UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

Incorporação do Processo de Engenharia Reversa de Bancos de Dados Relacionais na Ferramenta brModeloNext

Aluno: Elton Luis Hoffmann Junior

Orientador: Prof. Dr. Ronaldo dos Santos Mello

Florianópolis – SC

2013/2

Sumário

Lista de Figuras

Lista de Siglas

ER: Entidade-Relacionamento

EER: Entidade-Relacionamento Estendido

BD: Banco de Dados

SGBD: Sistema Gerenciador de Banco de Dados

SQL: Structured Query Language

GBD: Grupo de Banco de Dados

UFSC: Universidade Federal de Santa Catarina

MVC: Model-View-Controller

XML: Extensible Markup Language

DDL: Linguagem de definição de dados

ANSI: Instituto Nacional Americano de Padrões

DTD: Definição de Tipo de Documento

TCC: Trabalho de Conclusão de Curso

1 Introdução

É pública e notória, para os usuários e criadores de bancos de dados, a importância de uma modelagem conceitual que o represente, o que comumente é feito utilizando o modelo Entidade-Relacionamento (ER). Entre outras coisas, uma modelagem conceitual contém informações sobre quais dados serão armazenados no banco de dados, porém, sem aprofundar-se em detalhes de baixo nível, e ignorando a maneira com que tais dados serão armazenados fisicamente. Para Heuser (2009), “Esta forma de proceder permite envolver o usuário na especificação do banco de dados, pois os detalhes compreensíveis por técnicos ainda não estão sendo definidos, estamos apenas descrevendo objetos de uma organização”. Date (2004), ainda diz que, “Em termos gerais, a visão conceitual pretende ser uma visão dos dados ‘como eles realmente são’, em vez de forçar os usuários a vê-los pelas limitações (por exemplo) da linguagem ou do hardware que eles possam estar utilizando”.

Para os projetistas do banco de dados, tal abordagem permite que estes pensem com mais clareza sobre as regras de negócio do banco, postergando os detalhes de implementação; para os usuários e mantenedores do banco, essa modelagem permite um melhor entendimento da lógica encontrada, facilitando a localização dos dados e a manutenção do esquema do banco; para a organização, serve para especificar formalmente as características da sua base de dados, mantendo a substancialidade e contribuindo para a persistência do conhecimento gerado.

Uma das ferramentas mais populares no meio acadêmico para a modelagem de bancos de dados relacionais é a brModelo, criada pelo Grupo de Banco de Dados (GBD) da Universidade Federal de Santa Catarina (UFSC) e, recentemente, a sua sucessora, a brModeloNext, que se encontra ainda em desenvolvimento [BRMODELO, 2013]. Ela consiste em uma aplicação independente de plataforma, que permite a criação de diagramas conceituais e lógicos para o esquema do banco de dados relacional que está sendo desenvolvido. Após esta criação, é permitido que o usuário processe o esquema criado, gerando assim o modelo físico do banco de dados, podendo este então ser inserido no Sistema Gerenciador de Bancos de Dados (SGBD)

de sua escolha. Menna (Menna et. al, 2011) cita vantagens desta ferramenta, em comparação com outras soluções presentes no mercado: “Um diferencial da brModeloNext é a flexibilidade no mapeamento de esquemas conceituais para esquemas lógicos. Ela executa, de forma semi-automática, o processo de mapeamento, oferecendo a possibilidade de escolha de uma (1) dentre diversas alternativas de conversão de um conceito da modelagem EER. Assim, o usuário tem a liberdade de orientar a conversão para uma estrutura lógica mais adequada a um dado domínio.“

No estudo de bancos de dados, um ponto importante é a engenharia reversa. Trata-se de um algoritmo não-automatizado capaz de extrair dados relativos ao esquema relacional de uma base de dados existente e através disso gerar um esquema conceitual propriamente normalizado. Partindo para um ambiente corporativo, tem-se que nem sempre as boas práticas de modelagem são utilizadas; frequentemente são encontrados bancos de dados sem documentação, o que acarreta na dificuldade da manutenção rotineira da aplicação que consome o banco de dados. Outro exemplo demonstrando a importância de possuir um esquema conceitual de um banco já implementado é a migração de tecnologias desejando-se conservar as características lógicas e conceituais do esquema.

No entanto, o mercado carece de uma aplicação que realize as etapas de engenharia reversa para bancos de dados. Da mesma forma, no ambiente acadêmico, em disciplinas avançadas de BD, existe a necessidade de uma ferramenta que permita a realização e visualizar o processo de engenharia reversa.

Com o preenchimento dessa lacuna em mente, define-se o objetivo deste trabalho: a incorporação do processo de engenharia reversa à ferramenta brModeloNext, aproveitando as facilidades de interatividade da ferramenta e tornando-a mais completa em termos de atividades de projeto de bancos de dados.

Um maior detalhamento sobre esses aspectos serão abordados nos capítulos seguintes. O capítulo 2 apresenta a base teórica na qual se fundamenta este trabalho. No capítulo 3 é apresentada a ferramenta

brModeloNext. O capítulo 4 contém os detalhes da implementação realizada ao incorporar o processo de engenharia reversa à brModeloNext. O capítulo 5 conclui o presente trabalho e o capítulo 6 sugere trabalhos futuros referentes ao mesmo.

2 Fundamentação teórica

2.1 Projeto de banco de dados

Na área de desenvolvimento de *software*, é comum encontrar livros e manuais que sugerem ao programador melhores abordagens na resolução de determinados problemas. Tais técnicas, parte integrante da área de *Engenharia de Software*, são conhecidas como *Padrões de Projetos*, e podem ser encontradas para qualquer tecnologia ou metodologia já consolidada, e para qualquer grau de abstração dos problemas, desde padrões de arquitetura de *software* até técnicas de legibilidade de código. Pierin (2011) descreve que: “O padrão de projeto nada mais é do que um modelo de uma experiência amplamente testada e aprovada que pode ser usado como um guia para a resolução de problemas específicos”.

Também na área de bancos de dados esses padrões são encontrados. Tais práticas permitem um armazenamento consistente, facilitando a manutenibilidade do BD; e um eficiente acesso aos dados, garantindo prontidão de resposta das requisições.

Um banco de dados bem projetado implica em diversos benefícios futuros como:

i. Garantia de padrões. Ao definir formatos padronizados para exibição, relatórios, terminologias, etc., está-se facilitando a comunicação e cooperação dos usuários e garantindo a robustez do sistema;
ii. Redução do tempo de desenvolvimento da aplicação. Quando um banco de dados está pronto e funcionando, o tempo de desenvolvimento da aplicação que o consumirá tende a diminuir.
iii. Flexibilidade. Com boas modelagens de dados, não deve ser um problema a inclusão de novos itens ao domínio da aplicação. Também, ao serem feitas manutenções na estrutura do banco, é importante que os dados armazenados não sejam afetados.

iv. Disponibilidade. Utilizando-se bons SGBDs, garante-se a atualização imediata dos dados, mesmo tratando-se de bancos de dados distribuídos ou com alto nível de concorrência.
v. Economia de escalas. A consolidação de dados evita a redundância de informações. Além disso, conhecendo as necessidades do sistema, pode-se fazer uma modelagem que auxilie no desempenho da aplicação, reduzindo custos de processamento de dados [NAVATHE, 2011].

Com o propósito de modelar um banco de dados, e após ter sido feita a análise de requisitos, parte-se então à modelagem *de fato*, a qual seguirá um fluxo linear de três etapas, conforme exibido na *Figura 1*:

i. Modelagem conceitual. Uma descrição concisa dos requisitos de dados dos usuários. Esta modelagem contém detalhes dos tipos de entidades, relacionamentos e restrições;
ii. Modelagem lógica. Gerada de forma automática ou semi automática a partir da modelagem conceitual, esta modelagem é voltada à detalhes técnicos do futuro banco de dados. Ao término do processo, esta modelagem irá espelhar o banco de dados físico;
iii. Modelagem física. Etapa em que a organização dos arquivos, índices e caminhos de acesso serão especificados. É totalmente voltada ao SGBD escolhido, e seu resultado é código SQL pronto para ser executado a fim de criar o banco de dados [NAVATHE, 2011].

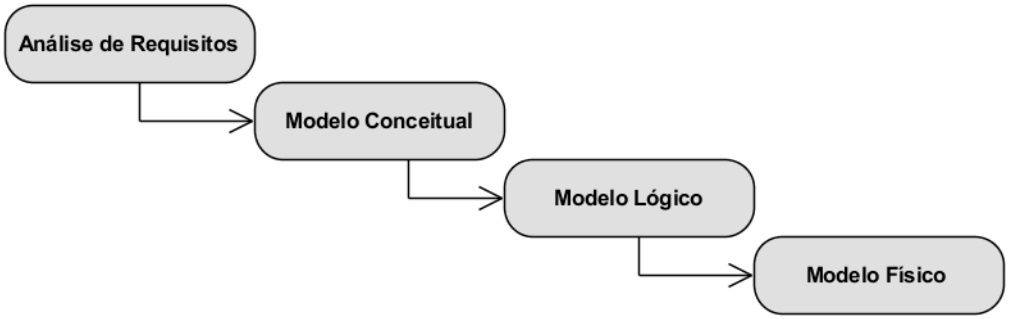

Figura 1: Fluxo de criação de um banco de dados relacional.

2.1.1 Modelagem Conceitual

A modelagem conceitual descreve as características do banco de dados de forma independente das tecnologias nele utilizadas. Nela, é descrito quais dados serão registrados, porém, sem entrar em detalhes de *como* eles serão armazenados em nível de SGBD [HEUSER, 2009].

Esta deve ser a modelagem a ser utilizada para o nível de conversão, entendimento, transmissão, validação de conceitos, mapeamento do ambiente, etc. [COUGO, 1997]. Também pode ser utilizada para garantir que todos os requisitos do usuário sejam atendidos e não estejam em conflito [NAVATHE, 2011].

É importante, para o processo de projeto de um banco de dados relacional, que não seja feita uma modelagem de dados diretamente em nível lógico. Embora isso seja possível, e por vezes induzido pela ferramenta de modelagem utilizada, estar-se-ia criando um esquema totalmente dependente da tecnologia de implementação [COUGO, 1997]. Além do mais, o esforço feito para a geração do esquema conceitual é igual ou, muitas vezes, menor do que o empregado para a criação de um esquema lógico. Tendo em vista que a transcrição entre estas duas modelagens é feita de forma automática ou semi automática, pode-se então concentrar os esforços nos aspectos conceituais, obtendo melhores detalhes sobre os objetos que interessam neste instante.

Após a obtenção da especificações de requisitos, dá-se início ao processo de modelagem. Primeiramente deve-se identificar um conjunto de conceitos úteis à aplicação [DATE, 2004]:

i. Identificam-se *entidades* que representam intuitivamente elementos concretos do mundo real.
ii. Classificam-se as entidades criadas, e diferencia-se uma da outra através de seus *atributos* individuais.
iii. Define-se, para cada entidade, uma ou mais propriedades que servirão como *identificadores* para aquela entidade.

iv. Faz-se o *relacionamento* entre entidades, criando-se assim uma lógica semântica para o modelo.

A técnica de modelagem mais difundida é o modelo Entidade-Relacionamento, na qual o esquema conceitual é descrito de forma diagramática em um Diagrama Entidade-Relacionamento, conforme exibido na *Figura 2*.

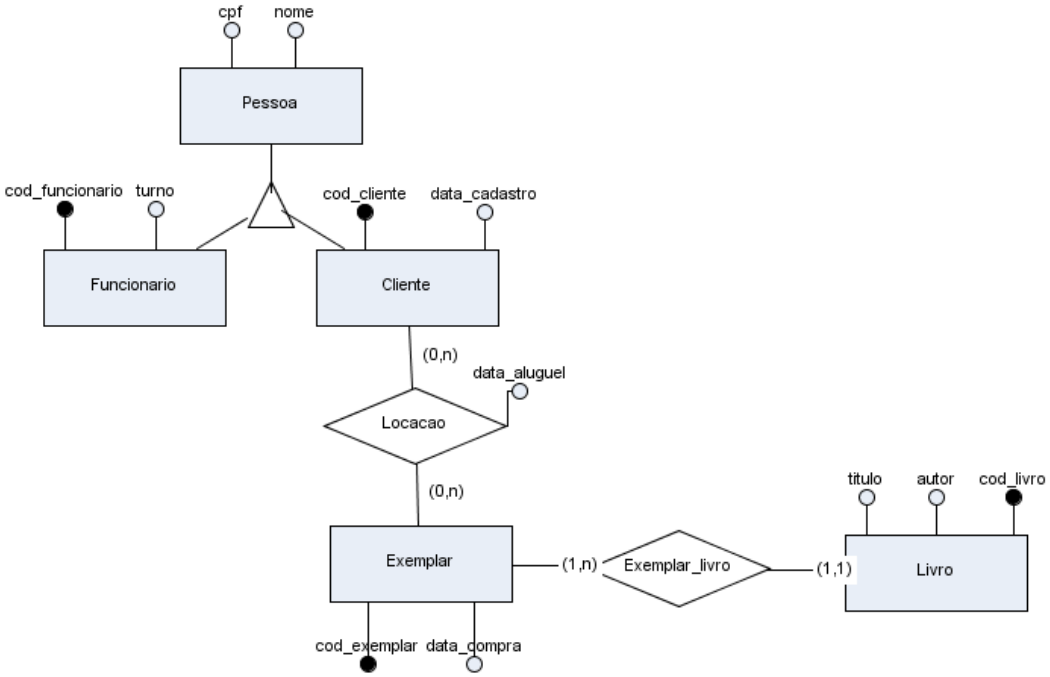

Figura 2: Exemplo de diagrama Entidade-Relacionamento.

2.1.1.1 Modelo Entidade-Relacionamento

O modelo ER é um modelo formal, preciso e não-ambíguo [HEUSER, 2009]. Suas características gráficas intuitivas tornam sua leitura elementar tanto para usuários humanos quanto para computadores; e sua precisão permite que leitores diferentes façam o mesmo entendimento de uma mesma modelagem ER.

Os conceitos centrais da modelagem ER são: *(i) Entidades*; *(ii) Atributos*; *(iii) Relacionamentos*; *(iv) Especializações* e *(v) Entidades Associativas*.

Uma entidade representa um objeto do mundo real. Tal objeto deve ser reconhecido por suas propriedades, que serão armazenadas em formas de

atributos. Uma entidade pode representar tanto objetos reais (e.g. Avião, Cliente) como objetos abstratos (e.g. Departamento, Projeto). Uma entidade é representada por um retângulo contendo o nome da mesma. A *Figura 3* representa uma entidade da forma como ela é exibida na brModeloNext. Na *Figura 2*, pode-se identificar as entidades *Pessoa, Funcionario, Cliente, Exemplar* e *Livro*.

Figura 3: Uma entidade.

Atributos são características que podem ser vinculadas à entidades e relacionamentos a fim de armazenar informações a respeito das mesmas. Atributos, simples ou compostos, podem ser utilizados como identificadores de entidades. São representados por círculos – cheios se forem identificadores, vazados se forem atributos simples – ligados às entidades. A *Figura 4* representa uma entidade com seus respectivos atributos, *titulo*, *autor*, e *cod_livro*, sendo que *título* e *autor* representam atributos simples, e *cod_livro* representa um atributo identificador.

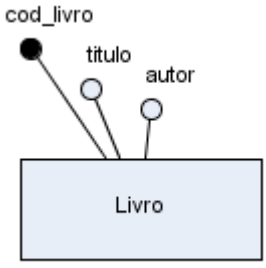

Figura 4: Atributos em uma entidade.

Entidades associam-se entre si por meio de *relacionamentos*. Ao definir-se relacionamentos, deve-se definir também uma cardinalidade mínima (i.e. o número mínimo de instâncias da entidade X que estão relacionadas com cada instância da entidade Y) e uma cardinalidade máxima (i.e. o número máximo de instâncias de X que estão relacionadas com cada instância de Y). Um

relacionamento pode se dar entre duas entidades (chamado relacionamento binário), entre instâncias da mesma entidade (chamado auto relacionamento, o qual exige uma identificação de papéis), ou entre *n* entidades (chamado relacionamento n-ário). Relacionamentos são representados por losangos. A *Figura 5* representa um relacionamento entre as entidades *Exemplar* e *Livro*.

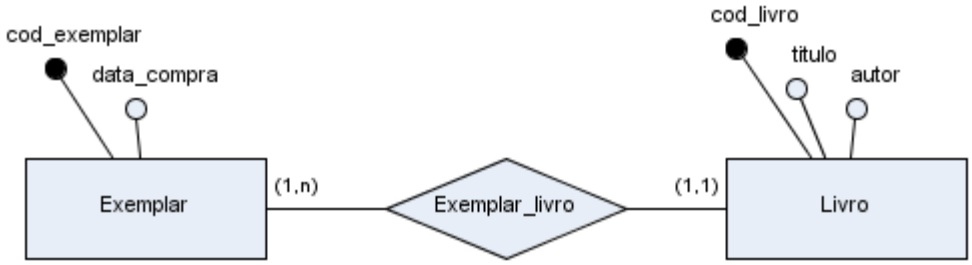

Figura 5: Um relacionamento entre entidades.

Além de relacionamentos e atributos, uma entidade pode possuir especializações. Ao definir uma entidade como uma especialização de outra, faz-se com que as propriedades da entidade *pai* sejam herdadas pela entidade *filha*. As especializações podem ser: *(i)* Parciais ou totais. No caso de especialização total, uma instância da entidade *pai* deve, impreterivelmente, pertencer também a uma entidade *filha*. Já em casos de especialização parcial, pode-se ter instâncias da entidade *pai* que não pertençam a nenhuma entidade *filha; (ii)* Exclusiva ou compartilhada. Caso a especialização seja exclusiva, diz-se que uma instância da entidade *pai* deve pertencer a apenas uma entidade filha. Por outro lado, em uma especialização compartilhada, um elemento *pai* pode possuir simultaneamente atributos de mais de uma entidade *filha*. Especializações são representadas por triângulos isóceles partidos da entidade *pai*. A *Figura 6* representa uma hierarquia de entidades conforme exibido na ferramenta brModeloNext. Nela, pode-se notar a entidade *Pessoa* sendo a entidade *pai*, da qual derivam as entidades *filhas Funcionário* e *Cliente*.

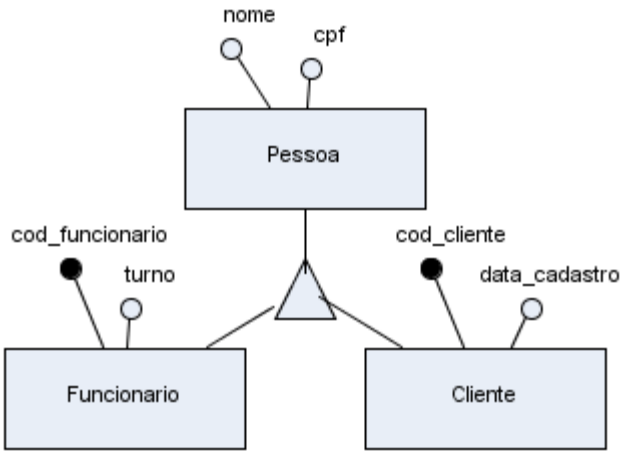

Figura 6: Uma hierarquia de entidades.

Por último, temos as entidades associativas, utilizadas quando deseja-se associar uma entidade a um relacionamento. Devido ao fato de que Relacionamentos não podem associar-se com outros Relacionamentos, e que Entidades não podem relacionar-se com Relacionamentos, fez-se necessário a criação deste outro objeto, utilizado quando deseja-se alterar uma modelagem permitindo uma associação a mais à um Relacionamento. A *Figura 7* demonstra como este objeto é representado na ferramenta.

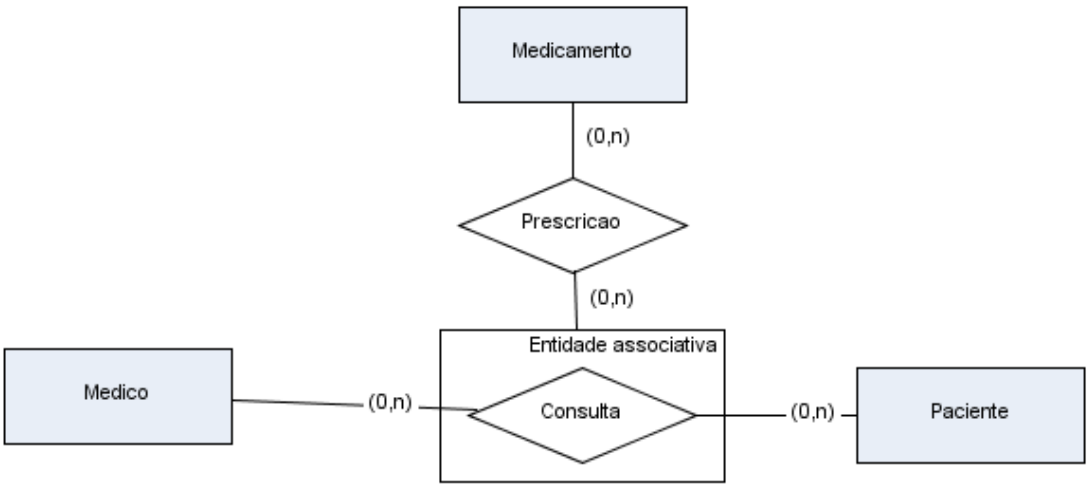

Figura 7: Uma Entidade associativa.

2.1.2 Modelagem Lógica

Diz-se da modelagem lógica aquela em que as entidades, atributos e relacionamentos, estão descritos de acordo com as regras de implementação de alguma tecnologia. No entanto, ainda não se tem a estrutura física do banco de dados. Esta modelagem é totalmente dependente de tecnologia, ou seja, deve-se conhecer o modelo de banco de dados (e.g. relacional, orientado a objetos, xml, etc.) e gerar uma modelagem lógica própria para tal.

Nesta fase são definidas características como, chaves estrangeiras (i.e. referências entre tabelas relacionadas) e tipagem dos dados. Também, é feito o tratamento adequado para as especializações, atributos multivalorados e relacionamentos encontrados na modelagem conceitual, identificando-se a melhor maneira de implementar a situação.

A *Figura 8* representa a modelagem lógica convertida a partir do esquema conceitual exibido na *Figura 2*.

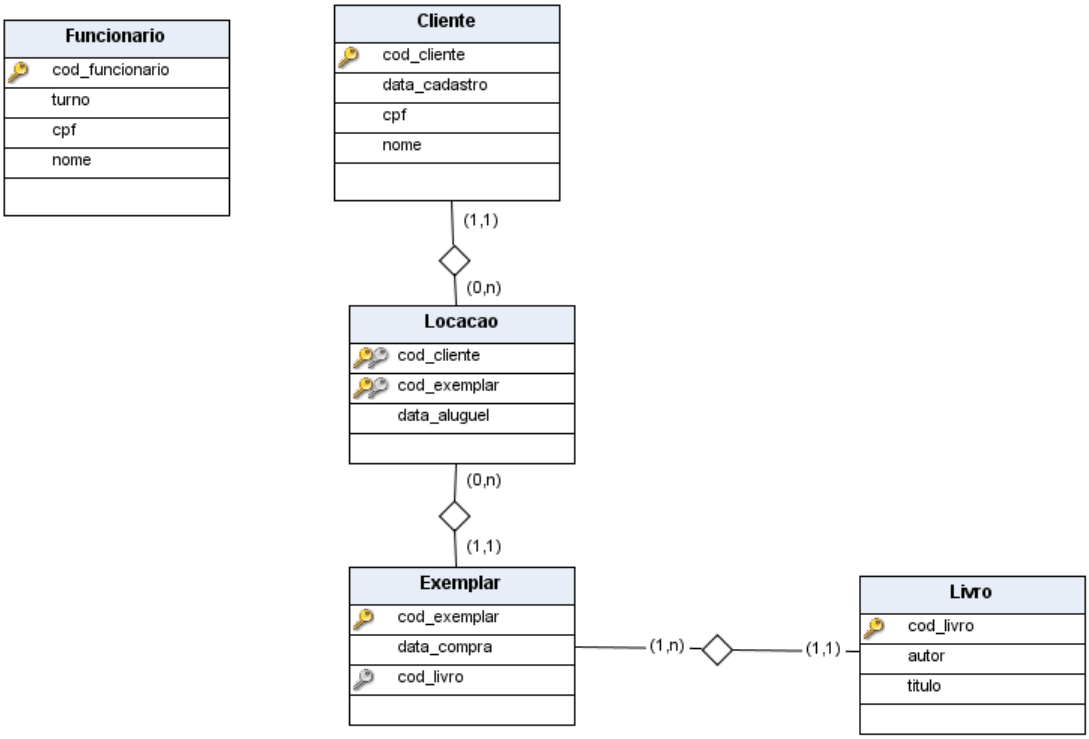

Figura 8: Exemplo de modelagem lógica.

2.1.3 Modelagem Física

Após o esquema do banco de dados ter sido alterado para abranger, além das características de domínio do problema, as regras de implementação, é realizado o passo de geração do código do banco de dados. Esta é uma etapa automatizada, a qual resulta em uma especificação SQL/DDL com o esquema do BD a ser criado.

Tratando-se de bancos de dados relacionais, este passo é dependente de SGBD, tendo em vista que cada sistema possui dialetos SQL diferentes, com diferentes maneiras de especificar restrições e procedimentos.

Além disso, vale ressaltar novamente que é nesta etapa que se definem índices e *views*, visando um bom desempenho de acesso ao BD.

2.2 Engenharia Reversa

Se na área de desenvolvimento de softwares, o processo de passar por etapas de análise de requisitos, planejamento e implementação é chamado de *engenharia de software*, pode-se definir como *engenharia reversa*, justamente a ação de percorrer o mesmo caminho, porém, no sentido contrário.

Para projetos de bancos de dados, vale a mesma definição. Parte-se de um banco de dados já implementado e elabora-se a partir dele um esquema conceitual referente às características apresentadas originalmente.

Heuser (2009), diz que: “No caso de banco de dados, fala-se de engenharia reversa quando modelos de dados mais ricos em detalhes de implementação são transformados em modelos de dados mais abstratos.”

Este fluxo pode ser visto na *Figura 9*.

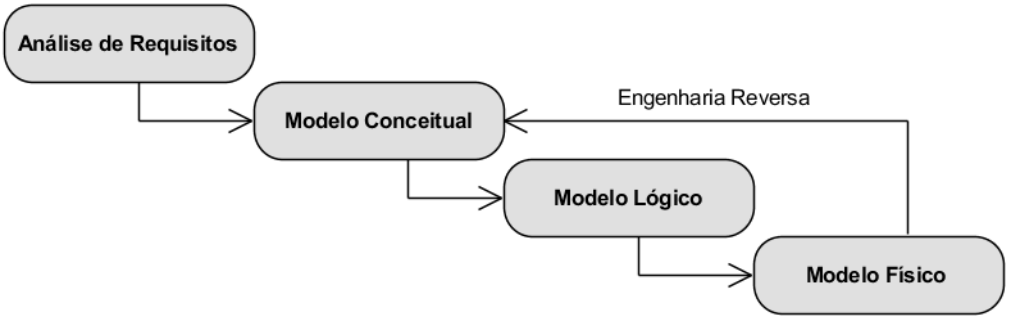

Figura 9: Fluxo de modelagem de banco de dados, incluindo Engenharia Reversa

A essência da utilização de um banco de dados consiste no desejo de persistir dados através do tempo. Para tal, exige-se, naturalmente, que o próprio banco de dados persista através do tempo. No entanto, vários fatores podem ocorrer que façam com que o esquema do banco de dados tenha que ser alterado. Estas alterações a nível de domínio do problema, assim como o próprio processo de criação de um BD, devem ser iniciadas pela modelagem conceitual da alteração, analisando-se o esquema atual e realizando-se as mudanças necessárias.

Porém, tem-se, a nível mercadológico, uma gama muito grande de bancos de dados implementados sem quaisquer documentações a respeito de suas características conceituais, pelo simples fato de tal documentação ter sido perdida, estar desatualizada ou nunca nem ter existido, devido a uma implementação feita sem a utilização da metodologia tradicional de projeto de banco de dados. Neste caso, para obter-se um projeto conceitual do BD físico, deve-se utilizar o processo de engenharia reversa.

A metodologia clássica de engenharia reversa é descrita em (Heuser, 2009). Este processo não é totalmente automático, pois está-se convertendo um modelo de baixo nível de abstração para um modelo com maior nível de abstração (no caso, o modelo ER). Neste caso, um usuário especialista deve dar apoio ao processo, completando de informações quando necessário, como por exemplo, identificando cardinalidades mínimas de relacionamentos.

O processo é dividido em quatro passos que são detalhados na sequência:

1. **Identificação dos construtores conceituais de cada tabela:**

Define-se, para cada tabela da modelagem relacional, qual elemento da modelagem conceitual ela representa.

Uma tabela pode vir a ser:

i. Um relacionamento de cardinalidade M..N;
ii. Uma hierarquia de especialização;
iii. Um atributo multivalorado;
iv. Uma entidade fraca;
v. Uma entidade forte.

O fator determinante na escolha do construtor conceitual provém da formação da chave primária que a tabela possui, esta podendo ser de três alternativas:

a. **Chave primária composta por mais de uma chave estrangeira**.

Neste caso, a tabela em si é a implementação de um relacionamento 0..N ou 1..N localizado entre as entidades as que possuem como chave primária as chaves estrangeiras presentes no relacionamento. As cardinalidades mínimas não podem ser obtidas automaticamente, devendo ser informadas pelo usuário especialista, ou identificada após uma análise dos dados existentes no BD. A *Figura 10* representa esta situação.

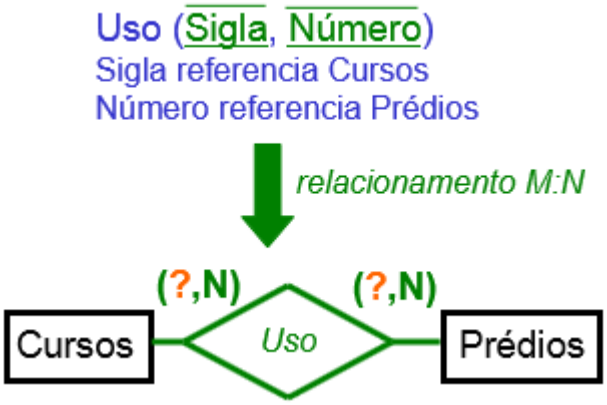

Figura 10: Chave primária composta por mais de uma chave estrangeira.

b. **Chave primária ser também uma chave estrangeira**.

Neste caso, a tabela representa uma especialização da entidade que possui como chave primária a chave estrangeira presente na tabela. O tipo da expecialização (i.e. total, parcial, exclusiva, compartilhada), deve ser informado pelo usuário especialista. A *Figura 11* representa a situação.

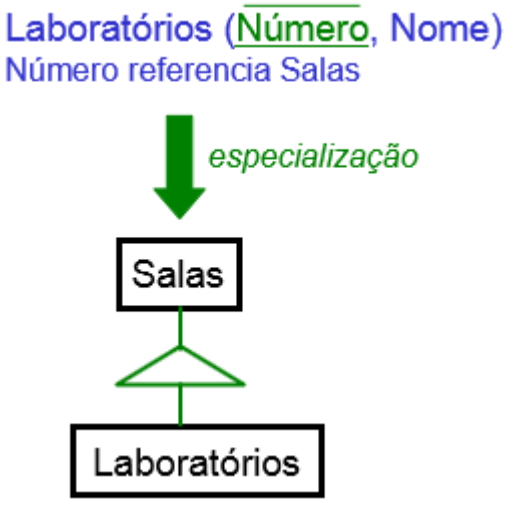

Figura 11: Chave primária ser uma chave estrangeira.

c. **Demais casos**.

Caso a tabela não possua como chave primária apenas chaves estrangeiras (uma ou mais), ela pode ser um construtor conceitual correspondente a:

i. Um atributo multivalorado: a tabela possui como atributos apenas a chave primária e uma chave estrangeira (da tabela possuidora deste atributo)(*Figura 12*);
ii. Uma entidade fraca: quando existe uma chave primária e pelo menos uma chave estrangeira na tabela (*Figura 13*);
iii. Hierarquia de especialização: a tabela deve possuir um atributo de qualificação de especialização, como exibido na *Figura 14*;
iv. Entidade forte: nos demais casos. (*Figura 15*)

Em todos estes casos, a intervenção do usuário é necessária para prover informações ausentes no nível lógico, como cardinalidades mínimas e tipos de especializações.

Figura 12: Atributo multivalorado.

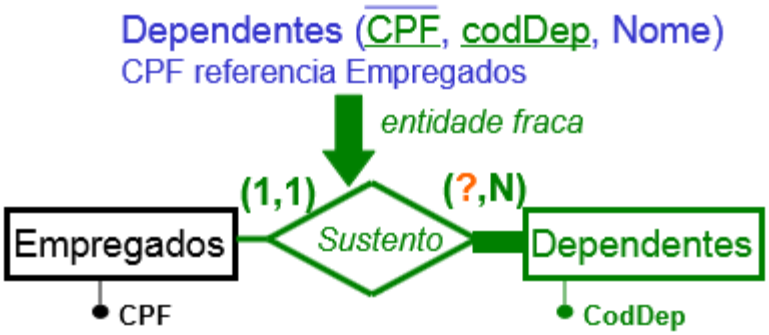

Figura 13: Entidade fraca.

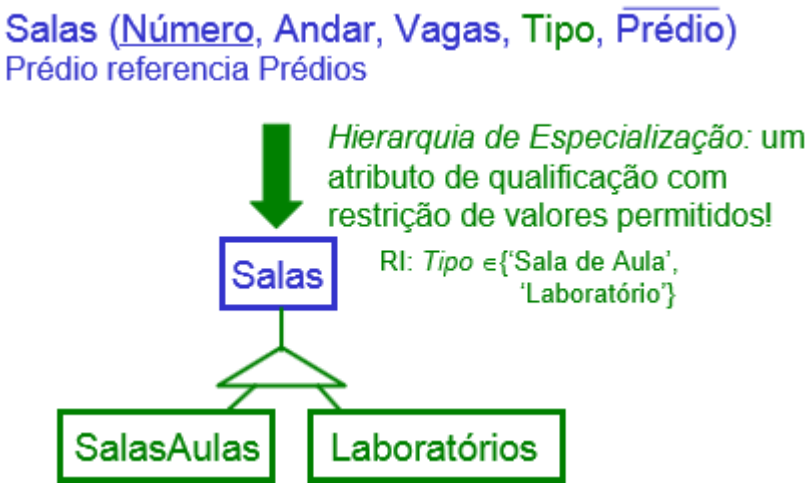

Figura 14: Hierarquia de especialização.

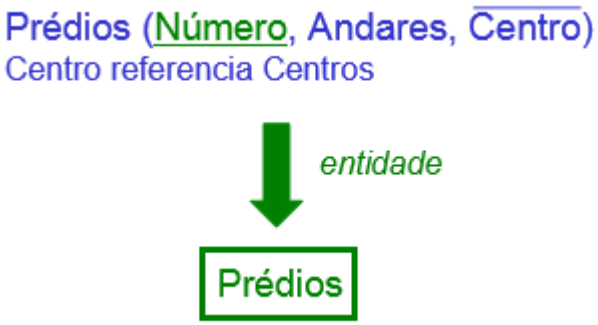

Figura 15: Entidade forte.

2. **Identificação de relacionamentos 1..N e 1..1:**

Neste passo, analisa-se as chaves estrangeiras que não se enquadraram no primeiro passo (i.e. chave estrangeira não é chave primária, não está contida nas chaves primárias nem está associada a casos de atributo multivalorado ou entidade fraca).

Este passo não pode ser facilmente automatizado, pois nele não há garantia de acerto na cardinalidade do relacionamento, o qual poderia ser 1..1 ou 1..N.

Duas são as alternativas para este passo:

a. A tabela representa um relacionamento binário 1..1 ou 1..N, conforme exibido na *Figura 16*;

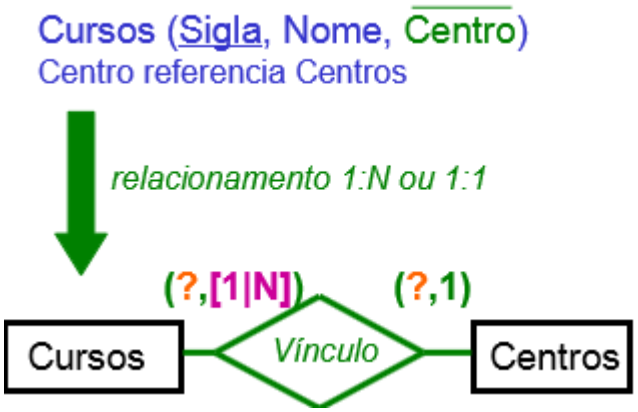

Figura 16: Relacionamento binário.

b. A tabela representa uma entidade associativa, no caso de a chave primária ser composta, conforme exibido na *Figura 17*.

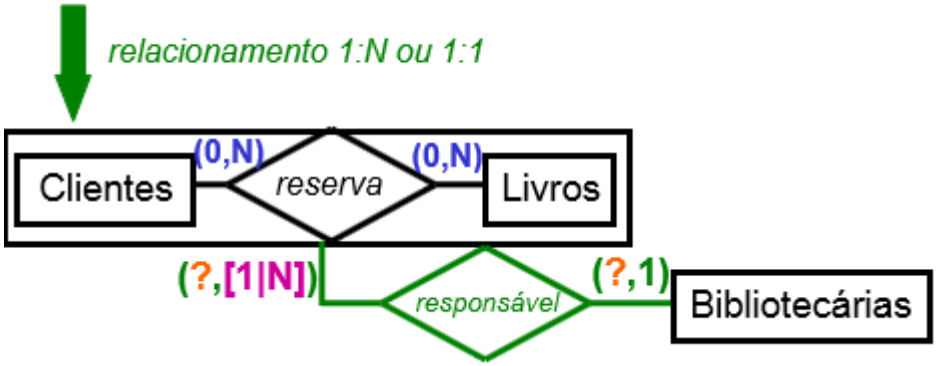

Figura 17: Entidade associativa.

3. Identificação de atributos:

Este passo analisa os atributos presentes nas tabelas que não são chaves estrangeiras nem chaves primárias.

Três alternativas são possíveis:

a. Atributo pertence a relacionamento (já identificado no passo 1). A *Figura 18* representa tal situação.

b. Atributo pertence a entidade especializada. Neste caso, deve-se analisar as entidades a fim de identificar se o atributo deverá pertencer à entidade pai ou à entidade filha. A *Figura 19* representa este caso.

c. Atributo pertence a entidade. Esta regra se aplica nos demais casos, e está representada na *Figura 20*.

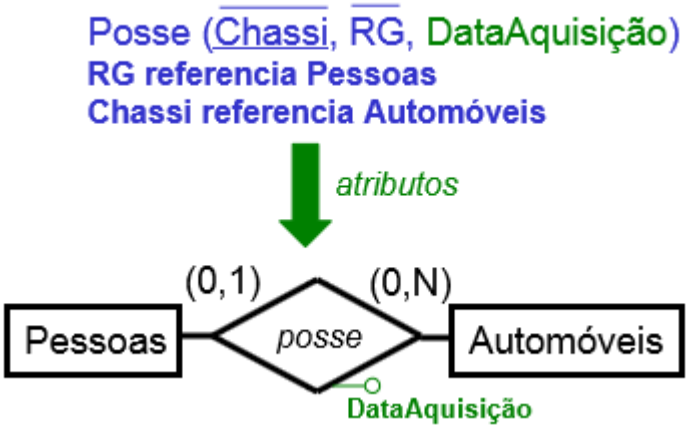

Figura 18: Atributo de relacionamento.

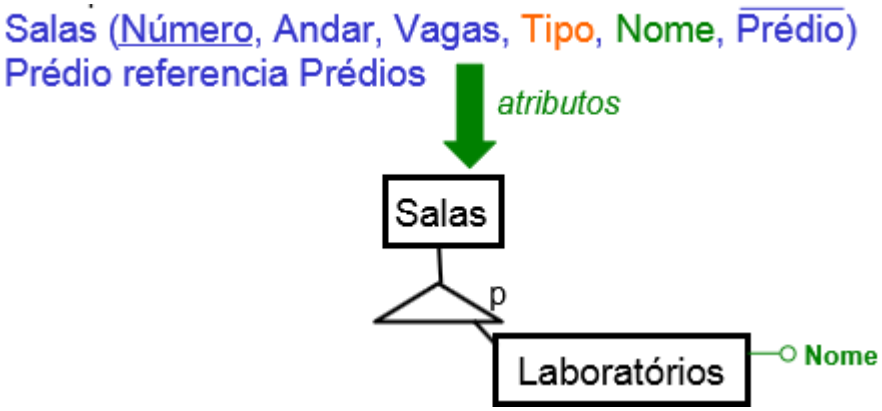

Figura 19: Atributo de entidade especializada.

Figura 20: Atributo de entidade.

4. Identificação de entidades e relacionamentos:

No derradeiro passo, identifica-se atributos pertencentes a chave primária que não são chave estrangeira, tornando-os identificadores de entidades ou relacionamentos, conforme exibido na *Figura 21* e na *Figura 22*.

Figura 21: Identificadores de entidades.

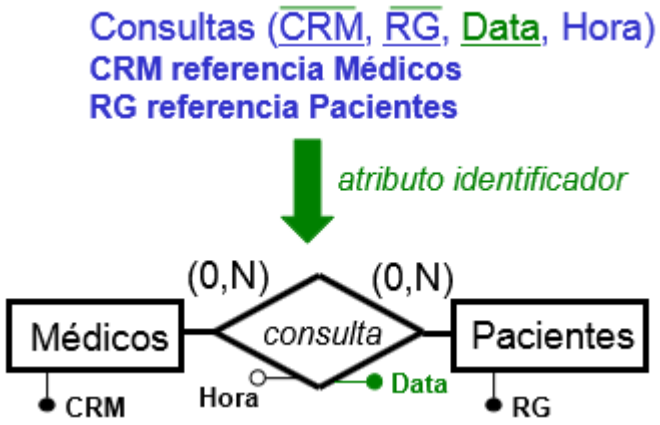

Figura 22: Identificadores de relacionamentos.

3 A Ferramenta brModeloNext

A ferramenta brModelo foi criada em 2005 pelo GBD/UFSC para servir de insumo no processo de aprendizagem de projetos de bancos de dados relacionais [BRMODELO, 2013]. No ano de 2011, desenvolveu-se a ferramenta brModeloNext "(...) visando sanar os erros existentes na versão atual [brModelo, 2013], além de revisar e melhorar a sua usabilidade com o auxílio de experimentos com usuários e também deixá-la independente de plataforma (...)" (Menna et. al, 2011).

Abrangendo os três passos da modelagem de esquemas relacionais, a ferramenta dá ênfase ao processo de criação de um esquema conceitual, visto que não existem ferramentas voltadas a essa prática no mercado. Desta forma, partindo-se da modelagem conceitual, e de uma forma semi automatizada (i.e. interagindo com o usuário em pontos críticos de tomada de decisão), a ferramenta faz a conversão do esquema para uma modelagem lógica e, em seguida, para uma modelagem física, gerando para o usuário, um script SQL/DDL ANSI.

Desde seu lançamento, a ferramenta tem sido amplamente utilizada no processo de aprendizado de projetos de bancos de dados. No entanto, ela não oferece ao usuário a possibilidade de realização do processo de engenharia reversa. Este é o fato que impulsionou o desenvolvimento deste trabalho.

A brModeloNext foi desenvolvida utilizando-se a plataforma Java, o que propicia uma execução independente de sistema operacional. Como componente gráfico para a diagramação dos esquemas, foi utilizada a biblioteca JGraph. Além disso, ela segue o paradigma de orientação a objetos e utiliza o padrão de projeto estrutural MVC.

A *Figura 23* demonstra a interface gráfica da ferramenta, realizando uma modelagem conceitual.

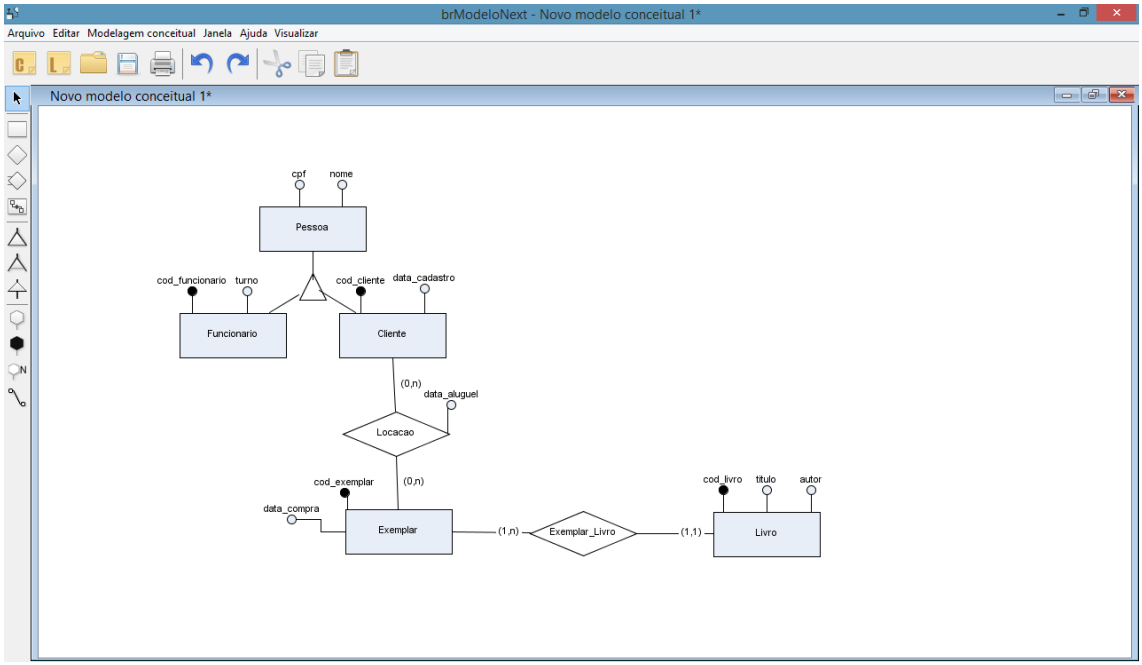

Figura 23: A ferramenta brModeloNext.

Sendo uma aplicação Java e sobre o paradigma de orientação a objetos, o código fonte da brModeloNext é composto por uma série de classes, as quais são subdivididas em pacotes para facilitar seu entendimento e sua organização. Originalmente, a ferramenta é composta por um pacote de classes chamado *App*, que armazena a classe inicializadora do sistema; um pacote chamado *Control*, responsável por manter as classes que realizam a lógica da aplicação, ou seja, o gerenciamento das modelagens e as conversões entre tais; o pacote *UI*, que contém os arquivos responsáveis pela interface com o usuário, o qual ainda é subdividido em *Resources* e *Images*, que contém separadamente textos e imagens, respectivamente; um pacote *Util*, que contém classes com funções pontuais e isoladas; e por fim, o pacote *Model*, que armazena, nos subpacotes *Objects* e *Shapes*, os objetos dos diagramas conceitual e lógico em tempo de execução. A *Figura 24* representa diagramaticamente esta arquitetura.

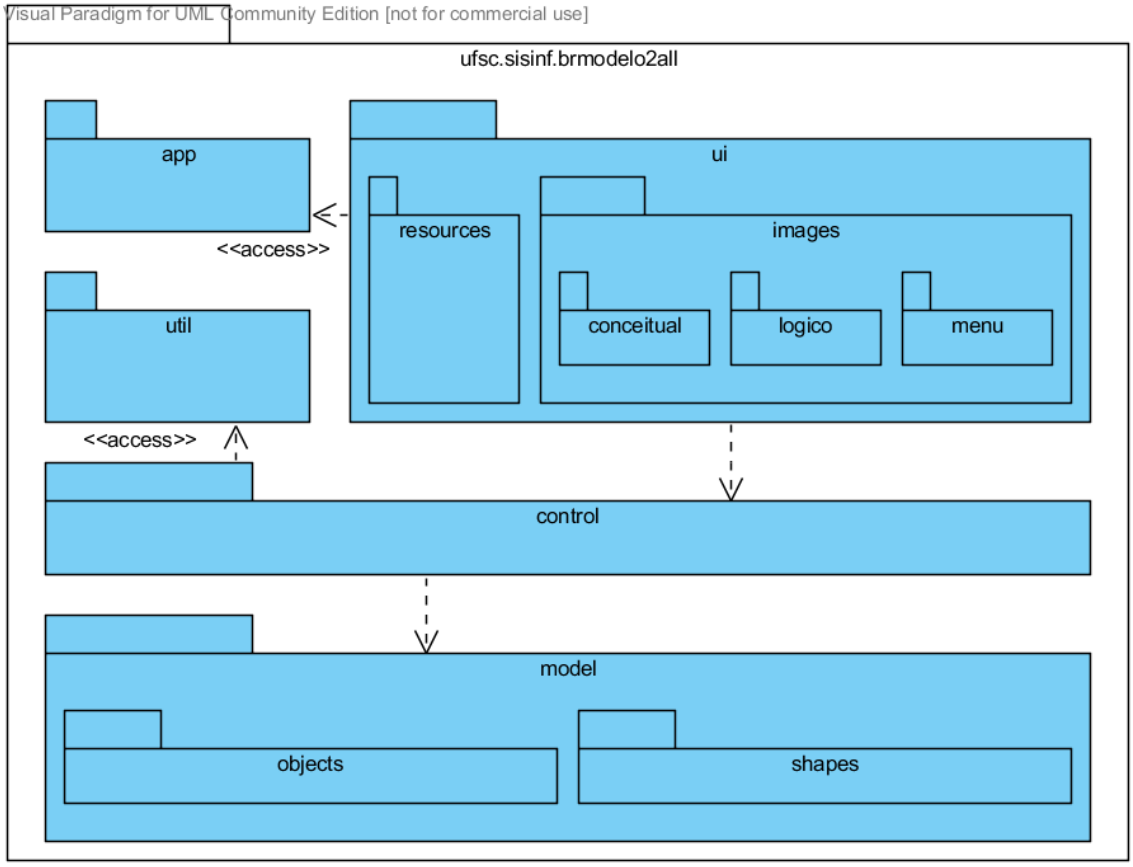

Figura 24: Arquitetura da ferramenta brModeloNext.

Dentro do pacote *Model*, estão as classes que representam, em tempo de execução, os objetos conceituais ou lógicos criados pelo usuário em seus diagramas. A principal classe deste pacote é a classe *ModelingObject*, que contém atributos comuns a todos os objetos. Dela, especializam-se as classes que dão origem a objetos conceituais e lógicos. Estes objetos estão localizados no subpacote *Object* e são eles: *EntityObject, RelationObject, InheritanceObject, AttributeObject, AssociativeEntityObject* e *AssociativeRelationObject*, constituindo os objetos relativos ao modelo conceitual, e: *TableObject* e *ColumnObject*, relativos ao modelo lógico. Também neste pacote estão presentes os objetos *ConnectorObject* e *Cardinality*, responsáveis por representar as ligações entre os objetos e suas cardinalidades. A *Figura 25* representa o diagrama de classes desta estrutura conceitual, enquanto a *Figura 26* representa as classes do modelo lógico.

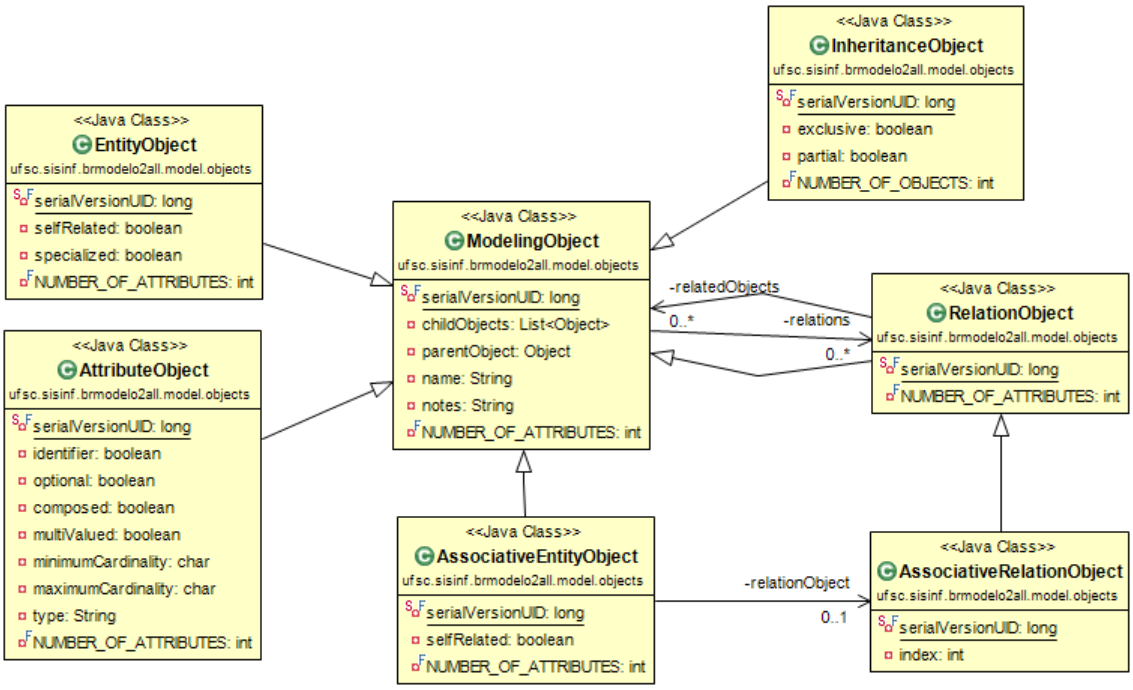

Figura 25: Diagrama de classes do modelo de dados (conceitual).

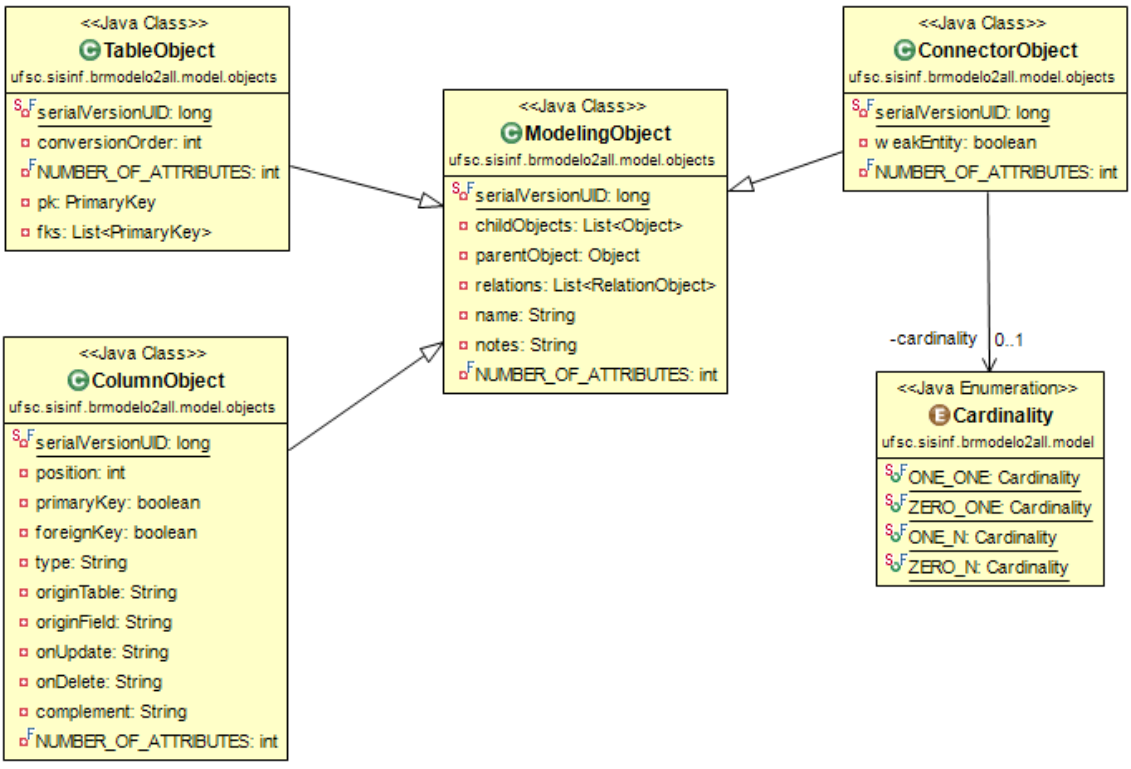

Figura 26: Diagrama de classes do modelo de dados (lógico).

No pacote de classes responsável pela lógica de negócio, os *controllers* são compostos por dois grupos de funções distintas, sendo que as classes *ModelingManager, ModelingEditor* e *ModelingComponent* são responsáveis por tratar as ações que o usuário faz ao interagir com o programa. Já as classes *ConceptualConversor, LogicalConversor* e *SqlEditor* são responsáveis pelas funções de conversão e tratamento das modelagens conceitual, lógica e física. A *Figura 27* representa tais classes.

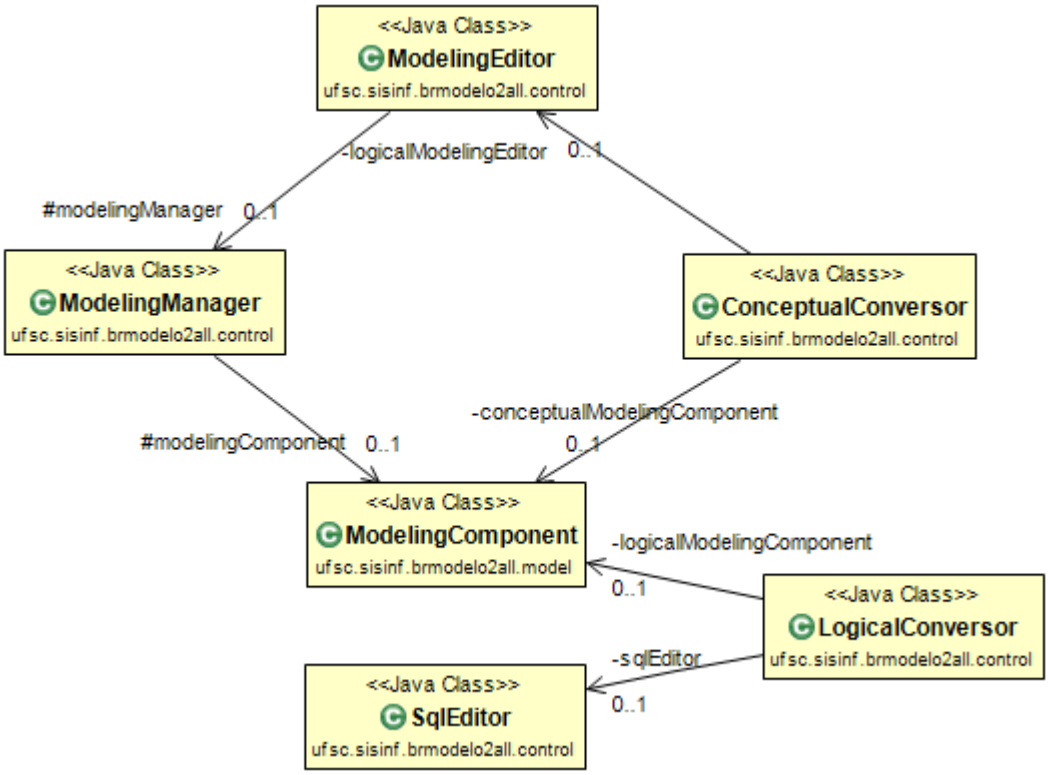

Figura 27: Diagrama de classes dos controllers.

4 Incorporação do processo de engenharia reversa

Pelo fato de o projeto aqui apresentado tratar-se da implantação de uma nova funcionalidade a um programa já existente, alguns pontos devem ser observados.

i. Seguiu-se a metodologia já empregada no desenvolvimento inicial da ferramenta, procurando-se sempre a utilização de padrões já existentes no código fonte da aplicação.
ii. Utilizou-se o conjunto de classes já existentes para a representação (a nível computacional) dos objetos a serem modelados na ferramenta.
iii. Utilizou-se como modelo de entrada de dados para o processo de engenharia reversa, uma modelagem lógica da ferramenta brModeloNext, o que pode ser construído pela própria ferramenta, ou através de um carregamento de um arquivo já salvo.

Ao dar-se início ao processo de desenvolvimento da extensão, teve-se como passo inicial a incorporação de um botão no menu superior da ferramenta, o qual pudesse disparar o processo de engenharia reversa. Tal implantação deu-se através de mudanças nas classes *MenuBar* e *CommandActions*, presentes no pacote de classes *UI*.

Com o intuito já declarado de seguir os mesmos padrões presentes no código fonte da ferramenta, foi criada, no pacote *Control*, a classe *ReverseEngineeringConversor*, para fins de realizar os passos do algoritmo de engenharia reversa.

O ponto de partida para o algoritmo proposto é uma modelagem lógica. Tendo isto apresentado na tela, o sistema identifica as tabelas e cria, para cada, objetos, que são entidades das classes já existentes no software, as quais vêm sendo utilizadas no processo de modelagem lógica. Essas classes compreendem o modelo de dados utilizado no sistema, e podem ser encontradas no pacote *ufsc.sisinf.brmodelo2all.model.objects*, juntamente com as classes que representam os objetos criados durante a modelagem conceitual.

Durante a geração do esquema conceitual, no processo de engenharia reversa, foram utilizadas as mesmas classes que o sistema já usava para fazer a modelagem conceitual. Ao longo do desenvolvimento, entretanto, percebeu-se que a ferramenta brModeloNext não tratava de forma adequada chaves primárias e chaves estrangeiras compostas. Incluiu-se então uma nova classe no pacote *Model*, chamada *PrimaryKey*. Esta nova classe nada mais é do que um encapsulamento de uma lista de objetos *ColumnObject* (que representam as colunas de uma tabela do banco de dados). Foi refatorado o sistema então para que a classe que representa as tabelas (*TableObject*) possua um atributo *PrimaryKey* e um atributo chamado *foreignKeys*, o qual consiste em uma lista de *PrimaryKeys*, cada uma herdada de uma tabela com quem essa tem ligação. Na *Figura 28*, ao ser comparada com a *Figura 26*, pode-se observar a presença da nova classe.

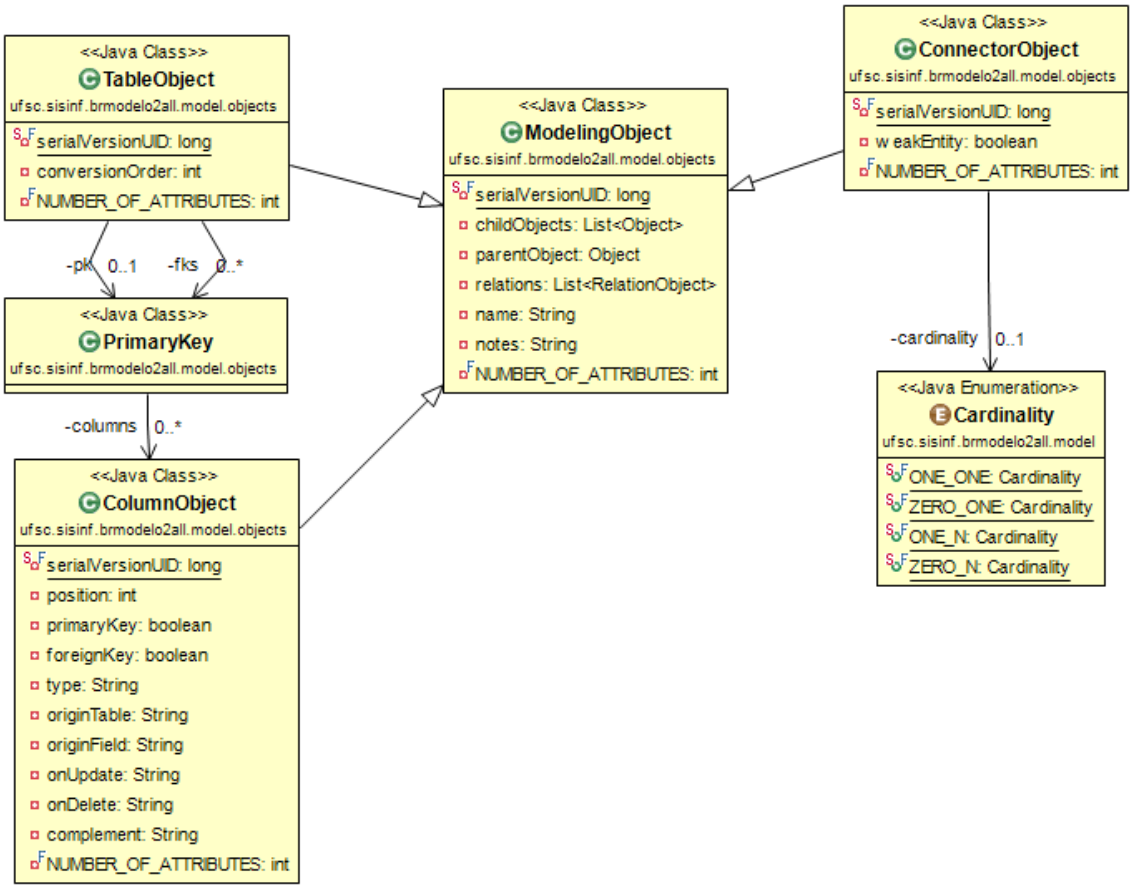

Figura 28: Diagrama de classes do modelo de dados (lógico) após as modificações.

Ao longo do desenvolvimento dos passos do algoritmo proposto por Heuser, o grau de complexidade da classe responsável pela engenharia reversa foi crescendo consideravelmente, necessitando-se assim que ela fosse repartida em diversas classes menores para facilitar o entendimento do código. Esta criação de novas classes fez com que também fosse interessante a criação de um novo pacote destinado apenas às classes responsáveis pela nova funcionalidade da ferramenta. Foi criado então o pacote *Reverse*, dentro do pacote *Control*. Esta alteração na arquitetura definiu o estado final dos pacotes da ferramenta, de forma que a *Figura 29* apresenta a sua versão final. Em comparativo com a *Figura 24*, pode-se perceber nessa a presença do novo pacote *Reverse*.

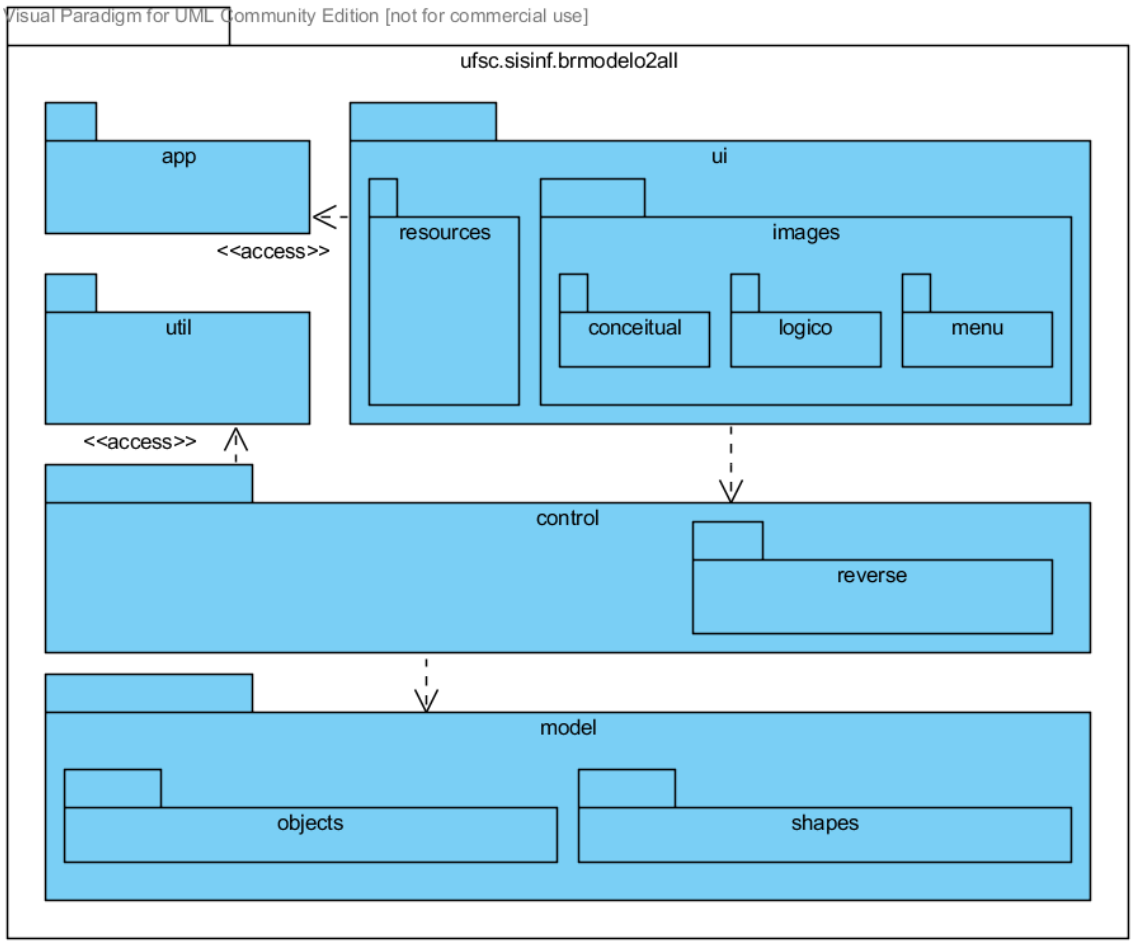

Figura 29: Arquitetura da ferramenta brModeloNext versão final.

Pelo fato de a ferramenta ser desenvolvida na linguagem Java, a qual segue o paradigma de orientação a objetos, procurou-se utilizar ao máximo os conceitos deste paradigma, tais como *herança*, *polimorfismo* e desacoplamento de código. Desta forma, as classes que foram criadas ao refatorar a classe *ReverseEngineeringConversor,* tornaram-se pontuais em suas funções, o que fez com que o pacote *Reverse,* contesse um grande número de classes. Um diagrama conciso destas classes pode ser visto na *Figura 30*. A classe anteriormente chamada de *ReverseEngineeringConversor* foi renomeada para *ReverseEngineeringConversorWizard*, a fim de seguir a nomeclatura adequada ao padrão de projetos utilizado.

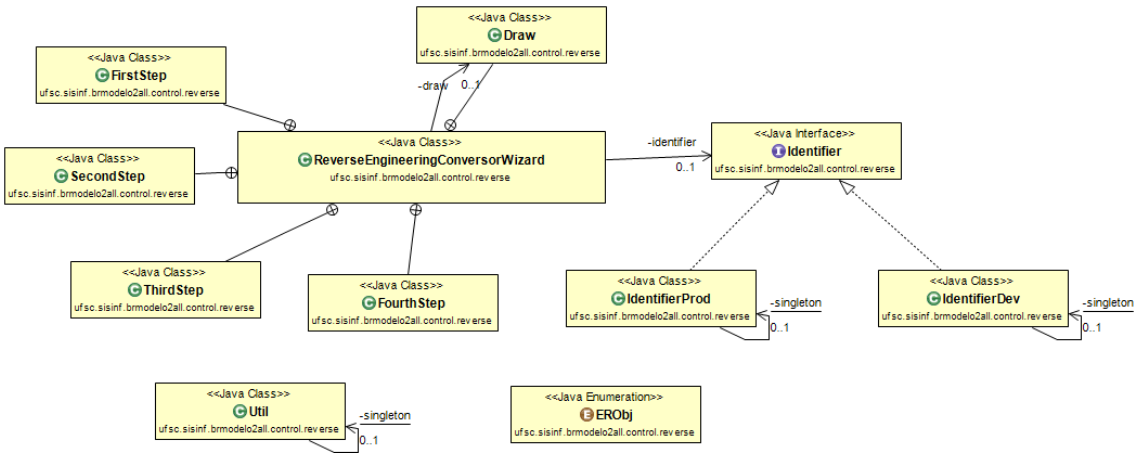

Figura 30: Diagrama de classes da engenharia reversa.

Estão presentes no pacote, quatro classes responsáveis pelos quatro passos do algoritmo de engenharia reversa, respectivamente, *FirstStep, SecondStep, ThirdStep e FourthStep*. Também foi criada uma classe responsável por apenas analizar as tabelas e conter as regras de identificação dos objetos lógicos. Por fim, foram criadas classes com responsabilidades auxiliares, como a *Util* e a *Draw*, responsável pela comunicação com o usuário. A *Figura 31* representa as classes do pacote *Control* depois da criação da classe responsável pelo processo de engenharia reversa.

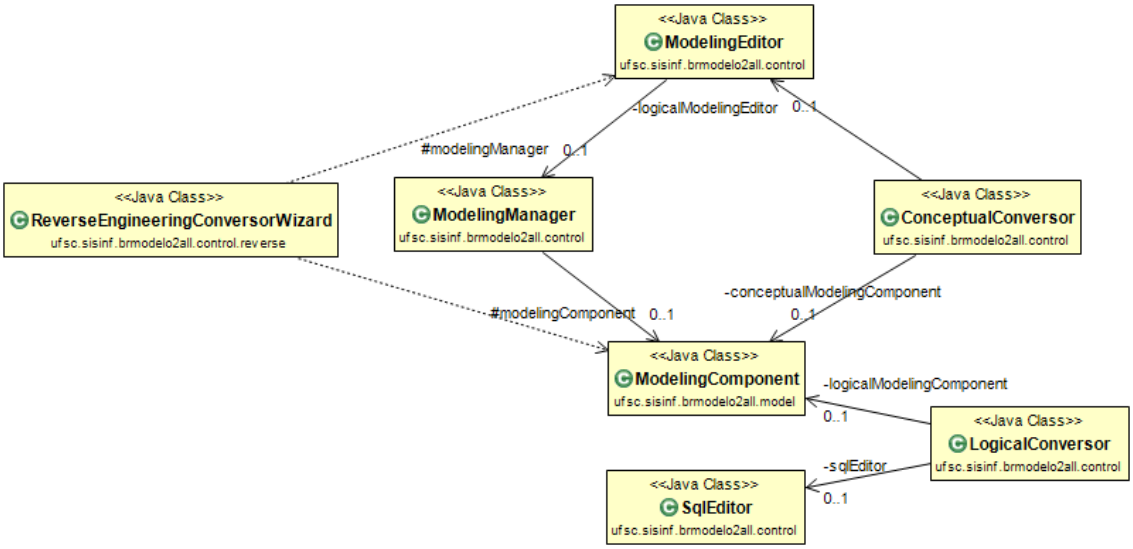

Figura 31: Diagrama de classe dos controllers após as modificações.

Tendo-se já discutido sobre a importância dos padrões de projeto para o desenvolvimento de software, aqui reforçamos estes argumentos com a utilização do padrão de projetos *Wizard*. Esta forma de programação inspira-se nos chamados Assistentes de instalação e configuração de softwares (ou, wizards). Tais assistentes são úteis quando necessita-se de uma sequência de execução para realizar alguma tarefa. Por o processo de engenharia reversa consistir em uma sequência de passos linear, optou-se por utilizar este padrão de projetos no código fonte da aplicação. Além disso, este padrão deixa o código limpo e legível, facilitando o entendimento do que está sendo feito e auxiliando assim as futuras manutenções do sistema. Um exemplo de código final utilizando-se este padrão pode ser visto na *Figura 32*.

```
ReverseEngineeringConversorWizard wizard = new ReverseEngineeringConversorWizard(
        modelingComponent, newModelingEditor, mainWindow);
wizard.startConverting().
    identifyTables().
    createConceptualObjects().
    createNewRelations().
    andDrawItOnScreen();
```

Figura 32: O padrão de projetos Wizard.

O *Wizard* faz uso de várias classes representando os passos onde encontra-se a execução do programa. A *Figura 33* demonstra as classes intermediárias criadas para realizar o processo de engenharia reversa seguindo o padrão utilizado.

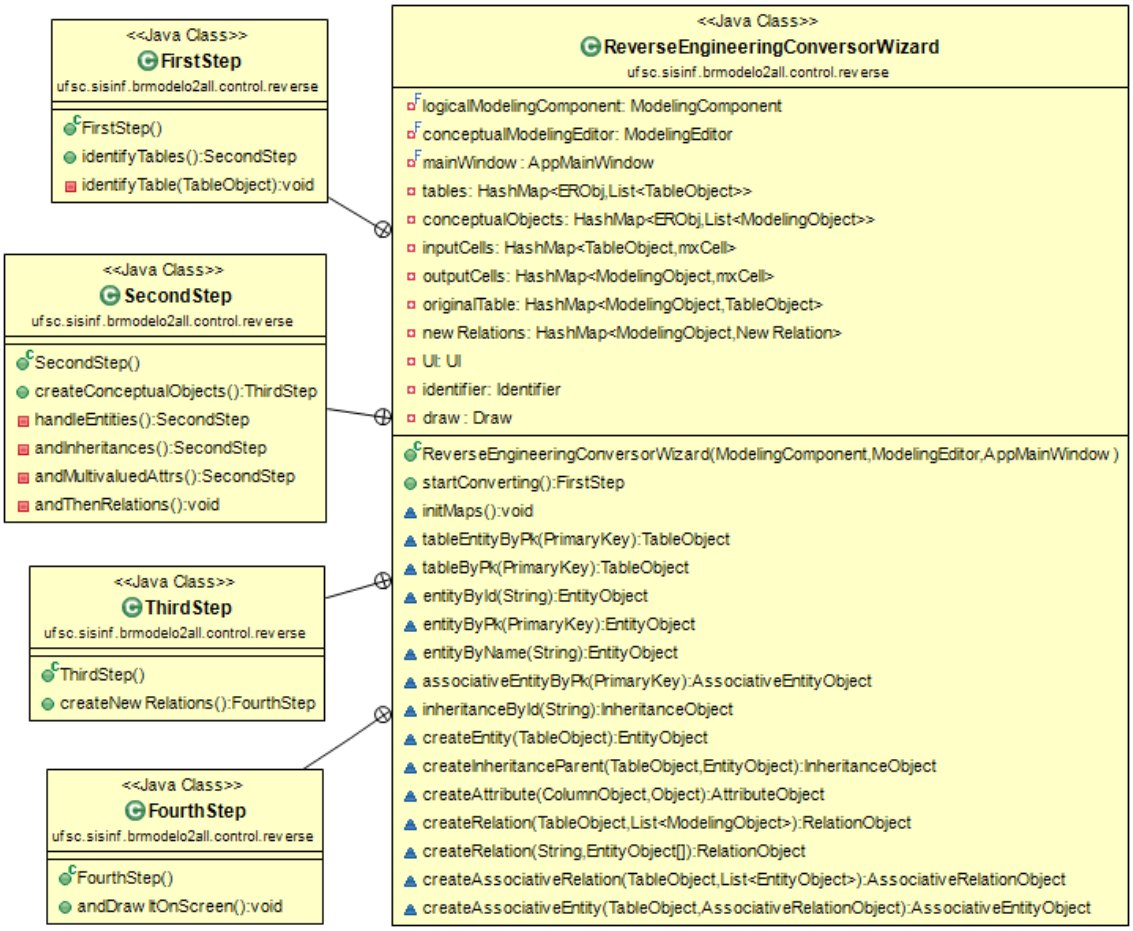

Figura 33: Diagrama de classes de engenharia reversa.

A principal classe responsável pela conversão do esquema lógico para o conceitual segue os passos descritos na metodologia de (Heuser, 2011). Ao receber um evento de clique no botão que indica o processo de engenharia reversa, começa-se o fluxo de conversão.

O processo inicia-se com uma análise das tabelas do modelo lógico recém criado, utilizando as regras descritas no capítulo 2.2 para classificá-las entre Entidades fortes ou Hierarquias de especialização, Relacionamentos, Entidades fracas, Atributos multivalorados e Entidades especializadas.

Pelo fato de que Entidades fortes e Hierarquias de especialização terem as mesmas características relativas às suas chaves primárias, sua diferenciação não pode ser automatizada por meio de algoritmos. Utilizou-se então a interação com o usuário para que este informe a origem da tabela analisada. A *Figura 34* mostra a janela de interação com o usuário para adquirir a informação necessária. No entanto, levou-se em consideração que o usuário possa não conhecer a base de dados que está sendo convertida. Desta forma, esta interação apenas funciona como uma orientação a ser seguida. Ainda assim, ao longo da análise são feitas outras verificações. Por exemplo, caso o usuário diga que a entidade X é uma entidade forte e não uma hierarquia de especialização, mas, ao longo da análise, identifica-se uma entidade Y filha da entidade X, X é promovida ao status de hierarquia. Também vale lembrar que, enquanto são realizados os passos da engenharia reversa, vai sendo exibido para o usuário, em segundo plano, o que já foi feito, de forma que ele pode basear-se na modelagem lógica e no que já foi identificado da modelagem conceitual para responder as perguntas que forem feitas. Este resultado parcial é exemplificado no capítulo referente ao Estudo de caso.

A identificação de Entidades especializadas, Entidades fracas, Atributos multivalorados e Relacionamentos é feita de forma automática, pois suas identificações são pontuais e estão contidas nas regras de identificação das chaves primárias.

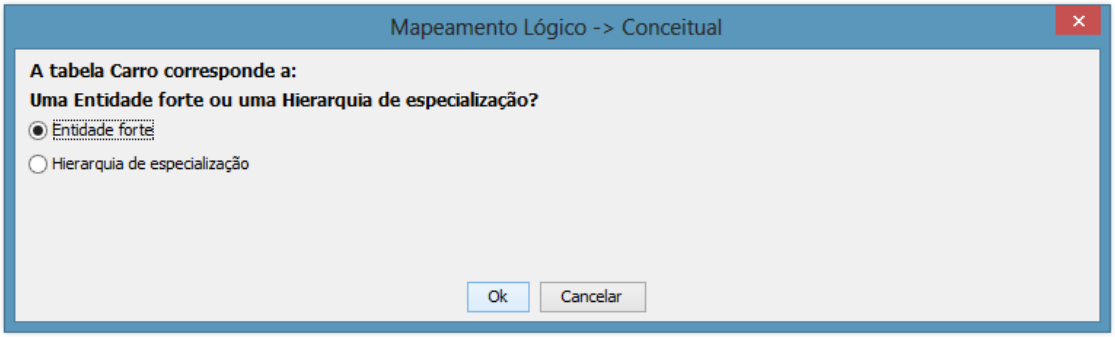

Figura 34: Identificação de Entidades.

Uma vez feita a identificação de Entidades e Relacionamentos, faz-se uma nova varredura nas tabelas encontradas, a fim de poder identificar os relacionamentos que serão promovidos a Entidades Associativas, caso houver.

Ao fim deste primeiro processo, já tem-se, armazenado em memória, todas as tabelas presentes no esquema já devidamente identificadas quanto a sua origem.

Esta primeira parte do processamento configura-se a mais complexa e demorada, pois nela são identificados os objetos que serão a essência da modelagem a ser criada. Os passos seguintes, apesar de também possuirem interação com o usuário e processamentos avançados, baseiam-se nas entidades e relacionamentos identificados anteriormente, o que reforça o fato de que a primeira etapa deve ser concluída sem nenhuma incerteza.

A segunda parte do algoritmo de engenharia reversa consiste na identificação das cardinalidades presentes entre os relacionamentos e as entidades. Novamente, não há método preciso para identificar tal coisa. Para tanto, o usuário é requisitado novamente, seguindo o mesmo padrão de interação presente no sistema até então. Neste caso, mesmo o usuário não conhecendo o banco de dados, a resposta não interfere nas etapas seguintes do processo. Para esta interação com o usuário, ao perguntar se a cardinalidade do relacionamento é 0..N ou 1..N, sugere-se sempre 0..N, lembrando-se que, ao término da conversão, o usuário poderá alterar estas cardinalidades, bem como qualquer objeto da modelagem gerada.

O processamento se encerra com a identificação dos atributos que não compõem a chave primária nem as chaves estrangeiras da tabela (terceiro passo); e com a determinação das chaves primárias de cada entidade do esquema. Com isso, tem-se finalizado o processo de identificação e montagem do esquema conceitual.

Como derradeira ação, a ferramenta exibe na tela, de maneira formal e precisa, e seguindo as normas do modelo Entidade-Relacionamento, todos os objetos identificados, juntamente com suas características, permitindo que o usuário dê sequência à seu trabalho em cima do esquema do banco de dados,

agora sendo exibido no modelo conceitual, facilitando o entendimento e a manutenção futura do esquema.

5 Estudo de caso

A fim de exemplificar a solução e demonstrar o processo com a ferramenta em funcionamento, foi criado um esquema lógico sobre o qual foi executado o processo de engenharia reversa. A modelagem pode ser vista na *Figura 35*. Ela simula um banco de dados que armazena dados sobre revendedoras de carros. O esquema conta com uma tabela genérica *Pessoa* e duas especializações para esta: *Cliente* e *Funcionario*. Possui a tabela *Revendedora*, que realiza *Transações* de *Carros*. Cada carro possui informações a respeito de seus *Acessórios* e de seu *Modelo*. Cada modelo possui também uma referência ao modelo de carro predecessor a este. O esquema também possui as tabelas *Compra*, *Funcionario_Revendedora* e *Carro_Modelo*, que fazem a ligação entre tabelas com cardinalidades múltiplas.

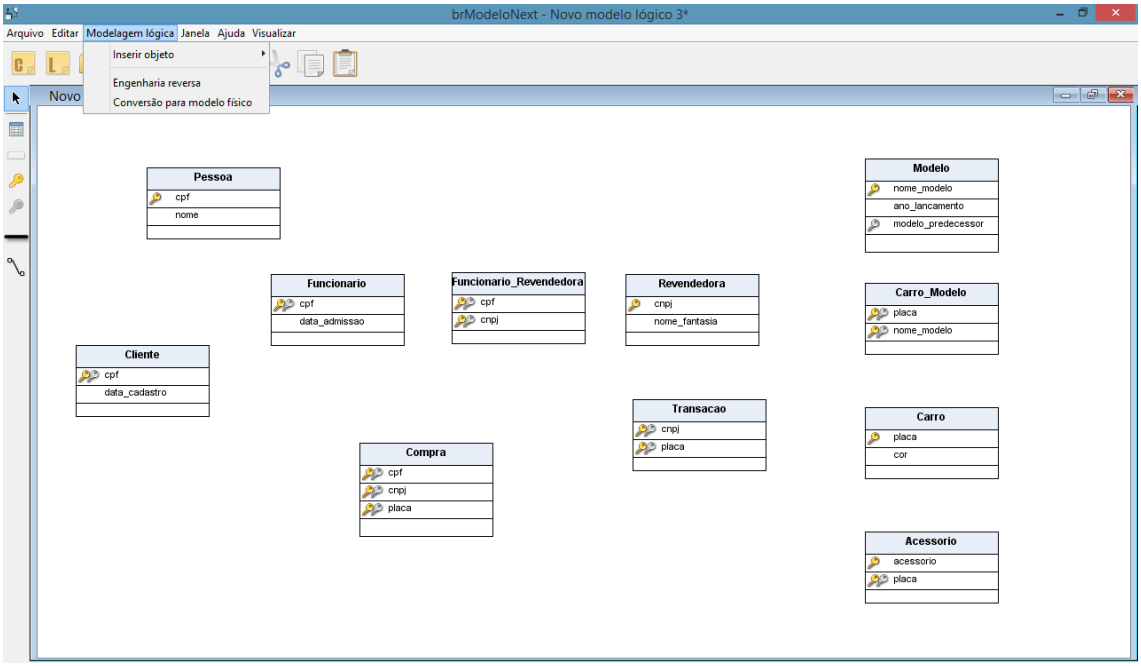

Figura 35: Entrada de dados para Engenharia Reversa.

Ao dar entrada no programa, o esquema é carregado e exibido ao usuário em uma modelagem lógica. O usuário então dá início ao processo de engenharia reversa através da seleção da respectiva opção no menu superior, como demonstrado na *Figura 35*.

Aplicando as regras já apresentadas, o sistema começa identificando o construtor conceitual de cada tabela. A ferramenta então interage com o

usuário ao encontrar uma situação a qual ela não consegue identificar automaticamente. No exemplo a seguir, a primeira interação ocorre para adquirir informações sobre se determinada tabela é uma entidade forte ou uma hierarquia de especialização. Este exemplo de interação pode ser visto na *Figura 36*. No exemplo, esta pergunta é feita para as tabelas *Pessoa*, *Revendedora*, *Carro* e *Modelo*.

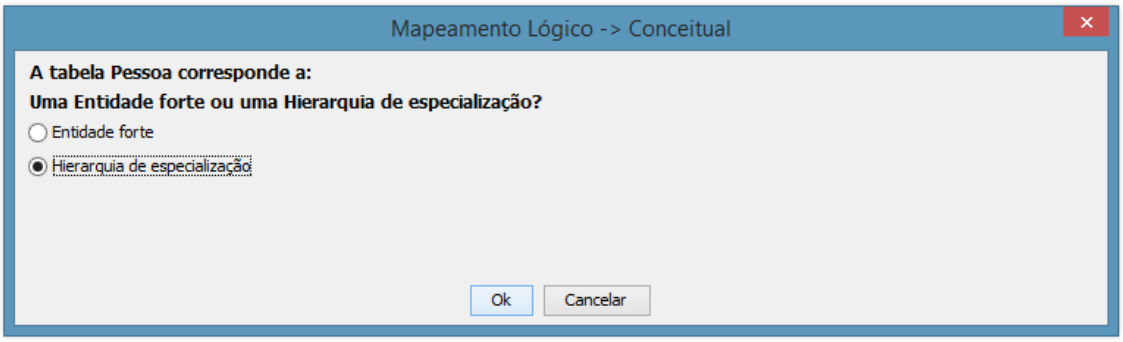

Figura 36: Interação com o usuário para obter informações sobre entidade.

Após este passo, a ferramenta já exibe na tela as entidades identificadas. A *Figura 37* mostra o resultado parcial após esta etapa.

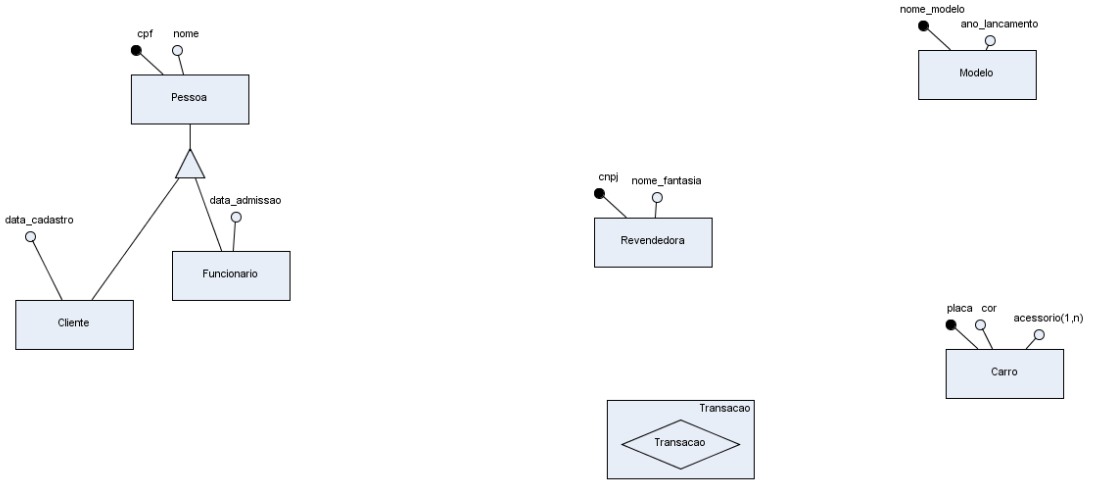

Figura 37: Resultado parcial após o primeiro passo.

Dando continuidade ao processo, o sistema pede que o usuário informe as cardinalidades dos relacionamentos identificados. A interação se dá através de uma janela semelhante à apresentada na *Figura 38* . Esta pergunta é feita para todas as associações que possam ser 0..N ou 1..N.

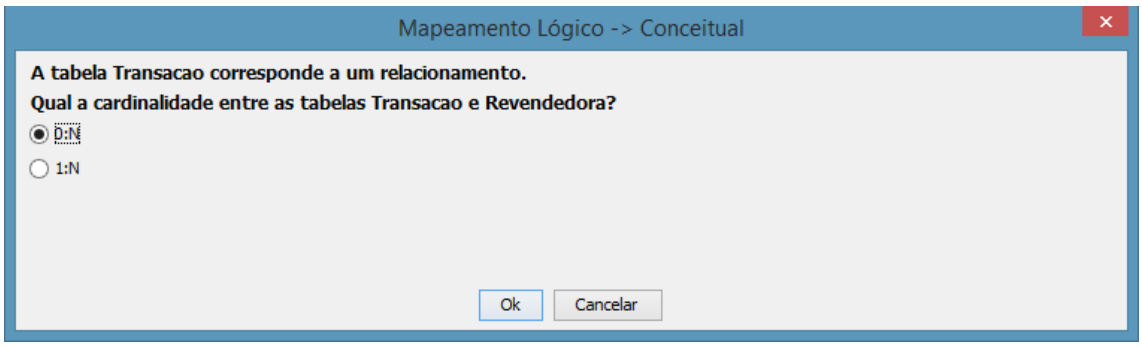

Figura 38: Interação com o usuário para obter informações sobre cardinalidades.

Ao final da análise, o processamento resultou na modelagem conceitual apresentada na *Figura 39*. Pode-se perceber que a tabela *Pessoa* transformou-se em uma hierarquia, especializada nas entidades *Cliente* e *Funcionário*. A tabela *Transacao* foi identificada como uma entidade associativa, e a tabela *Acessorio* como um atributo multivalorado. Ainda, foi identificado um auto-relacionamento na tabela *Modelo*. Assim, fica exibido ao usuário o esquema conceitual referente à modelagem lógica inserida na ferramenta.

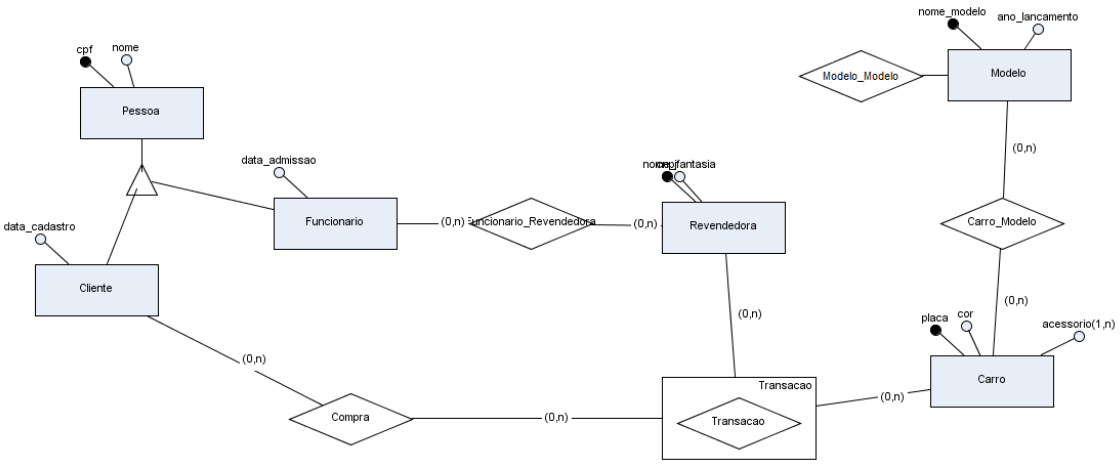

Figura 39: Resultado da Engenharia Reversa.

6 Conclusão e trabalhos futuros

A modelagem conceitual apresenta-se na literatura como o primeiro passo para a elaboração de um esquema de banco de dados relacional. Isso dá-se pelo fato de ser um modelo formal simples e preciso, contendo todas as características essenciais para o entendimento da modelagem de dados representada pelo esquema. Suas distinções removem a complexidade do trabalho a ser realizado e fornecem ao projetista do banco de dados uma visão ampla, indiferente às tecnologias empregadas nos passos seguintes da modelagem.

A ferramenta brModelo já está no mercado há um longo período, suficiente para ter-se arraigado principalmente no meio acadêmico, onde a procura pelos resultados por ela obtidos são profundos e constantes. Para sua próxima versão, chamada brModeloNext, desenvolveram-se melhorias gráficas e no que diz respeito a portabilidade. Além disso, para agregar novas funcionalidades à ferramenta, desenvolveu-se um novo módulo para realizar o processo de engenharia reversa de esquemas lógicos bancos de dados relacionais. Esta é a contribuição deste trabalho de conclusão de curso.

Assim como a modelagem conceitual, foco da brModeloNext, o processo de engenharia reversa é pouco abordado pelas ferramentas presentes no mercado, as quais dão mais ênfase à modelagem lógica, e seguindo o modelo *top-down*. Esta nova funcionalidade vem engrandecer a ferramenta e trazer uma oportunidade para que os usuários possam ver seus esquemas de bancos de dados de forma conceitual, proporcionando um melhor conforto no processo de projeto e manutenção das bases de dados.

Atualmente, para a realização do processo de engenharia reversa, a ferramenta brModeloNext necessita que o usuário dê entrada no sistema com uma modelagem lógica já pronta. Sugere-se como trabalho futuro a ser desenvolvido, que o usuário possa realizar uma conexão direta com o SGBD que está gerenciando a base de dados e extrair assim as informações relativas ao modelo físico do banco de dados. Desta forma, a extração de dados do SGBD e a inclusão destes dados na brModeloNext seriam realizadas em apenas um passo, reduzindo uma complexidade do sistema. Este trabalho já

vem sendo feito e é o foco de outro Trabalho de Conclusão de Curso no âmbito da ferramenta brModeloNext.

Outra sugestão é a criação de uma ferramenta web que realize as funções da brModeloNext, propiciando a usuários do mundo inteiro a utilização de suas funcionalidades de maneira ainda mais simples e direta. Isso implicaria também na internacionalização da ferramenta. Este trabalho também já está em andamento em outro TCC.

Referências

HEUSER, Carlos A. **Projeto de Banco de Dados**. 6ª edição. Porto Alegre: Bookman, 2009

DATE, C. J. **Introdução a Sistemas de Bancos de Dados**. 8ª edição. Rio de Janeiro: Elsevier, 2004

MENNA, O. S. ; RAMOS, L. A. ; MELLO, R. S. **brModeloNext: a Nova Versão de uma Ferramenta para Modelagem de Bancos de Dados Relacionais**. In: Sessão de Demos - XXVI Simpósio Brasileiro de Banco de Dados, 2011, Florianópolis, SC. Anais da VI Sessão de Demos - XXVI SBBD, 2011.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 6ª edição. São Paulo: Pearson Addison Wesley, 2011

COUGO, Paulo S. **Modelagem conceitual e projeto de bancos de dados**. Rio de Janeiro: Elsevier, 1997

MELLO, Ronaldo dos Santos, **Apostilas aulas de Projeto de Banco de Dados**, ministradas na Universidade Federal de Santa Catarina, 2010

BRMODELO, **brModelo 2.0**. Disponível em <http://sis4.com/brModelo/>. Acessado em julho de 2013.

PIERIN, Felipe, **Padrões de Projetos, um estudo sobre os 23 padrões GoF**. Java Magazine. Dezembro de 2011; Edição 13.