + databaseID + "</DatabaseID>" +
                  "<CubeID>" + cubeID + "</CubeID> " +
                  "</Object><Type>ProcessFull
                  </Type><WriteBackTableCreation>
                  UseExisting</WriteBackTableCreation> </Process>
                  </Parallel> </Batch>";
            try
            {
                Console.WriteLine("Executing Command...");
                cmd.ExecuteNonQuery();
            }

            catch (Exception ex)
```

```
            {
                Console.WriteLine(ex.Message);
            }

            finally
            {
                cn.Close();
                Console.WriteLine("Finished");
            }
        }
```

Quadro 16 – Algoritmo de processamento do cubo de decisão

3.4 RESULTADOS E DISCUSSÃO

O desenvolvimento deste trabalho proporcionou a automatização dos processos de carga de dados realizados pela equipe de desenvolvimento da FEESC em seus DWs, substituindo a mão de obra na criação de longos *scripts* de migração de dados, sem a criação de metadados do DW. Proporciona-se desta forma, a configuração e ativação do processo de carga de dados a longa distância, sem a necessidade de deslocamento de seus funcionários, reduzindo custos na prestação de serviços para seus clientes.

O antigo processo de carga de dados acontecia através da criação de várias visões materializadas do banco de dados Oracle do sistema legado. Essas visões são responsáveis por reunir todas as informações necessárias para o processo de carga. Porém, essas visões materializadas consomem uma grande quantidade de espaço físico do banco de dados, além de serem redundantes. Em seguida é criada outra visão para agrupar todos os dados obtidos pelas visões materializadas previamente criadas. Esta visão será usada como fonte de dados para o processo de carga DW.

Não havia uma estrutura homogenia para armazenar os dados, muito menos a possibilidade de integrar dados de sistemas diferentes usando as visões materializadas para agrupamento dos dados.

O antigo processo de carga de dados consistia na execução de *scripts* PL/SQL do banco de dados Oracle, que criavam cursores com base na visão materializada criada para agrupar os dados, e inseriam dados diretamente nas tabelas de dimensões e tabelas de fato do DW. Todas as regras do negócio estão implementadas nestes *scripts* de inserção, praticamente inviabilizando a sua reutilização. Devido à regra estar fixa no *script*, dificultava a sua localização e possíveis correções sem afetar outros processos.

A implementação do sistema Freedom proporcionou a organização dos processos de carga de DW da FEESC, que agora estão bem divididos e são organizados e estruturados um a um. As regras de negócio extração e transformação estão mais claras, proporcionado a sua discussão e possibilidades de adequação.

Os problemas encontrados no antigo processo de carga de dados da FEESC, dentre eles está a falta de padronização na execução do processo, foram resolvidas com a implementação do sistema Freedom. A quantidade de tempo gasta por analistas e programadores para executar o processo de carga era muito grande. A freqüência com que a carga de dados é realizada está abaixo da necessária, não havendo um mecanismo de agendamento para a sua execução.

Em relação a ferramentas que também realizam a extração, transformação e carga de dados, como o Kettle, citado nos trabalhos correlatos, a grande vantagem do sistema desenvolvido neste trabalho é a possibilidade de configurar e executar o processo de carga de dados *online*, sendo esta a maior necessidade da FEESC. Ainda é uma ferramenta limitada graficamente se comparado ao Kettle, e necessita de profissionais com elevado conhecimento de estrutura de banco de dados e DW, além do domínio da linguagem SQL.

Assim como a aplicação desenvolvida neste trabalho, a base para modelagem dos processos de extração e transformação do Kettle está baseada nos metadados do DW. Isso garante a qualidade dos dados que estão sendo inseridos no DW, pois proporciona a identificação da origem dos dados e todos os processos que foram realizados justificando sua atual situação.

Outro ponto a ser considerado no sistema desenvolvido é a sua curva de aprendizagem, que necessita da compreensão da estrutura homogênea para poder elaborar as regras de transformação de dados, exigindo alto nível de abstração por parte dos seus usuários.

Realizando uma comparação com a ferramenta Kettle, que também implementa a funcionalidade de carga de dados em um DW, pode se observas várias particularidades entre os dois softwares. O Kettle possui uma ferramenta visual destinada somente à modelagem dos fluxos do processo ETL. Neste trabalho ainda não há uma ferramenta visual para modelagem do fluxo do processo de migração, ao invés disso o usuário irá cadastrar estes mapeamentos de forma intuitiva em formulários HTML.

Este projeto não é tão abrangente quanto a funcionalidades e tão pouco madura como a ferramenta Kettle. Não é objetivo do projeto desenvolvido neste trabalho concorrer diretamente com as ferramentas ETL disponíveis no mercado. Esta aplicação surgiu da necessidade da equipe de desenvolvimento de software da FEESC de realizar processos de

carga de dados em seus clientes de forma remota, sem a necessidade de deslocamento de seus profissionais, visando ganho de tempo e economia de recursos financeiros.

A grande novidade apresentada pela ferramenta Freedom é o fato de ela ser totalmente *online*. Uma vez implantada no servidor de um cliente, os desenvolvedores responsáveis pela manutenção do DW terão acesso em qualquer computador que possua conexão com a internet, podendo configurar processos de carga de dados e agendá-los para serem executados em um determinado horário do dia.

A aplicação desenvolvida neste trabalho ainda não possui uma *interface* amigável para o usuário final, demandando conhecimento técnico de banco de dados, sendo implementada para os técnicos responsáveis por manter DW da FEESC, diferente da ferramenta Kettle, que possui uma *interface* visual para o usuário final.

A seguir, é apresentado o quadro 17, comparando-se a ferramenta Kettle e a ferramenta Freedom,

	Kettle	Freedom
Ferramentas visuais para elaboração de regras	X	
Mantém metadado	X	X
Processamento online		X
Suporte a múltiplos banco de dados	X	
Integração com SSAS		x

Quadro 17 – Comparação das ferramentas Kettle e Freedom.

4 CONCLUSÕES

Já existem ferramentas específicas no mercado que executam o processo de extração, transformação e carga de dados em DW, porém as ferramentas analisadas pela FEESC não possuem *interface* web que possibilite executar o processo de carga *online*, sem a necessidade de deslocar seus profissionais. A principal vantagem oferecida pela ferramenta desenvolvida neste trabalho é a possibilidade de configurar e executar os processos de extração, transformação e carga de dados *online*, reduzindo de forma considerável a necessidade de deslocamento de profissionais para manutenção do DW.

Em Inmon, Welch e Glassey (1999) cita-se várias vantagens e desvantagens no uso de uma ferramenta para automatizar o processo ETL em um DW. Dentre as vantagens citadas, as que foram obtidas com o desenvolvimento do sistema Freedom foram:

a) programas podem ser construídos rapidamente;
b) produção automática de metadados;
c) redução significante dos custos;
d) redução da possibilidade de falhas em processos conhecidos;
e) redução na manutenção nos processos ETL;
f) alocação de programadores em outras tarefas.

Realizando uma comparação aos itens citados por Inmon, Welch e Glassey (1999), a respeito das desvantagens de não usar ferramentas automatizadas com os processos manuais realizados pela FEESC, pode-se observar itens:

a) a lógica que um programador emprega na torna-se intelectualmente entediante após as primeiras quatro ou cinco execuções de carga manualmente;
b) metadados não são reunidos automaticamente;
c) os processos levam muito tempo para serem construídos;
d) os processos são constantemente alterados, tornando-se um fardo a sua manutenção;
e) custos de infra-estrutura são altos.

Inmon, Welch e Glassey (1999) citou nos itens de desvantagens de não usar uma ferramenta para o processo ETL exatamente o que acontece no ambiente de desenvolvimento de DW da FEESC.

O sistema conseguiu atender ao seu objetivo principal e seus objetivos específicos, automatizando o processo de carga de dados em DW e processando os cubos de decisão (OLAP) proporcionando a visualização dos novos dados do DW em relatórios analíticos.

Esta ferramenta foi construída para atender as necessidades de desenvolvimento da FEESC, que trabalha somente com banco de dados da empresa Oracle. Portanto, esta aplicação é limitada ao uso da tecnologia Oracle para acessar os dados.

Ferramentas visuais, presentes em outros sistemas que executam processos ETL não estão presentes nesta aplicação, devido a restrições quanto ao tempo de desenvolvimento e complexidade do uso de tecnologias que proporcionem a manipulação visual de dados na internet.

Por fim, este trabalho veio engrandecer e muito em termos de conhecimentos pessoais, sobre manipulação de dados, técnicas utilizadas em um DW e na própria lógica de programação web, se teve a oportunidade de aprimorar os conhecimentos em C#, mas também no que diz respeito à superação de todas as dificuldades enfrentadas para realização e conclusão deste trabalho.

4.1 EXTENSÕES

Neste trabalho foram desenvolvidas rotinas que executam o processo ETL em um DW, processo este extremamente complexo pelo fato de manipular dados de origens e estruturas diversas, limitado a tecnologia de banco de dados Oracle. Existem muitas funcionalidades que podem ser inclusas nesta aplicação, dentre as de maior relevância se destacam:

a) construção de ferramentas visuais para construção de regras de extração e transformação dos dados;
b) implementação de relatórios de cargas;
c) tornar a ferramenta compatível com uma variedade maior de banco de dados;
d) possibilitar a leitura de documentos, como planilhas do Excel e arquivos XML;
e) possibilidade de visualização gráfica dos modelos ETL gerados pela aplicação, como diagramas e mapeamentos dos modelos lógicos e unidades lógicas criadas nos processos de carga;
f) sistema que monitore as alterações dos dados de forma automática, evitando carga de dados desnecessários.

REFERÊNCIAS BIBLIOGRÁFICAS

BAPTISTA, E. **Alternativa para migração para ambientes *Data Warehouse***. 1998. Monografia de Pós Graduação(Tecnologia de Desenvolvimento de Sistemas) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BARBIERI, C. **BI Business Intelligence Modelagem & Tecnologia**. Rio de Janeiro. Axcel Books, 2001.

BRACKETT, Michael H. **The *Data Warehouse* Chalange**. Nova York: Editora Wiley Computer Publishing, 1996.

CODE ONTIME. **Code Generation**. San Diego. Disponível em: < http://codeontime.com/default.aspx>. Acesso em: 26 out. 2009.

COOLITE INK. **Coolite Toolkit**. Alberta. Disponível em: < http://www.coolite.com>. Acesso em: 26 out. 2009.

GONSALVES, M. **Extração de dados para *Data Warehouse***. Rio de Janeiro. Axcel Books, 2003.

INMON, W H; **Como construir o *Data Warehouse***. Rio de Janeiro: Editora Camous Ltda., 1997

INMON, W H; TERDEMAN, R H; IMHOFF, C. **Data Warehousing**: como transformar informações em oportunidades de negócios. São Paulo: Editora Barkeley, 2001.

INMON, W H; WELCH, J D; GLASSEY, K, L . **Gerenciando *Data Warehouse***. São Paulo: Editora Makron Books, 1999.

JACOBSON, R; STACIA, M. **Microsoft SQL Server Analysis Services**. Porto Alegre: Editora Bookman, 2007.

KIMBALL, R. **The *Data Warehouse* Toolkit**. Nova York: Editora John Wiley & Sons Inc., 1996.

MACHADO, F. N. R. **Tecnologia e projeto de *Data Warehouse*:** uma visão multidimensional. São Paulo: Érica, 2004.

PENTAHO. **Kettle:** data integration. Orlando. Disponível em: < http://kettle.pentaho.org/>. Acesso em: 15 nov. 2009.

ROB, C; HANSELMAN, S; HAACK, P; GUTHRIE, S. **Asp.Net MVC 1.0**. Indianápolis: Wiley Publishing, 2009.

SANZON, G. **Sistema de Informação Gerencial Baseado em *Data Warehouse* Aplicado a uma Software House**. 2006. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SCHENKER, Gabriel. **Your first NHibernate based application**.2009. Disponível em: < http://nhforge.org/wikis/howtonh/your-first-nhibernate-based-application.aspx >. Acesso em: 10 nov. 2009.

SHAMMAS, Gabriel. I. J. **Esquema Estrela**. São Paulo. 2009. Disponível em: < http://www.shammas.eng.br/acad/sitesalunos0106/012006dtw/modelagem.htm>. Acesso em: 10 nov. 2009.

SINGN, H. S. ***Data Warehouse*:** Conceitos, Tecnologias, Implementação e Gerenciamento. São Paulo: Makron Books, 2001.

STRUBE, J. H. **Estudo de um caso real de migração de banco de dados de sistemas transacionais para *Data Warehouse***. 2001. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.