UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

SAMUEL NICOLAU BROGNOLI

SISTEMA DE APOIO A ENGENHARIA REVERSA DE BANCOS DE DADOS RELACIONAIS POR MEIO DA EXTRAÇÃO DE METADADOS

CRICIÚMA, JULHO DE 2008

SAMUEL NICOLAU BROGNOLI

SISTEMA DE APOIO A ENGENHARIA REVERSA DE BANCOS DE DADOS RELACIONAIS POR MEIO DA EXTRAÇÃO DE METADADOS

Trabalho de Conclusão de Curso apresentado para a obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. MSc. Daniel Pezzi da Cunha

Co-Orientador: Prof. Fabrício Giordani

CRICIÚMA, JULHO DE 2008

SAMUEL NICOLAU BROGNOLI

SISTEMA DE APOIO A ENGENHARIA REVERSA DE BANCOS DE DADOS RELACIONAIS POR MEIO DA EXTRAÇÃO DE METADADOS

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

_____

Profa. MSc. Ana Claudia Garcia Barbosa

Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

_____

Prof. MSc. Daniel Pezzi da Cunha (UNESC)

Orientador

_____

Prof. Fabrício Giordani (UNESC)

Co-Orientador

_____

Prof. MSc. Ana Claudia Garcia Barbosa (UNESC)

_____

Prof. MSc. Cristiane Raquel Woszezenki (UNESC)

Dedico este trabalho à meus pais: Aldo e Noêmia, que me educaram e me proporcionaram a oportunidade de realizar este sonho.

AGRADECIMENTOS

Agradeço a Deus, por mais essa conquista e pela força necessária para superar todos os obstáculos que encontrei pelo caminho.

A meus pais Aldo e Noêmia, que me educaram e sempre me deram apoio em todos os momentos difíceis da minha vida.

Ao meu Orientador Daniel Pezzi, pelo seu apoio e dedicação durante todo o desenvolvimento do trabalho de conclusão.

Ao Co-Orientador Fabrício Giordani, que mesmo na correria do dia-a-dia, esteve sempre disposto a compartilhar seus conhecimentos e sua experiência.

A todos os colegas do curso, pelas alegrias e dificuldades compartilhadas em sala de aula ao longo do curso.

A Useall Software, empresa na qual trabalho, pela compreensão e apoio quando precisei me ausentar das atividades da empresa, para a realização do projeto de conclusão.

Aos amigos Tiago Camargo, Tiago Giusti, Marlon de Oliveira e Paula Machado, que me auxiliaram nas correções desse trabalho.

A todos que não mencionei, mas que de certa forma me ajudaram direta ou indiretamente na concretização deste projeto. Obrigado!

RESUMO

A utilização de sistemas de bancos de dados vem se tornando cada vez mais comum no cenário atual, passando a fazer parte do cotidiano de milhares de empresas. Nesse ambiente, a exigência de se aprimorar o gerenciamento de dados tem se tornado cada vez maior. Para isso, é necessário compreender o esquema conceitual, objetivando identificar e corrigir possíveis erros. No entanto, um problema muito comum, é a inexistência de documentação no desenvolvimento de projetos de bancos de dados, principalmente quanto à modelagem semântica. Objetivando solucionar o problema, são aplicados métodos de engenharia reversa capazes de reconstruir o modelo lógico e conceitual de bancos de dados já implementados, a fim de facilitar o processo de manutenção desses ou então a migração dos seus dados para uma nova aplicação. Este trabalho culminou no desenvolvimento de uma ferramenta experimental, de código fonte aberto, como solução para auxiliar o processo de engenharia reversa em bancos de dados relacionais. Esta, limitada aos SGBD Oracle e Firebird, examina os metadados para a extração do diagrama E-R. Desta forma, possibilita que administradores de banco de dados consigam, por meio do esquema conceitual, aprimorarem seus sistemas.

Palavras Chave: Bancos de Dados Relacionais, Engenharia Reversa, Modelo Entidade-Relacionamento.

ABSTRACT

The database applications are becoming more and more common nowadays, taking part of the thousands companies daily. In this environment, the requirement of an improved database management becomes higher and higher. Thereunto, it's necessary to understand the conceptual schema, objectifying to identify and correct possible mistakes. Although, an usual problem is the documentation absence in the database project development, mainly about semantic modeling. Objectifying to solve this problem, it is applied reverse engineer methods that can rebuild the logic and conceptual database model, in order to make easier the maintenance process or, so, its data migration to a new application. This work has finished with a development of a experimental tool, open source, as a solution to assist the reverse engineer process in relational database. Restricted to Oracle and Firebird DBMS, it checks the metadata to extract the E-R diagram. In this way, it makes capable that the database administrator can improve their systems through a conceptual schema.

Key-Words: Relational Database, Reverse Engineer, Entity-Relationship Model.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANSI	*American National Standards Institute*
AWT	*Abstract Window Toolkit*
CASE	*Computer Aided Software Engineering*
E-R	Entidade-Relacionamento
E-RE	Entidade-Relacionamento Estendido
GEF	*Graphical Editing Framework*
IDE	*Integrated Development Environment*
ISO	*International Organization for Standardization*
JDK	*Java Development Kit*
ODBC	*Open Database Connectivity*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
SWT	*Standard Widget Toolkit*

SUMÁRIO

1 INTRODUÇÃO

A utilização de sistemas de bancos de dados vem se tornando cada vez mais comum no cenário atual, passando a fazer parte do cotidiano de milhares de empresas. As necessidades de cada negócio exigem que as empresas dependam cada vez mais das suas infra-estruturas de Tecnologias de Informação (TI), com mudanças freqüentes tais como novas aplicações em *E-commerce*, *Enterprise Resource Planning* (ERP), entre outras. Neste ambiente, a exigência de gerenciar grandes quantidades de informações com segurança e agilidade tem se tornado cada vez maior (HEUSER, 2001).

Ao desenvolver um sistema de banco de dados é imprescindível que seja realizado um bom planejamento, seguindo metodologias de desenvolvimento adequadas. Este planejamento é indispensável para que a elaboração de projetos de banco de dados atenda todas as necessidades dos usuários, obtendo o mínimo possível de falhas e facilitando a sua manutenção posteriormente.

Durante a fase de projeto, normalmente são elaborados e documentados os modelos de dados ou Diagramas Entidade-Relacionamento (DE-R). Um modelo é a abstração da complexidade do banco de dados, onde se podem definir a estrutura e comportamento de maneira compreensível aos usuários, programadores e analistas. O modelo de dados é essencial para a manutenção de um sistema de bancos de dados, pois por meio dele é possível compreender, de forma rápida e fácil, o funcionamento do sistema.

No entanto, um problema muito comum encontrado nas instituições é que boa parte dos sistemas de bancos de dados existentes não possuem a documentação adequada e nem mesmo os modelos de dados, pois não foram utilizadas metodologias adequadas no seu desenvolvimento. Existem situações em que o modelo de dados pode

ser de grande valia, por exemplo, na manutenção do sistema, implementação de novas consultas e na migração dos dados para uma nova plataforma. Com a ausência de modelos, a realização dessas tarefas se torna muito complicada e conseqüentemente mais demorada.

Buscando solucionar o problema anteriormente identificado, foram desenvolvidos métodos de engenharia reversa para bancos de dados capazes de reconstruir o modelo lógico e conceitual a partir desses bancos já implementados, a fim de facilitar o processo de manutenção desses sistemas ou então a migração dos seus dados para uma nova aplicação. Atualmente, no mercado existem diversas ferramentas *Computer Aided Software Engineering* (CASE), que auxiliam na criação de projetos de bancos de dados e no processo de engenharia reversa dos principais Sistemas Gerenciadores de Banco de Dados (SGBD).

O fato da maioria das ferramentas para projetos de bancos de dados realizarem o processo de engenharia reversa revela a importância do assunto e também reforça os objetivos deste trabalho.

1.1 OBJETIVO GERAL

Desenvolver um sistema de apoio à engenharia reversa para a geração do modelo conceitual a partir de bancos de dados relacionais por meio da extração de metadados[1].

[1] descrições dos dados contidos no banco de dados (ELMASRI; NAVATHE, 2005).

1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral deste trabalho podem-se destacar os seguintes objetivos específicos:

a) estudar conceitos de modelagem semântica de bancos de dados;
b) compreender conceitos de engenharia reversa;
c) compreender o dicionário de dados dos SGBD Oracle e Firebird;
d) permitir a análise do esquema de um banco de dados relacional já implementado;
e) implementar uma solução para a engenharia reversa de bancos de dados relacionais a partir da extração de metadados.

1.3 JUSTIFICATIVA

O advento e a popularização dos bancos de dados relacionais permitiram que muitas pessoas pudessem implantar bases de dados em empresas de diversos segmentos. Para isso, existem inúmeras ferramentas que permitem a criação de bancos de dados, normalmente em notação SQL, de forma simples e rápida. Porém, em grande parte dos casos, os bancos de dados são desenvolvidos sem um planejamento adequado, ou seja, sem a análise prévia e sem a criação de modelos conceituais ocasionando erros de implementação e dificuldades na manutenção do sistema.

O modelo E-R é de fundamental importância para se obter um bom projeto de banco de dados e permitir sua manutenção de forma rápida e eficaz. Existem várias ferramentas capazes de obter um modelo de dados a partir da engenharia reversa, porém, a maioria delas tem um alto custo ou então, os modelos obtidos não

compreendem alguns conceitos importantes da modelagem de bancos de dados, como por exemplo, especialização/generalização, apresentação da cardinalidade correta dos relacionamentos, entidades fracas entre outros.

Optou-se pelo desenvolvimento de uma solução livre e de código aberto, para aplicar esses conceitos de engenharia reversa, pois o desenvolvimento desta ferramenta proporcionará ao usuário, desenvolvedor ou analista, uma maior facilidade de compreensão dos bancos de dados, facilitando sua manutenção ou reengenharia e também fornecerá a documentação por meio do modelo conceitual. Por ser um software livre e de código aberto, este trabalho poderá contribuir para a comunidade acadêmica, permitindo dar continuidade às pesquisas realizadas durante o seu desenvolvimento.

1.4 ESTRUTURA DO TRABALHO

O trabalho foi subdividido em seis capítulos, incluindo a fundamentação teórica, implementação prática e análise dos resultados. Os objetos de pesquisa apresentados no decorrer do trabalho são: projeto de banco de dados, modelo E-R, engenharia reversa e metodologia para a realização da engenharia reversa de bancos de dados relacionais.

A pesquisa realizada sobre o projeto de banco de dados e as suas etapas de desenvolvimento é abordada no Capítulo 2. No Capítulo 3 é apresentado o modelo E-R assim como o modelo E-R Estendido (E-RE) e exemplos de ferramentas para o desenvolvimento de projetos de bancos de dados.

No Capítulo 4 são apresentados conceitos de engenharia reversa e seu co-relacionamento com reengenharia, reestruturação e engenharia avante. Neste mesmo capítulo é abordada a engenharia reversa aplicada a bancos de dados e, em seguida, é

apresentada a metodologia para a realização da engenharia reversa aplicada neste trabalho. O Capítulo subseqüente apresenta dois trabalhos correlatos nos quais foram utilizadas diferentes metodologias para a realização da engenharia reversa.

O Capítulo 6 apresenta a descrição da ferramenta de testes implementada durante este trabalho, a qual é composta da modelagem do sistema, detalhes sobre seu desenvolvimento e explicações sobre seu funcionamento. Neste capítulo também é feito uma análise dos resultados obtidos, os desafios encontrados e suas respectivas soluções. Por fim, são apresentadas as considerações finais e sugestões para trabalhos futuros.

2 O PROJETO DE BANCO DE DADOS

Nos anos 70, Ted Codd da IBM foi o primeiro a introduzir conceitos de modelos relacionais em um documento que chamou a atenção devido a sua simplicidade e seus fundamentos matemáticos. Os bancos de dados relacionais surgiram no início da década de 80, quando as empresas começaram a substituir seus arquivos convencionais e bancos de dados hierárquicos por bancos de dados relacionais (ELMASRI; NAVATHE, 2002).

Antes de surgirem os bancos de dados relacionais, os projetos dos bancos de dados eram realizados apenas por profissionais da área de informática, especializados neste tipo de desenvolvimento. Normalmente, esses especialistas tinham formação superior em matemática ou computação e trabalhavam em grandes *mainframes*. Entretanto, com o surgimento dos bancos de dados relacionais, surgiram softwares de bancos de dados que rodavam em microcomputadores, que eram muito mais fáceis de gerenciar do que nos *mainframes*.

A facilidade de usar esses softwares fez com que muitas pessoas desenvolvessem seus próprios bancos de dados para sua organização ou departamento, pois é relativamente simples criar tabelas e campos na maioria desses softwares, assim como também existe pouca complexidade para criar formulários, relatórios e visões. Porém, sem o planejamento adequado, os resultados obtidos na maioria das vezes, são bancos de dados incompletos ou superficiais (HERNANDEZ, 2000).

Os efeitos de um banco de dados mal projetado começam a surgir mais tarde, podendo destacar os problemas mais comuns, que são dados faltantes ou inconsistentes, informações sem precisão e dados redundantes (HEUSER, 2001).

Com o objetivo de evitar esses problemas foram desenvolvidas técnicas padronizadas para o desenvolvimento de projetos de bancos de dados, onde todas as etapas devem ser documentadas durante a execução.

2.1 ETAPAS DO PROJETO DE BANCO DE DADOS

A elaboração de projetos de bancos de dados se dá em três etapas principais: o projeto conceitual, o projeto lógico e o projeto físico conforme a Figura 1 (HEUSER, 2001; MACHADO; ABREU, 1999).

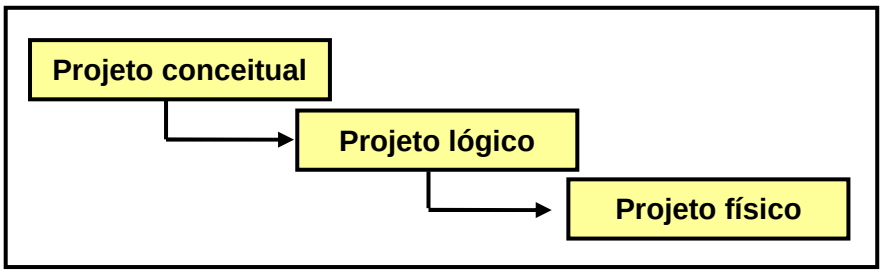


Figura 1. Etapas de um Projeto de Banco de Dados
Fonte: HEUSER; C. (2001); MACHADO; F.; ABREU; M. (2002)

Durante a fase de projeto conceitual de um sistema de banco de dados, devem ser elaborados e documentados os modelos de dados ou diagramas E-R. De acordo com Date (2004, p.14) um modelo de dados “é uma definição abstrata, autônoma e lógica dos objetos, operadores e outros elementos que, juntos, constituem a *máquina abstrata* com a qual os usuários interagem”. Ou seja, o modelo é a abstração da complexidade do banco de dados, onde se podem definir a estrutura e o comportamento de maneira compreensível aos usuários, programadores e analistas. Conforme Heuser (2001), Elmasri e Navathe (2002), o modelo de dados é essencial para a manutenção de um sistema de bancos de dados, pois por meio dele é possível compreender de forma rápida e fácil o funcionamento do sistema.

A primeira etapa denominada projeto conceitual, consiste em definir a estrutura do banco de dados com alto nível de abstração. No projeto lógico deve-se transformar esta abstração em uma representação de dados, de maneira que ela possa ser interpretada por um SGBD (ELMASRI; NAVATHE, 2002).

Cada etapa do projeto de banco de dados deve ser documentada formalmente. Essa documentação servirá como referência para correções e novas implementações sobre o mesmo.

2.1.1 Projeto Conceitual

O projeto conceitual é a descrição de um banco de dados de modo independente de como ele será implementado, ou seja, através deste podem-se ver quais os dados deverão conter o banco, porém não como esses dados serão implementados (ELMASRI; NAVATHE, 2002). O projeto conceitual é então, um modelo abstrato que descreve a estrutura do banco de dados independente de um SGBD (HEUSER, 2001).

2.1.2 Projeto Lógico

O projeto lógico é a descrição de um banco de dados em um nível mais baixo de abstração do que o projeto conceitual, pois ele é dependente do tipo particular do SGBD utilizado. O projeto lógico define a estrutura dos dados como ela é vista pelo usuário de um SGBD (HEUSER, 2001; ELMASRI; NAVATHE, 2002).

2.1.3 Projeto Físico

Durante esta etapa são especificadas as estruturas de armazenamento interno, os caminhos de acesso e as organizações para os arquivos do banco de dados (ELMASRI; NAVATHE, 2002). O projeto físico pode ser tratado como uma atividade separada, a ser executada em seguida, pois conforme Date (2004) o modo “certo” de projetar um banco de dados é primeiro criar o projeto lógico e depois, como uma etapa separada e subseqüente, realizar o mapeamento do projeto lógico nas estruturas físicas de um SGBD.

3 O MODELO ENTIDADE-RELACIONAMENTO

Conforme Machado e Abreu (2002) o modelo E-R foi definido por Peter Chen e teve como base a teoria relacional criada por Codd (1970). Este modelo pode ser considerado como um padrão para a modelagem conceitual.

No modelo definido por Chen, destaca-se a importância de reconhecer os objetos que compõem o negócio para o qual se deseja desenvolver um sistema, sem preocupar-se com as formas de tratamento das informações. Esses objetos foram classificados por ele como entidades e relacionamentos (MACHADO; ABREU, 2002).

O modelo E-R é formado por três componentes básicos: conjunto de entidades, relacionamentos e atributos.

3.1 ENTIDADES

Uma entidade representa um conjunto de objetos da realidade modelada. Para a modelagem de bancos de dados o que interessa são somente os objetos aos quais se deseja manter informações, por exemplo, clientes, fornecedores, produtos, pedidos, compras e vendas, etc. Em um modelo E-R, uma entidade é representada por um retângulo que contém o nome da entidade, conforme a Figura 2 (HEUSER, 2001).

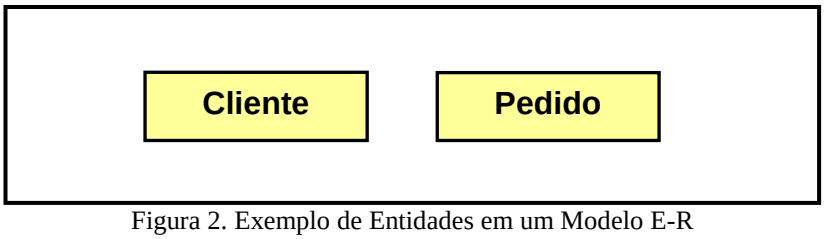


Figura 2. Exemplo de Entidades em um Modelo E-R

3.2 ATRIBUTOS

Além das entidades, o modelo E-R permite especificar outra propriedade, que é a identificação de seus atributos. Estes são utilizados para associar informações à ocorrências de entidades ou de relacionamentos.

De acordo com Silberschatz, Korth e Sudarshan (2006) em um modelo E-R os atributos podem ser classificados como:

a) **atributos simples e compostos**: um atributo simples ou atômico é aquele que por si só pode-se obter a informação completa em uma consulta. Este tipo não pode ser subdividido. O atributo composto precisa de outro para compor a informação completa. Esses podem ser subdivididos em vários outros atributos. Um exemplo poderia ser o endereço de um cliente que está estruturado em *rua, número, bairro e CEP*. Os atributos compostos agrupam atributos correlacionados, tornando o modelo de certa forma mais claro, conforme visualizado na Figura 3;

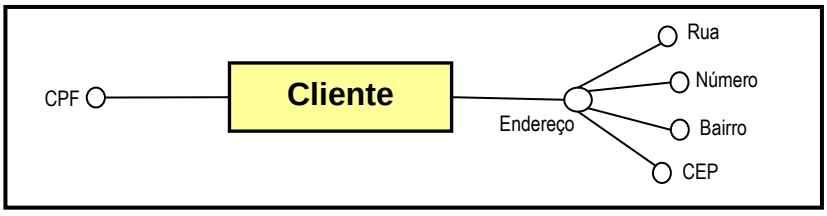


Figura 3. Exemplo de Atributo Simples e Atributo Composto

b) **atributos multivalorados**: podem ocorrer casos em que um atributo possua um conjunto de valores para uma única entidade. Pode-se considerar a entidade *cliente* e os atributos *e-mail e telefone*. Um cliente, dependendo da situação, pode ter nenhum, um ou vários e-mails e

telefones. Esse tipo de atributo pode ser representado no modelo E-R, conforme a Figura 4;

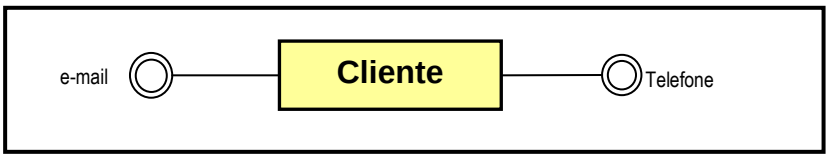


Figura 4. Exemplo de Atributos Multivalorados

c) **atributos derivados**: são aqueles cujos valores são obtidos por meio de outros atributos da mesma entidade. Por exemplo, a idade pode ser obtida a partir da data de nascimento. Portanto, esses não necessitam permanecer armazenados no banco de dados, uma vez que podem ser obtidos por meio de outros atributos. Estes podem ser representados no modelo E-R conforme a Figura 5;

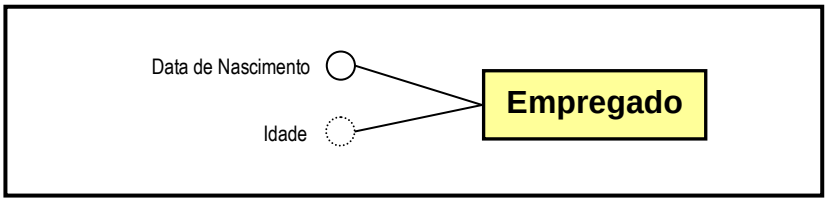


Figura 5. Exemplo de Atributo Derivado

3.2.1 Atributos Chave

Atributos chave são identificadores únicos, utilizados para distinguir a ocorrência de uma entidade das demais ocorrências da mesma. Esses identificadores podem ser conjuntos de um ou mais atributos, que devem possuir valores únicos dentro de uma entidade. Em um modelo E-R, os atributos chaves podem ser representados por um círculo preto, como mostrado na Figura 6, ou podem ser representados por um círculo contendo o nome do atributo sublinhado.

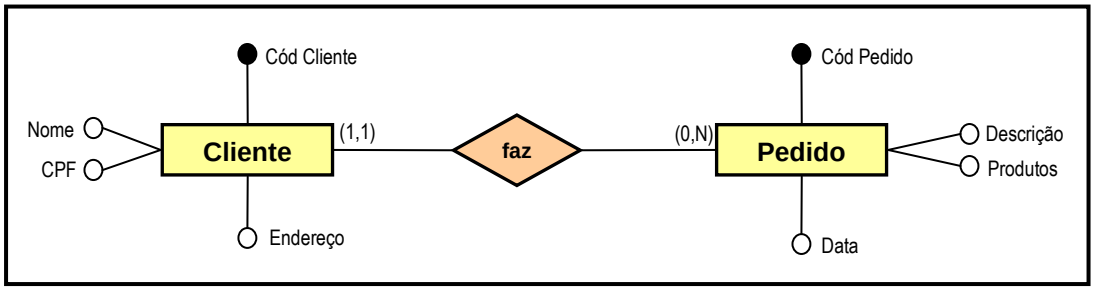


Figura 6. Exemplo de Entidades, Atributos e Relacionamentos em um Modelo E-R

A Figura 6 apresenta uma forma muito comum para representação de entidades, atributos e relacionamentos em um modelo E-R. No entanto, existem inúmeras outras formas de representação. O projetista é quem irá definir qual a melhor forma para representar os objetos de seus projetos (ELMASRI; NAVATHE, 2002).

3.3 ENTIDADES FRACAS

Entidades que não possuem seus próprios atributos chave são chamadas de entidades fracas. Estas são relacionadas à entidades específicas de um outro tipo de entidade em combinação com alguns de seus atributos. Esses outros tipos são denominados entidades identificadoras ou proprietárias, e o tipo de relacionamento que atribui um tipo de entidade fraca a sua proprietária denomina-se relacionamento identificador da entidade fraca (ELMASRI; NAVATHE, 2002). Essas podem ser representadas no modelo E-R, como um retângulo com borda mais espessa ou dupla, conforme se pode visualizar na Figura 7.

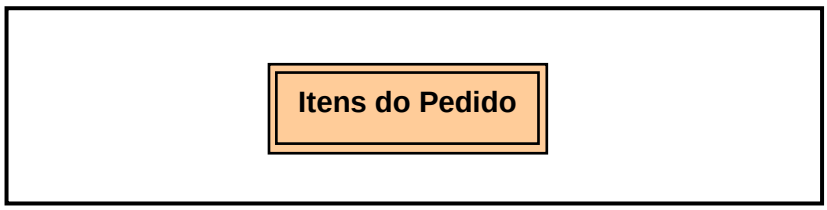


Figura 7. Representação de uma Entidade Fraca

3.4 RELACIONAMENTOS

Relacionamentos são as associações entre as entidades representadas, ou seja, é a conexão estabelecida entre elas. Em um modelo E-R, geralmente os relacionamentos são representados por um losango ligado às respectivas entidades. Uma entidade pode ter um relacionamento com ela mesma ou com outras. O mais comum é o relacionamento entre diferentes entidades (HERNANDEZ, 2000). A Figura 8 demonstra como um relacionamento pode ser representado em um modelo E-R.

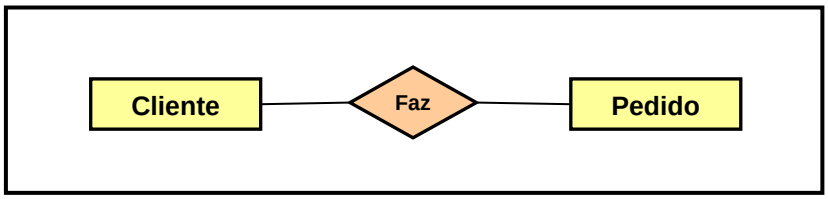


Figura 8. Forma mais comum de representar Relacionamentos

3.4.1 Classificação de Relacionamentos Binários

Conforme Heuser (2001, p.16), “um *relacionamento binário* é aquele cujas ocorrências contêm duas ocorrências de entidade”. Os relacionamentos binários podem ser classificados em:

a) **relacionamento um-para-um (1:1):** ocorre quando um registro na entidade *A* pode estar relacionado a no máximo um registro na entidade *B*, e um registro na entidade *B* pode estar relacionado a no máximo um registro na entidade *A*, conforme representado na Figura 9;

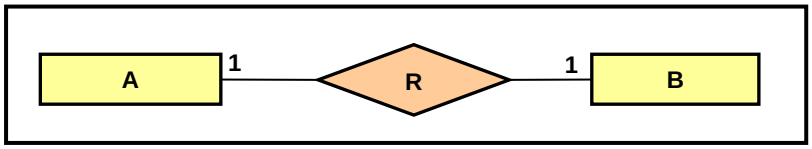


Figura 9. Relacionamento 1:1

b) **relacionamento um-para-muitos (1:n):** ocorre quando um registro na entidade *A* pode estar relacionado a mais de um registro na entidade *B*, e um registro na entidade *B* pode estar relacionado a no máximo um registro na entidade *A*, conforme representado na Figura 10;

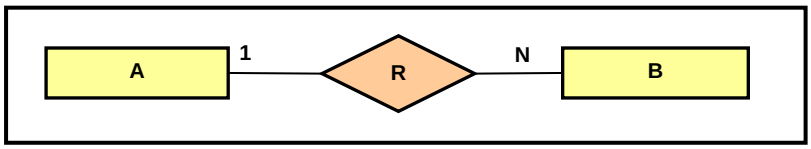


Figura 10. Relacionamento 1: N

c) **relacionamento muitos-para-muitos (n:n):** ocorre quando um registro na entidade *A* pode estar relacionado a mais de um registro na entidade *B*, e um registro na entidade *B* pode estar relacionado a mais de um registro na entidade *A*, conforme representado na Figura 11.

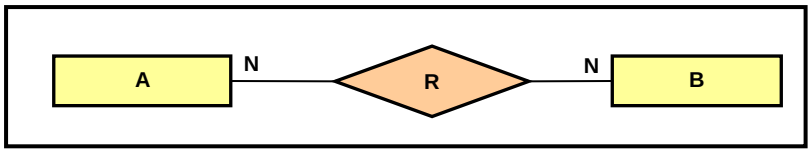


Figura 11. Relacionamento N: N

Não existe uma notação padronizada para a modelagem conceitual de bancos de dados. A notação pode ser usada com diferentes formas de representação, conforme a preferência do projetista, assim como as ferramentas para modelagem também utilizam várias notações diferentes. Algumas possuem conceitos e restrições adicionais e outras possuem menos restrições (ELMASRI; NAVATHE, 2002). No Anexo A podem ser visualizadas algumas das formas mais comuns de representações de um modelo E-R.

3.4.2 Cardinalidade dos Relacionamentos

Uma importante propriedade de um relacionamento é a de quantas ocorrências de uma entidade pode estar associada a uma determinada ocorrência de outra entidade. Essa propriedade denomina-se cardinalidade ou multiplicidade. Existem dois tipos de cardinalidade a considerar**:** *cardinalidade mínima e cardinalidade máxima* (HEUSER , 2001). “Cardinalidade (mínima, máxima) de um relacionamento é o número (mínimo, máximo) de ocorrências de entidades associadas a uma ocorrência da entidade em questão através do relacionamento” (HEUSER, 2001, p.16). Exemplo:

a) cardinalidade máxima: Em um modelo E-R a cardinalidade máxima pode ser representada conforme a Figura 12. Neste caso, a entidade *Pedido* tem cardinalidade máxima **1** no relacionamento *Faz*. Isso significa que um registro de *Pedido* pode estar associado a no máximo um registro de *Cliente*. Por sua vez, a entidade *Cliente* tem cardinalidade **N** no relacionamento *Faz*. Isso significa que um registro na entidade *Cliente* pode estar relacionado a vários registros na entidade *Pedido*;

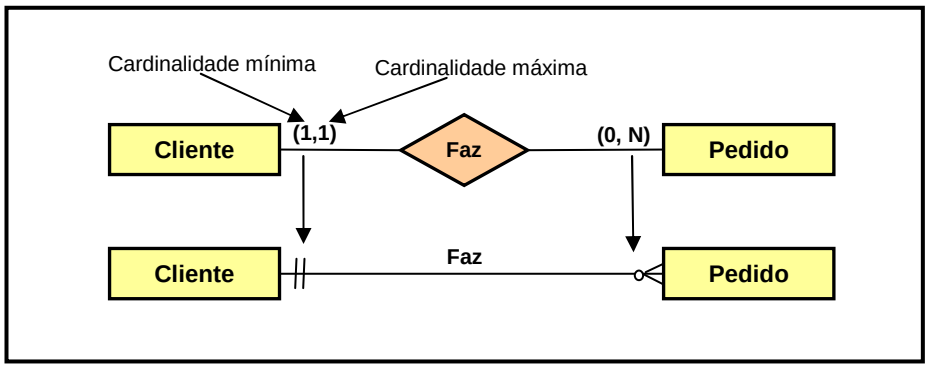


Figura 12. Exemplos de Representação de Cardinalidade ou Multiplicidade
Fonte: Adaptado de HEUSER; C. (2001)

b) cardinalidade mínima: é o número mínimo de registros que uma entidade pode ter no relacionamento em questão, conforme o exemplo

demonstrado na Figura 12, um *cliente* tem cardinalidade mínima **1** no relacionamento *Faz* e um *pedido* tem cardinalidade mínima **0** neste mesmo relacionamento. Ou seja, um registro na entidade *cliente* pode existir mesmo sem estar relacionado a nenhum registro na entidade *pedido*, e um registro na entidade *pedido* não pode existir sem estar relacionado à no mínimo um registro na entidade *cliente*.

3.5 MODELO E-R ESTENDIDO

O modelo E-R tradicional, descrito nos capítulos anteriores, é suficiente para representar bancos de dados comuns, que não possuem muita complexidade, por exemplo, bancos de dados para indústria e comércio. Porém, com o advento dos bancos de dados, surgiram novas aplicações que também utilizam essa tecnologia, por exemplo, aplicações para telecomunicações, multimídia, *data warehouse*, *data mining*, sistemas de informações geográficas (SIG), entre outras aplicações para a *Web*, que possuem requisitos mais complexos do que aplicações convencionais. Para modelar esses novos requisitos devem-se usar conceitos de modelagem semântica de dados. Com a inclusão desses conceitos, tem-se como resultado o modelo E-R Estendido, ou modelo E-RE (ELMASRI; NAVATHE, 2002).

O modelo E-RE pode ser considerado uma adaptação do modelo E-R para representar novas soluções orientadas a objetos. Ele pode ser aplicado para representar subclasses, superclasses e herança de tipo no modelo E-R, assim como especialização/generalização.

3.5.1 Subclasses, Superclasses e Herança

Além dos conceitos de modelo E-R, o modelo E-RE possui conceitos de subclasse, superclasse e conceitos correspondentes de especialização e generalização.

No modelo E-RE, uma entidade pode representar um tipo ou um conjunto de entidades daquele tipo. Existem casos em que um tipo de entidade possui vários subgrupos pertencentes ao mesmo. Por exemplo, a entidade *veículo*, representa o tipo de cada subgrupo desta entidade. Subgrupos do tipo *veículo* podem ser, por exemplo, *automóvel, caminhão* e *moto*. Cada subgrupo é considerado uma subclasse, enquanto o grupo *veículo* é denominado superclasse.

O modelo E-RE permite representar cada subclasse como um objeto distinto no banco de dados. O relacionamento entre a superclasse e a subclasse é denominado de relacionamento *superclasse/subclasse* (ELMASRI; NAVATHE, 2002).

3.5.2 Especialização e Generalização

A especialização permite definir conjuntos de subclasses de um tipo de entidade ou superclasse. Esse conjunto de subclasses é definido com base em alguma característica que distingue as entidades na superclasse. No exemplo citado anteriormente, o conjunto de subclasses *automóvel, caminhão e moto*, é uma especialização da superclasse *veículo* com base no atributo *tipo de veiculo*, conforme ilustrado na Figura 13.

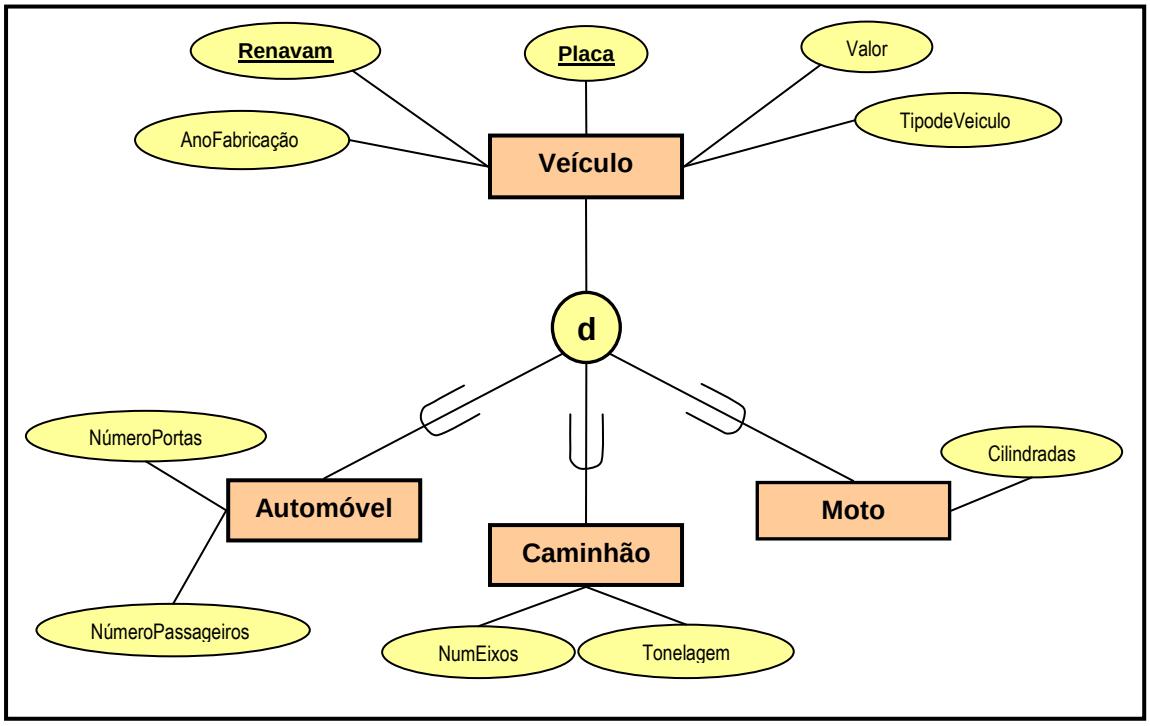


Figura 13. Exemplo de modelo ER estendido
Fonte: Adaptado de ELMASRI; NAVATHE (2002)

Ao analisar o modelo apresentado na Figura 13, pode-se verificar que a entidade *veículo*, que representa a superclasse, possui os atributos comuns para todas as subclasses. Estas por sua vez, possuem apenas os atributos específicos de cada uma. No entanto um *automóvel*, *moto* ou *caminhão* não podem existir sem que antes exista um *veículo*. Ou seja, uma mesma ocorrência na subclasse *Automóvel* deve existir também na superclasse *Veículo.* A especialização/generalização se assemelha a um relacionamento 1:1, sendo que a diferença é que no relacionamento 1:1 as duas entidades distintas estão apenas relacionadas entre si, já na especialização/generalização a entidade na subclasse é a mesma entidade da superclasse (ELMASRI; NAVATHE, 2002).

3.5.3 Restrições da Especialização/Generalização

Um mesmo tipo de entidade ou superclasse pode conter diversas especializações, assim como uma especialização pode conter somente uma única subclasse, que neste caso não é utilizado o círculo entre a subclasse e a superclasse. Para definir quais as subclasses se tornarão membros da superclasse, pode ser utilizada uma condição com base no valor de algum atributo da superclasse, que pode ser denominado de subclasse definida por predicado ou subclasse definida por atributo. Para representar uma subclasse definida por predicado deve-se adicionar a condição próxima à linha que liga o círculo à subclasse. Já em uma subclasse definida por atributo, o nome que define a especialização é colocado próximo à linha que liga o circulo à superclasse, e o valor que define cada subclasse é colocado próximo à linha que liga o circulo a cada subclasse, assim como exibido na Figura 14.

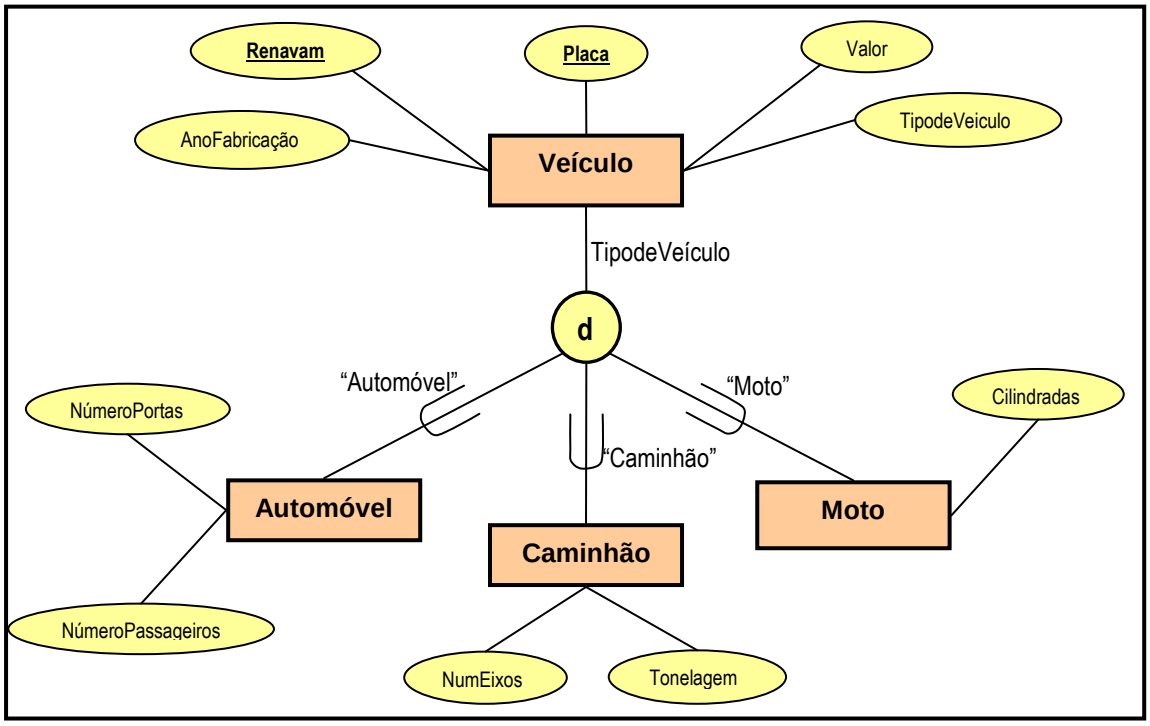


Figura 14. Exemplo de Disjunção no modelo E-RE
Fonte: Adaptado de ELMASRI; NAVATHE (2002)

Quando não existe nenhuma condição para determinar os membros na subclasse, pode-se dizer que a subclasse é definida pelo usuário.

Existem também outras restrições que devem ser utilizadas em uma especialização, são elas (ELMASRI; NAVATHE, 2002):

a) **disjunção**: ocorre quando uma entidade pode ser membro de no máximo uma das subclasses da especialização. A Figura 14 apresenta uma disjunção, onde a letra **d** no círculo significa disjunção.

b) **sobreposição**: ocorre quando a mesma entidade pode fazer parte de mais de uma subclasse da especialização. Este caso é representado colocando a letra **o** no círculo que liga a subclasse à superclasse, conforme apresentado na Figura 15.

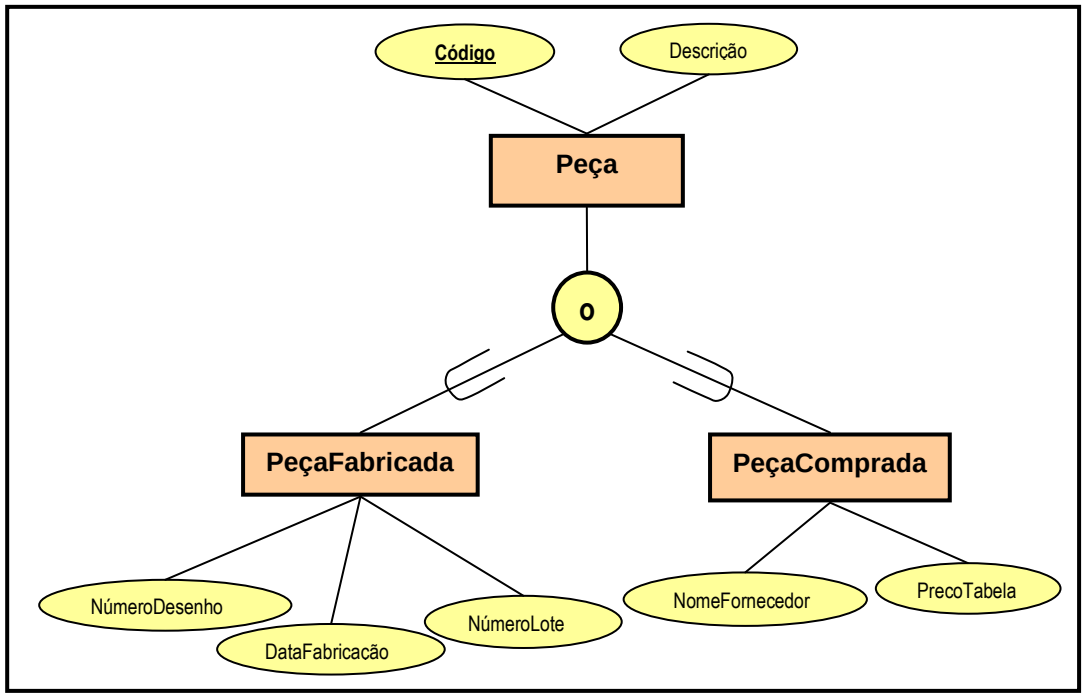


Figura 15. Exemplo de Sobreposição no modelo E-RE
Fonte: ELMASRI; NAVATHE (2002)

Existem ainda outras restrições que podem ser representadas em um modelo E-RE, que são as hierarquias e grades de especialização/generalização, porém, não será abordado nesta pesquisa, pois não será aplicado na ferramenta de testes proposta neste

trabalho. Nos capítulos seguintes serão apresentadas algumas ferramentas atualmente existentes no mercado de software, para a modelagem de bancos de dados.

3.6 FERRAMENTAS PARA PROJETO DE BANCO DE DADOS

Existem diversas ferramentas disponíveis no mercado para auxiliar os projetistas no desenvolvimento de projetos de bancos de dados. Essas ferramentas são denominadas ferramentas CASE, e são utilizadas na fase de projeto e desenvolvimento de sistemas de banco de dados (ELMASRI; NAVATHE, 2002).

As ferramentas CASE, para desenvolvimento de projetos de bancos de dados, se caracterizam pelo uso de uma interface gráfica para a construção do esquema conceitual e permitem também obter o esquema lógico através da própria ferramenta. Na Tabela 1, são apresentados alguns exemplos de ferramentas CASE disponíveis no mercado e o endereço eletrônico de cada fabricante.

Tabela 1. Exemplos de Ferramentas CASE

Ferramenta / Fabricante	Endereço Eletrônico
ERwin Data Modeler/ (Computer Associates)	www.ca.com/us/products/product.aspx?id=260
Toad Data Modeler/ (Quest Software).	www.quest.com/toad-data-modeler/upgrade_toaddatamodeler.aspx
Case Studio/ (CharonWare)	www.casestudio.com/enu/default.aspx
Dr. Case/ (Squadra)	www.squadra.com.br/site_squadra/PRODUCAO/portal/wwwsquadracombrprodutos/57.html
Visio/ (Microsoft)	www.microsoft.com/brasil/office/visio/default.asp
Together/ (Borland)	www.borland.com/br/products/together/index.html
Power Designer/ (Sybase)	www.sybase.com/products/modelingmetadata/powerdesigner
DB Designer/ (Fabulous Force Database)	http://fabforce.net/dbdesigner4/
Rational Rose Data Modeler (Rational)	www-306.ibm.com/software/awdtools/developer/datamodeler/
Poseidon/ (Gentle ware)	www.gentleware.com/products.html
Oracle Designer/ (Oracle)	www.oracle.com/technology/products/designer/index.html

Em seguida serão apresentadas duas ferramentas para o desenvolvimento de projetos de banco de dados, o DB Designer da Fabulous Force, que é uma solução gratuita para modelagem de bancos de dados e o Toad Data Modeler da Quest Software que é uma ferramenta proprietária.

3.6.1 DB Designer

O DB Designer é uma ferramenta CASE, que tem como objetivo principal criar projetos de banco de dados. Ele permite também realizar a engenharia reversa de bancos de dados, baseado no seu esquema relacional e gerar, a partir deste, o esquema conceitual.

O DB Designer é multi-plataforma e está disponível para os principais sistemas operacionais como Windows, Linux, Mac OS, entre outros. Este software foi desenvolvido para interação nativa com o SGBD MySQL. Entretanto, permite também a conexão a outros SGBD, desde que estes possuam uma interface *Open Data Base Connectivity* (ODBC). Esta ferramenta encontra-se disponível para uso livre, seguindo as premissas do projeto GNU e sobre as regras da Licença da General Public License (GPL) (FABULOUS FORCE DATABASE TOOLS, 2005). Na Figura 16 é apresentada um ilustração com a interface do DB Designer.

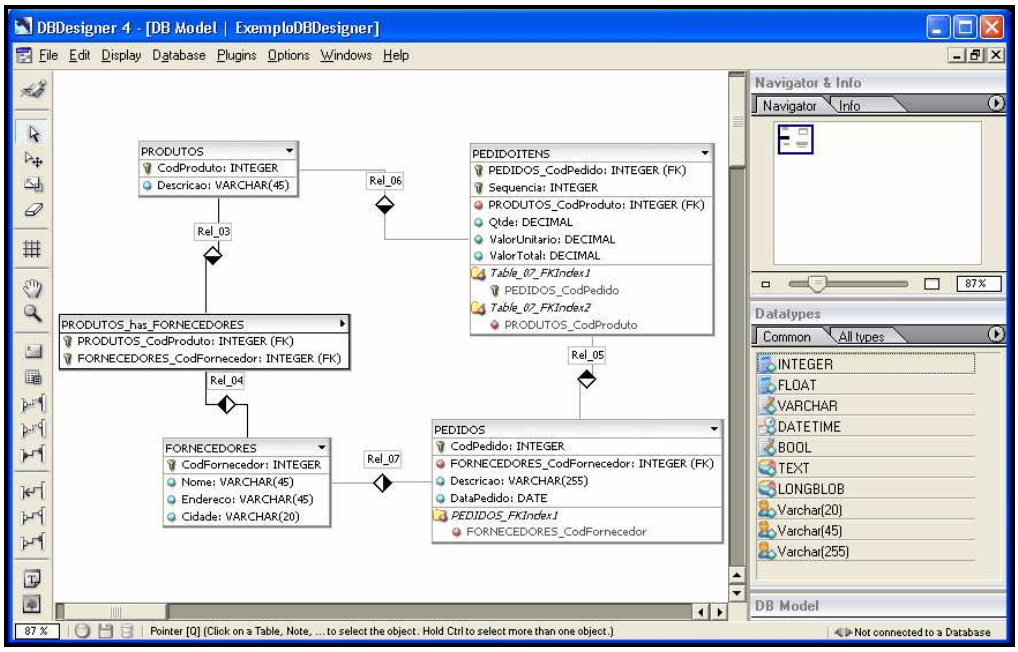


Figura 16. Interface Gráfica do DB Designer
Fonte: FABULOUS FORCE DATABASE TOOLS (2005)

O DB Designer possui uma interface simples e de fácil compreensão. Como mostra a Figura 16, na parte superior, encontra-se um menu com todas as opções e funcionalidades disponíveis. Na lateral esquerda, encontra-se uma paleta de opções para a manipulação, edição e também os objetos para a montagem do diagrama E-R. Na lateral direita existem duas abas, uma para navegar sobre o modelo e outra para visualizar informações sobre o objeto atualmente selecionado na tela. Logo abaixo, existe uma relação dos tipos de dados, que pode ser utilizada para a definição dos atributos e ao centro a visualização do modelo E-R.

3.6.2 Toad Data Modeler

O Toad Data Modeler é uma ferramenta CASE fornecida pela Quest Software para auxiliar no desenvolvimento de projetos de bancos de dados. Esta ferramenta permite criar diagramas E-R, diagramas de fluxo de dados (DFD) e o

modelo lógico para mais de 40 bancos de dados distintos. Permite também realizar a engenharia reversa de mais de 20 bancos de dados diferentes. Na Figura 17 é apresentada a interface do Toad.

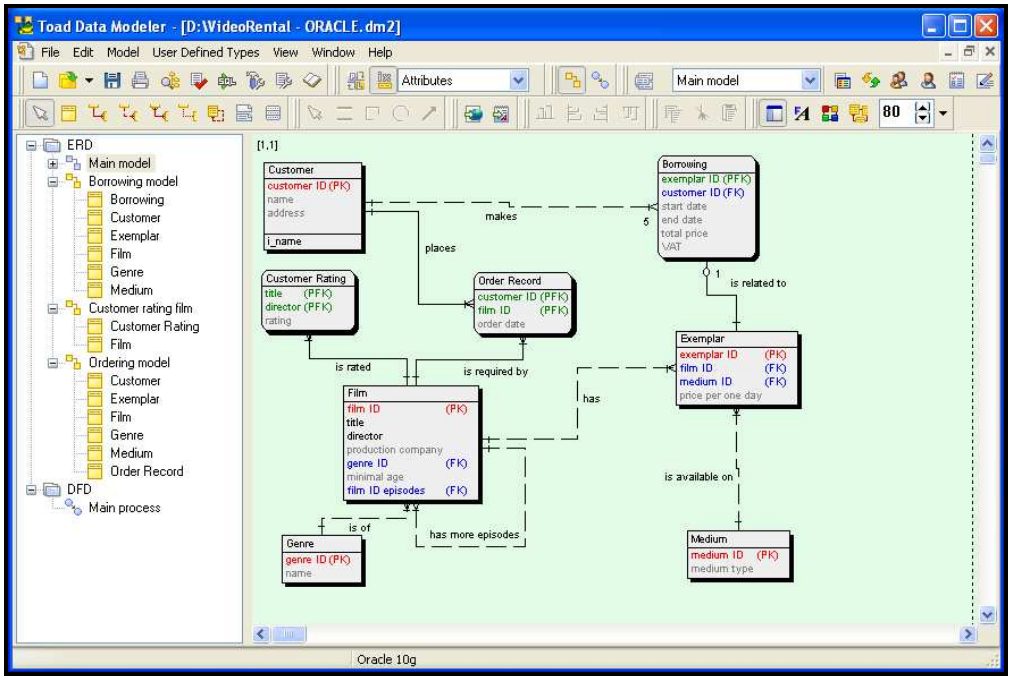


Figura 17. Interface Gráfica do Toad Data Modeler
Fonte: QUEST SOFTWARE (2007)

Ao observar a Figura 17, pode-se verificar um menu na parte superior da tela, assim como as barras de ferramentas com as principais opções de edição e modelagem. Na lateral esquerda encontra-se uma árvore, onde se podem visualizar os objetos do diagrama E-R localizado no centro direito da tela.

Pode-se observar também, que a notação utilizada pelo Toad, assim como na maioria das ferramentas CASE, podem ser diferenciadas e uma mesma ferramenta pode disponibilizar diferentes notações.

A grande maioria das ferramentas citadas anteriormente na Tabela 1, tem a capacidade de realizar a engenharia reversa dos principais bancos de dados, porém, o resultado obtido nem sempre é satisfatório, uma vez que a engenharia destas ferramentas não contempla algumas restrições da modelagem conceitual, como por exemplo, a obtenção da cardinalidade correta dos relacionamentos, a

especialização/generalização, assim como alguns conceitos da modelagem semântica de dados. Este fato vem reforçar ainda mais a importância da pesquisa sobre a engenharia reversa para bancos de dados.

4 ENGENHARIA DE SOFTWARE

A terminologia empregada na engenharia de software para referenciar as tecnologias já existentes é apresentada por Chikofski (1990 apud BRAGA, 1998) com o objetivo de racionalizar os termos que já estão em uso. Conforme a Figura 18, neste contexto existe quatro terminologias distintas (BRAGA, 1998):

a) **engenharia avante**: é o processo tradicional a partir de um alto nível de abstração para chegar ao nível físico de implementação do sistema;
b) **reestruturação**: é a transformação de uma forma de representação para outra no mesmo nível de abstração, preservando o comportamento externo do sistema;
c) **engenharia reversa**: é o processo de análise de um sistema, para identificar seus componentes e criar uma representação em um nível mais alto de abstração;
d) **reengenharia**: tem por finalidade examinar e alterar um sistema existente para reconstruí-lo de uma nova forma, com a finalidade de melhorar a qualidade global do sistema.

A Figura 18 ilustra os termos descritos anteriormente por Braga (1998) e mostra a engenharia reversa, reengenharia e reestruturação em todas as etapas de desenvolvimento do projeto, seja no levantamento de requisitos, projeto ou implementação.

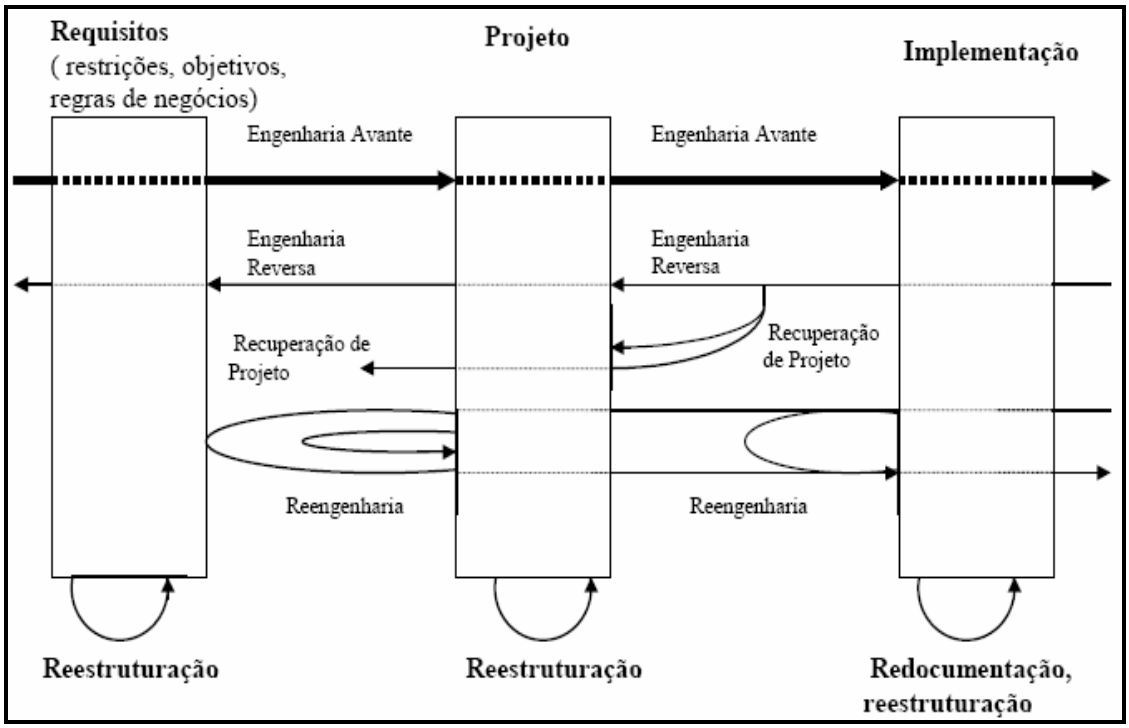


Figura 18. Relacionamento entre Engenharia Avante, Reengenharia, Engenharia Reversa
Fonte: CHIKOFSKI. E. (1990 apud BRAGA, 1998)

4.1 ENGENHARIA REVERSA DE BANCOS DE DADOS

A engenharia reversa de bancos de dados pode ser definida como um processo de abstração, que parte de um projeto de banco de dados pronto e resulta em um modelo conceitual. Ela pode ser considerada uma etapa incorporada ao projeto quando já existe um banco de dados implementado e deseja-se reestruturar ou criar um novo projeto (HEUSER, 2001). A Figura 19 ilustra a etapa de engenharia reversa incorporada ao projeto de banco de dados.

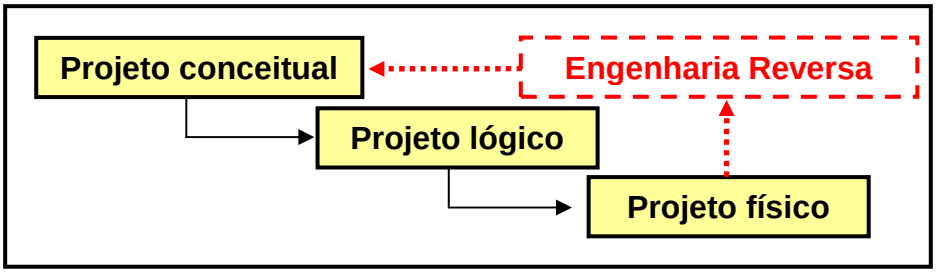


Figura 19. A Engenharia Reversa incorporada ao Projeto de Banco de Dados
Fonte: Adaptado de MACHADO, F.; ABREU, M. (2002)

Existem situações no ciclo de vida de um sistema em que o modelo de dados pode ser de grande valia, por exemplo, na manutenção rotineira do sistema, na criação de novas consultas ou então na migração dos dados para uma nova plataforma. A disponibilidade de uma documentação abstrata na forma de modelos conceituais pode acelerar significativamente o processo de migração, pois permite encurtar a etapa de modelagem de dados do novo sistema (HEUSER, 2001).

Um problema comum encontrado nas empresas atualmente é a inexistência de documentação e de modelos conceituais dos sistemas de bancos de dados. Isso ocorre, muitas vezes, pela não utilização de metodologias de desenvolvimento adequadas para o desenvolvimento dos sistemas.

Buscando solucionar este problema, foram desenvolvidos métodos de engenharia reversa para bancos de dados capazes de reconstruir o modelo conceitual de bancos já implementados, a fim de facilitar o processo de manutenção desses sistemas, ou então, a migração dos seus dados para uma nova aplicação.

Um exemplo importante para a utilização da engenharia reversa de bancos de dados é no desenvolvimento de novos sistemas, baseados em sistemas já existentes. Muitas vezes se torna mais fácil desenvolver um novo sistema a partir de outro que já se encontra em funcionamento do que reescrever todo o projeto. Nesse caso, o modelo conceitual pode ser usado como documentação abstrata durante discussões entre usuários, analistas e programadores, pois, o modelo conceitual permite que pessoas que não conheçam o sistema possam aprender mais rapidamente o seu funcionamento (HEUSER, 2001).

Outro caso de uso para esse tipo de ferramenta é no processo de migração de dados para uma nova plataforma. Essa prática tem se tornado muito comum atualmente, pois, o crescimento e o surgimento de novas necessidades, muitas vezes,

implicam na aquisição de uma nova ferramenta que atenda melhor as necessidades dos negócios da empresa. A disponibilidade de um modelo abstrato de dados do sistema existente pode acelerar o processo de migração dos dados, pois permite encurtar a etapa de modelagem de dados do novo banco de dados (HEUSER, 2001). A Figura 20 mostra o diagrama de casos de uso para o sistema de engenharia reversa proposto.

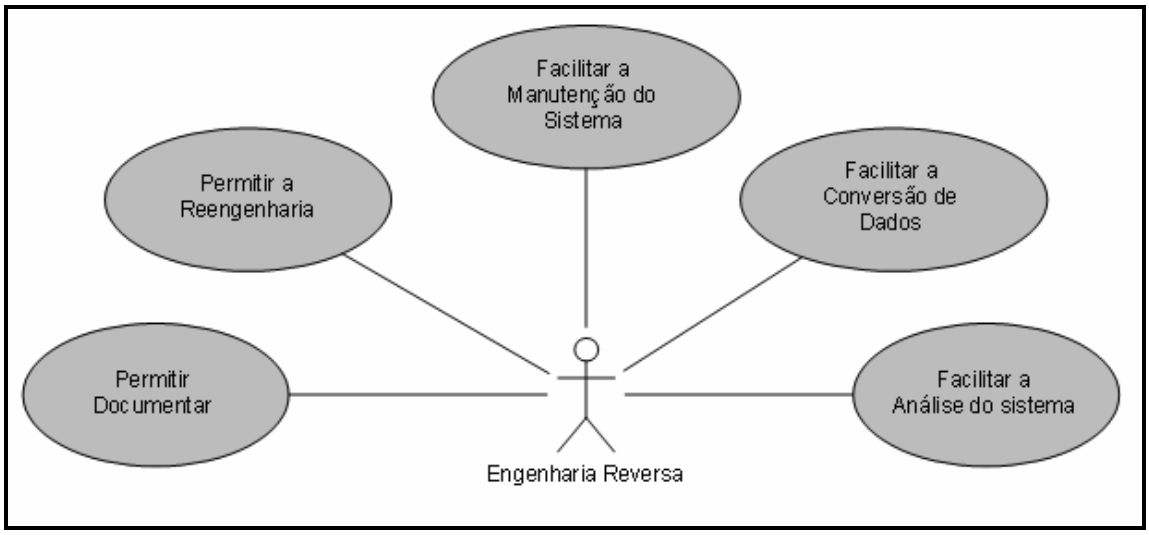


Figura 20. Casos de uso da Engenharia Reversa

No caso das empresas de desenvolvimento de sistemas, que precisam migrar os dados de sistemas desconhecidos para a uma nova aplicação, é essencial que se tenha conhecimento da estrutura do sistema que se pretende extrair os dados. No entanto, a linguagem SQL, padronizada pela ANSI e ISO, possui inúmeras variações e extensões para cada SGBD (SOUZA, 2005). Nesse caso, para facilitar o entendimento sobre o sistema original, seria ideal ter acesso ao modelo conceitual do mesmo. Porém, na maioria das vezes, as empresas que realizam a implantação de um novo sistema não têm acesso à documentação do antigo sistema ou então a documentação é inexistente ou incompleta.

Em razão disto, este trabalho propõe uma metodologia para auxiliar na reconstrução dos diagramas E-R de diferentes bancos de dados. Desta forma, pode-se

então encurtar as tarefas de desenvolvimento de novos sistemas baseados em sistemas já existentes e também a migração de dados para uma nova plataforma.

Assim como no projeto de banco de dados, a engenharia reversa deve seguir uma metodologia para obter o modelo conceitual. Para desenvolver este projeto será utilizada a metodologia proposta por Heuser (2001), que tem como ponto de partida um modelo lógico de um banco de dados relacional e, como resultado, um modelo conceitual.

4.2 METODOLOGIA PARA A REALIZAÇÃO DA ENGENHARIA REVERSA

Atualmente existem inúmeras metodologias para a criação do projeto de banco de dados, porém a maioria dos livros sobre o assunto não abordam a engenharia reversa como parte do projeto de banco de dados. Por este motivo, ainda existe pouca diversidade de metodologias sobre o assunto.

A metodologia apresentada neste trabalho segue as especificações apresentadas por Heuser (2001), que define o processo de engenharia reversa de modelos relacionais em quatro etapas distintas:

a) identificação da construção do modelo E-R correspondente a cada tabela;
b) definição dos relacionamentos 1:N e 1:1;
c) definição de atributos;
d) definição de identificadores de entidades e relacionamentos.

4.2.1 Construção do Modelo E-R correspondente a cada Tabela

Nesta etapa, deve-se definir para cada tabela do modelo relacional, qual a construção correspondente ao nível de modelo E-R. Uma tabela do banco de dados pode representar no modelo E-R:

a) uma entidade;

b) um relacionamento;

c) uma entidade especializada.

Para exemplificar este processo, será utilizado um banco de dados para um sistema de vendas, que está representado a seguir:

produtos (**codproduto**, descrição, valorunitario)

clientes (**codcliente**, nome, endereço)

pessoafisica (**codcliente**, cpf, rg, nomedopai, nomedamãe)

pessoajuridica (**codcliente**, razãosocial, cnpj, inscriçãoestadual)

produtosdocliente (**codcliente**,**codproduto**)

pedido (**codpedido**, codcliente, dataemissão, dataentrega)

pedidoitens (**codpedido,coditem**, codproduto, descrição, qtde, valorunitario)

O que irá determinar ao que corresponde uma tabela no modelo E-R é a composição da sua chave primária. As tabelas podem ser classificadas em três tipos, de acordo com sua chave primária (HEUSER, 2001):

a) **regra 1**: chave primária composta por mais de uma chave estrangeira. A tabela que possui uma chave primária composta de múltiplas chaves estrangeiras implementa um relacionamento N:N entre as entidades correspondentes às tabelas referenciadas pelas chaves estrangeiras.

Como exemplo, pode ser a tabela *produtosdocliente*, cuja chave primária é composta por *codcliente* e *codproduto*. Ambas as colunas são chaves estrangeiras em relação às tabelas cliente e produtos, respectivamente. Conforme representado na Figura 21, neste caso, a tabela *produtosdocliente* representa um relacionamento N:N entre clientes e produtos;

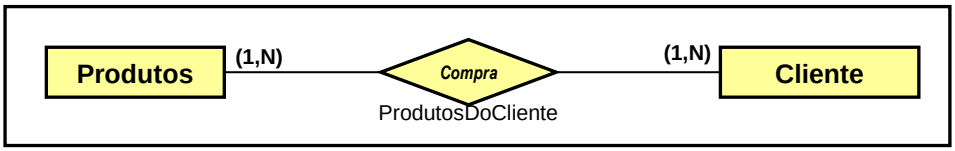


Figura 21. Exemplo de tabela implementando um Relacionamento N:N

b) **regra 2**: toda chave primária é uma chave estrangeira. A tabela cuja chave primária é toda ela uma única chave estrangeira, representa uma entidade que forma uma especialização da entidade correspondente à tabela referenciada pela chave estrangeira, por exemplo, as tabelas *pessoafísica* e *pessoajuridica* possuem a chave primária *codcliente*, que é chave estrangeira com a tabela *clientes*. Isso implica que uma linha na tabela *pessoafisica* ou *pessoajuridica* só poderá existir quando uma linha com a mesma chave na tabela *clientes* também existir, ou seja, essas tabelas são especializações da tabela *clientes* conforme representado na Figura 22;

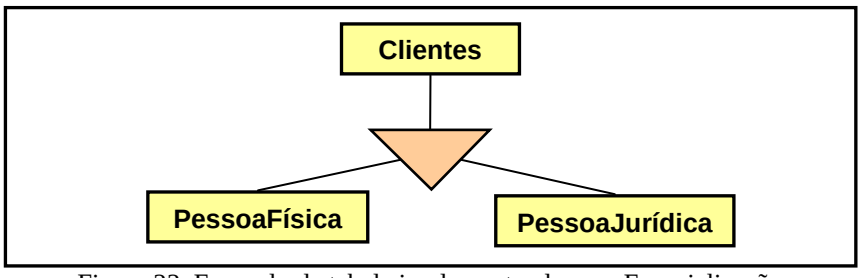


Figura 22. Exemplo de tabela implementando uma Especialização

c) **regra 3**: quando a chave primária da tabela em questão não for composta de múltiplas chaves estrangeiras (regra 1), nem for toda ela composta de uma única chave estrangeira (regra 2), a tabela em questão representa uma entidade forte ou uma entidade fraca, por exemplo, a tabela *pedido* cuja chave primária não contém chaves estrangeiras, representa uma entidade forte. Já na tabela *pedidoitens*, a chave primária é composta por *codpedido* e *coditem*, sendo que apenas o atributo *codpedido* é uma chave estrangeira e ao mesmo faz parte da chave primária. Assim sendo, não obedece a regra de multiplicidade de chaves estrangeiras, nem o requisito de toda a chave primária ser chave estrangeira, o que faz com que *pediditens* represente uma entidade fraca (Figura 23).

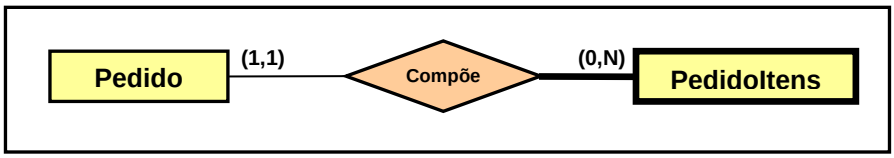


Figura 23. Exemplo de tabelas implementando Entidades Forte e Fraca

4.2.2 Identificação de Relacionamentos

Toda a chave estrangeira que não faz parte de uma chave primária composta por múltiplas chaves estrangeiras e nem é toda ela uma chave primária, representa um relacionamento 1:N ou 1:1. A regra não permite definir se a cardinalidade do relacionamento é 1:N ou 1:1. Para definir a cardinalidade desses relacionamentos é necessário verificar os possíveis conteúdos do banco de dados. No exemplo citado, todos os relacionamentos que não se encaixam nas regras 1 e 2 podem ser considerados relacionamentos 1:N, porém, nem sempre essa regra é válida. Na Figura 24 são demonstrados os relacionamentos identificados no banco de dados do exemplo citado

anteriormente.

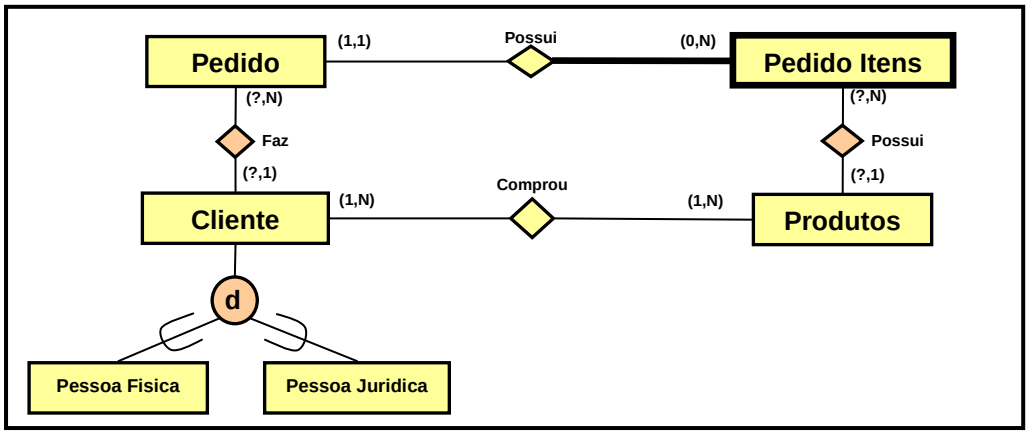


Figura 24. Identificação dos Relacionamentos

Ao analisar a Figura 24, pode-se verificar um ponto de interrogação nos relacionamentos onde existem mais de uma possibilidade para sua a cardinalidade. Para solucionar este problema, o sistema de testes implementado deve interagir com o usuário para que o mesmo auxilie na classificação deste relacionamento.

4.2.3 Definição de Atributos

No modelo E-R, as colunas das tabelas que fazem parte de uma chave estrangeira não são representadas como atributos. Os atributos das entidades são todas as colunas que não sejam chaves estrangeiras, assim como representado no modelo resultante desta etapa, na Figura 25.

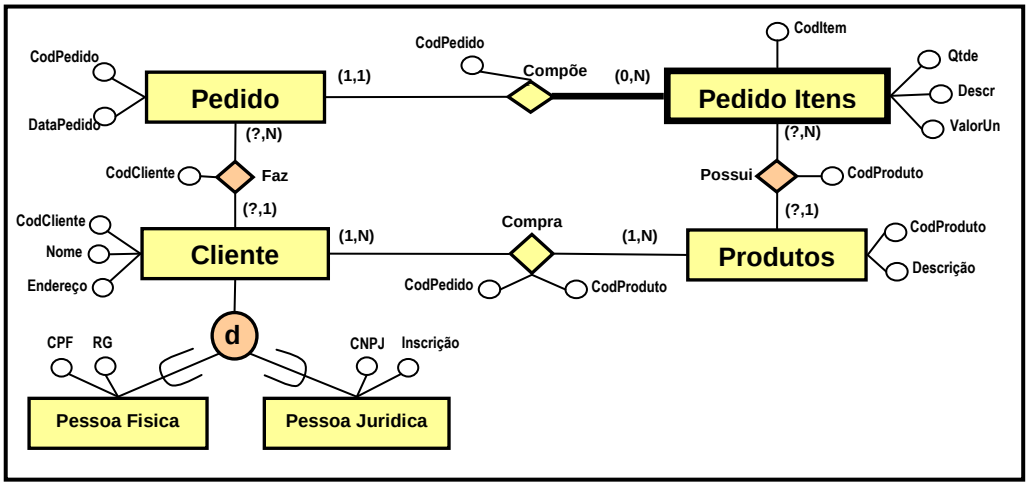


Figura 25. Definição dos Atributos correspondentes as Entidades e Relacionamentos

4.2.4 Definição dos Atributos Identificadores

Para definir os atributos identificadores de uma entidade, Heuser (2001) apresenta as seguintes regras:

a) **coluna da chave primária que não é chave estrangeira**: toda coluna que faz parte da chave primária e que não é chave estrangeira corresponde a um atributo identificador da entidade ou relacionamento;

b) **coluna da chave primária que é chave estrangeira**: toda coluna que faz parte da chave primária e que é uma chave estrangeira corresponde a um identificador externo da entidade.

Sendo assim, após a identificação dos atributos identificadores de cada entidade e relacionamento, pode-se obter como resultado da engenharia reversa, o modelo E-R mostrado na Figura 26.

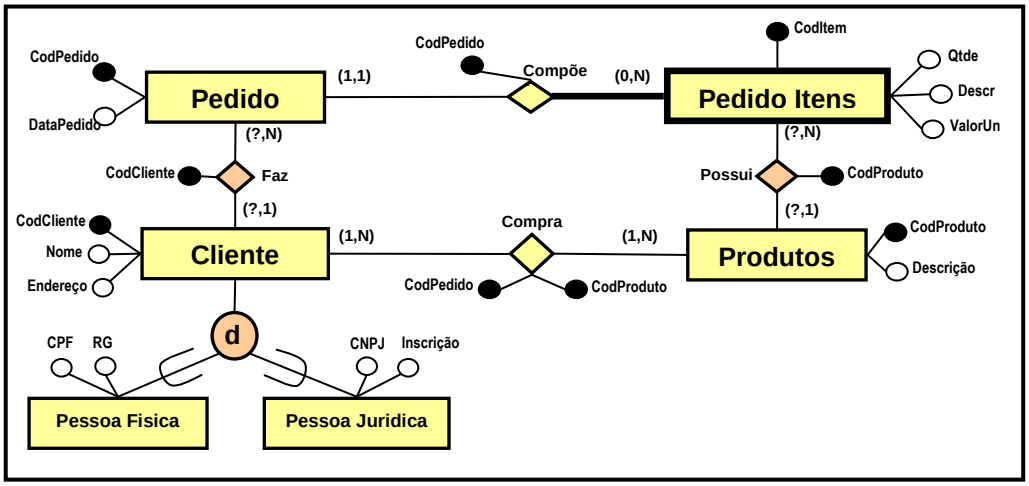


Figura 26. Definição dos Atributos Identificadores

Na Figura 26 é apresentado o modelo E-R do banco de dados exemplo, obtido a partir da metodologia de engenharia reversa proposta por Heuser (2001). Este modelo apresenta também os conceitos de modelo E-R Estendido, utilizado para representar especialização/generalização, conforme Elmasri e Navathe (2002).

O capítulo subseqüente apresenta dois trabalhos na área de engenharia reversa de bancos de dados relacionais, nos quais foram utilizados métodos diferentes para mapear os modelos de dados.

5 TRABALHOS CORRELATOS

Neste capítulo serão apresentados dois trabalhos relacionados à engenharia reversa de bancos de dados relacionais com metodologias diferentes e aplicadas para diferentes SGBD. O primeiro realiza a engenharia reversa do SQL Server, através do script SQL gerado pelo SGBD e o segundo realiza a engenharia reversa para o Firebird, através da extração do catálogo de dados por meio de uma conexão direta com o banco de dados.

5.1 FERRAMENTA DE APOIO À ENGENHARIA REVERSA DE UM BANCO DE DADOS RELACIONAL

A ferramenta descrita por Nobre e Souza (2005), realiza a engenharia reversa de um banco de dados relacional de forma semi-automática, interagindo com o usuário quando necessário. Esta interação é realizada através de uma interface que é apresentada ao usuário quando ocorrem situações em que mais de uma resposta é possível. Ela utiliza como ponto de partida apenas o *script* SQL, gerado pela exportação do esquema físico de um banco de dados criado com o SQL Server, sem a necessidade de conexão com o mesmo. A ferramenta apresenta como resultado final ao usuário, o esquema E-R gerado graficamente.

5.2 IMPLEMENTAÇÃO DE ENGENHARIA REVERSA DE BANCO DE DADOS RELACIONAIS NA FERRAMENTA LEARNING

O objetivo do trabalho desenvolvido por Ferreira (2205), é a implementação do processo de engenharia reversa na ferramenta LEaRning. Esta é uma ferramenta para projeto conceitual de banco de dados, desenvolvida com o auxílio do Programa de Bolsas de Iniciação Científica (PROBIC). Este programa possibilita o mapeamento do modelo E-R para o modelo relacional e através de sua interface gráfica, o usuário pode definir o esquema conceitual no modelo E-R. O LEaRning adota a notação do modelo E-R, conforme encontrado na literatura, mas inicialmente não realizava o processo de engenharia reversa. A partir da extração de informações do catálogo de dados do SGBD Firebird, ele devolve para o usuário o esquema conceitual, seguindo a notação encontrada na literatura para o modelo E-R.

6 SISTEMA EXPERIMENTAL DE ENGENHARIA REVERSA

O sistema desenvolvido neste trabalho tem como objetivo auxiliar a engenharia reversa de dois SGBD distintos: o Oracle e o Firebird, efetuando consultas ao dicionário de dados dos mesmos. Para que isso aconteça, é necessário que o SGBD forneça informações sobre a sua estrutura e que seja possível realizar uma comunicação com a base de dados.

6.1 DICIONÁRIO DE DADOS

Um sistema de banco de dados relacional precisa armazenar informações sobre as relações existentes no banco de dados. Estas informações ficam armazenadas no dicionário de dados ou catálogo de dados (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

O dicionário de dados pode ser considerado o coração do banco de dados, pois é nele que se encontram todas as informações que o SGBD necessita para trabalhar corretamente. O dicionário armazena inclusive sua própria estrutura. Entre as informações contidas nele podem ser citadas:

a) nomes das tabelas;

b) nomes dos atributos de cada tabela;

c) domínios e tamanho dos atributos de cada tabela;

d) nomes das visões e as definições dessas visões;

e) definições das restrições de integridade.

O dicionário de dados descreve os dados contidos no banco de dados. Esta descrição torna o banco de dados independente, pois é possível determinar a sua

estrutura e conteúdo independente da aplicação, usando para isto o próprio banco de dados.

Os dados existentes no dicionário de dados são denominados metadados e ficam armazenados dentro do banco de dados, assim como os dados do usuário, conforme mostra a Figura 27 (ELMASRI; NAVATHE, 2005).

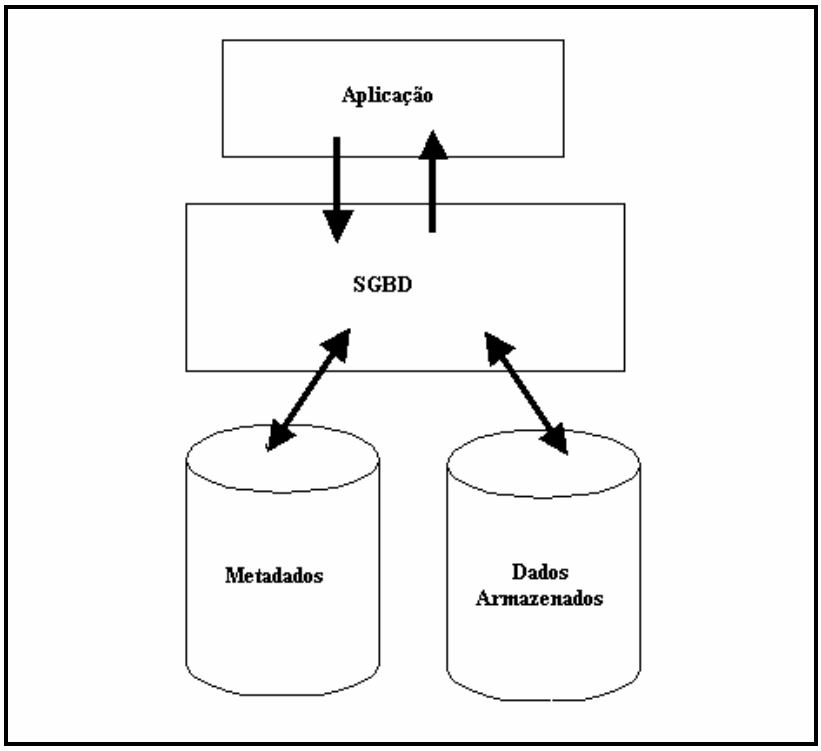


Figura 27. Sistema de Banco de Dados
Fonte: ELMASRI, R.; NAVATHE, S. (2005)

Os metadados constituem o elemento essencial para a realização da engenharia reversa. O próprio catálogo de dados constitui um banco de dados relacional, sendo assim, o processo de engenharia reversa pode ser feito por meio de consultas SQL a esse banco de dados.

Cada SGBD possui um dicionário de dados com estrutura distinta, e por esta razão, esta ferramenta não contempla todos os bancos de dados relacionais existentes hoje. Sendo assim, foram selecionados dois SGBD para realizar a engenharia reversa. Os bancos de dados selecionados para criar essa ferramenta foram o Oracle e o Firebird.

O Oracle, por ser o banco de dados mais utilizado atualmente nos servidores das grandes empresas e o Firebird por ser um software de código aberto e também considerado um SGBD estável e seguro.

6.1.1 Oracle

O Oracle surgiu no final dos anos 70, quando Larry Ellison identificou uma oportunidade que outras companhias não haviam percebido. Ellison encontrou uma descrição de um protótipo funcional de um banco de dados relacional e percebeu que havia um enorme potencial de negócios neste modelo de banco de dados. Até então, não havia nenhum produto comercial de banco de dados relacional no mercado (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

Ellison e os co-fundadores da Oracle Corporation, Bob Miner e Ed Oates, fundaram então a Oracle e decidiram criar um sistema de gerenciamento de banco de dados relacional como um produto comercial, tornando-a pioneira no mercado e posteriormente uma das maiores empresas de software empresarial do mundo (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

O SGBD da Oracle atualmente é líder do mercado, sendo que Oracle 9i também foi pioneiro no suporte ao modelo *web*. Além do banco de dados, a Oracle desenvolve diversas ferramentas de desenvolvimento, que juntas formam a Oracle *Developer* Suíte, utilizada na construção de programas de computador que interagem com a sua base de dados. A Oracle também criou a linguagem de programação PL/SQL, utilizada no processamento de transações e criação de formulários (ORACLE, 2007).

A versão do Oracle utilizada para realizar este projeto foi a versão 10g Express Edition (XE). Esta é uma versão gratuita do e está disponível para download no

site da Oracle em: www.oracle.com/technology/software/products/database/xe/.

O dicionário de dados do Oracle é formado por um conjunto de tabelas especiais criadas no usuário *SYS*, que têm como função registrar todas as informações de todos os objetos criados no banco de dados. Para ter acesso às informações, podem-se acessar as visões internas do Oracle, que são separadas por três grupos: USER_*, ALL_* e DBA_* (KREINES, 2003).

Para cada grupo existe uma função diferente:

a) **USER_***: exibe todos os objetos criados pelo usuário/esquema;

b) **ALL_*:** lista todos os objetos associados ao esquema atual e aos outros esquemas que o usuário tem autorização;

c) **DBA_*:** este grupo tem a função de exibir todos os objetos do banco de dados, independente do esquema em que o objeto foi criado;

Assim como os grupos, existem as *views* ou tabelas especiais, onde o Oracle armazena as informações do software, que será a fonte de consulta como mostra a Tabela 2.

Tabela 2. Dicionário de Dados do Oracle

VIEWS	**DESCRIÇÃO**
OBJECTS	Exibe todos os objetos do banco de dados como views, tabelas, índices, etc.
TABLES	Exibe informações de todas as tabelas do banco de dados.
TAB_COLUMNS	Exibe detalhes das colunas de todas as tabelas do banco de dados.
TAB_PARTITIONS	Exibe as informações sobre as partições das tabelas.
INDEXES	Exibe informações de todos os índices.
IND_COLUMNS	Fornece as informações exatas sobre as colunas que são chaves dos índices, nome, tabela, localização, tamanho e etc.
OBJECT_TABLES	Fornece todas as informações sobre os objetos da tabela relativas à utilização dentro do banco de dados.
METHOD_RESULTS	Exibe a descrição dos resultados dos tipos de dados.
IND_PARTITIONS	Mostra informações dobre partições de índices.
CONSTRAINTS	Exibe todas as regras de integridade de todas as tabelas e colunas.
CONS_COLUMNS	Mostra as colunas que fazem parte das restrições.
SEQUENCES	Listagem de todos os objetos *sequences* criados no banco.
SYNONYMS	Mostra todos os *synonyms* criados na banco de dados.
USERS	Exibe informações precisas de todos os usuários do banco de dados.
LOBS	Permite saber se a *lob* está com índice, em qual tabela e o nome do segmento, etc.
VIEWS	Exibe informações de todas as *visões* do banco de dados.
UPDATABLE_COLUMNS	Permite saber se as colunas de uma tabela podem ser alteradas, inseridas ou excluídas.
TYPES	Exibe todos os tipos de dados criados no banco de dados.
TYPE_ATTRS	Exibe todos os atributos criados para os tipos de dados.
TYPE_METHODS	Descreve os métodos criados aos tipos de dados.
NESTED_TABLES	Mostra a relação entre tabelas *Nested* e a tabela no qual está relacionada.
METHOD_PARAMS	Lista todos os parâmetros associados a cada método dos tipos de dados.

Fonte: KREINES (2003)

6.1.2 Firebird

O Firebird é originado do código fonte do Interbase 6 Open-Source. Por este motivo, pode-se dizer que seu desenvolvimento iniciou em meados de 1981 e desde então vem sofrendo diversas alterações. Em 1986 recebeu o nome InterBase, iniciando na versão 2.0 quando passou a ser um produto comercial da Borland Software Corporation (FIREBIRD, 2008).

O Firebird foi desenvolvido por uma equipe de programadores voluntários e independentes, ou seja, sem nenhuma ligação entre si e sem o patrocínio de empresas. O código fonte do Interbase 6 *Open-Source* foi liberado pela Borland em 25 de julho de

2000. A partir da versão 6.0, o Interbase deixou de ser *open-source*. Dessa forma o Firebird foi desenvolvido como uma resposta da comunidade de software livre à empresa Borland (FIREBIRD, 2008).

Este SGBD é conhecido por ser um banco de dados estável, seguro e robusto. A sua instalação ocupa pouco espaço em disco, quando comparado com outros SGBD disponíveis no mercado e pode ser considerado um software de pequeno porte.

O armazenamento dos dados ocorre todo em um único arquivo, ou seja, o conjunto de dados, procedimentos ligados ao banco, gatilhos disparados automaticamente, visões e outros objetos que constituem o banco de dados, ficam armazenados em um único arquivo de dados.

O dicionário de dados do Firebird pode ser resumido em poucas tabelas, como se pode visualizar na Tabela 3.

Tabela 3. Dicionário de Dados do Firebird

VIEWS	DESCRIÇÃO
RDB$RELATIONS	Exibe os dados das tabelas do banco de dados
RDB$VIEW_RELATIONS	Exibe o nome de cada visão criada pelo usuário
RDB$TYPES	Exibem os domínios dos atributos e seus respectivos tipos
RDB$FIELDS	Exibe o domínio de cada atributo;
RDB$RELATION_FIELDS	Exibe o nome dos campos e suas respectivas tabelas.
RDB$REF_CONSTRAINTS	Exibe o nome de cada restrição de chave estrangeira e o nome de cada restrição de chave primária que o atributo chave estrangeira referencia.
RDB$RELATION_CONSTRAINTS	Exibem as restrições de chave primária e chave estrangeira, seu respectivo índice e a respectiva tabela que cada restrição pertence.
RDB$INDEX_SEGMENTS	Cada índice representa um atributo ou um conjunto de atributos, esta view exibe o nome de cada índice e o respectivo atributo associado a este índice.

Fonte: FERREIRA, J. (2005)

6.2 RECURSOS UTILIZADOS

A ferramenta de testes foi codificada na linguagem de programação Java. A escolha dessa linguagem se deu pela sua grande disseminação nos últimos tempos. O ambiente de desenvolvimento Java, utilizado para o desenvolvimento do sistema, foi o Netbeans 6.0.1, disponível gratuitamente para download no endereço

http://download.netbeans.org/netbeans/6.0/final/. O Netbeans é um ambiente integrado de desenvolvimento (IDE), que permite ao programador criar programas utilizando recursos gráficos. Para trabalhar com o Netbeans é necessário ter instalado anteriormente uma das versões do Java Development Kit (JDK), que também pode ser obtido gratuitamente na internet, no endereço http://java.sun.com/.

A geração do modelo E-R foi implementada utilizando uma biblioteca de recursos gráficos, denominada *draw2D*. Esta é utilizada para a criação de componentes gráficos juntamente com a *Standard Widget Toolkit* (SWT). A biblioteca *draw2D* vem juntamente com as bibliotecas do *Graphical Editing Framework* (GEF) e está disponível no endereço www.eclipse.org/gef/.

No desenvolvimento da interface do sistema, foi utilizada uma biblioteca denominada *SWT*, que é uma camada de componentes sobre os componentes padrões do sistema operacional, ou seja, um botão criado com o *SWT*, é na verdade, um botão do sistema operacional que está executando, então, se for aplicado um tema que mude a cor ou a forma do sistema, este vai se modificar conforme o novo tema.

Aplicações feitas utilizando componentes *SWT* sempre têm a aparência do sistema operacional em que elas executam, pois elas utilizam diretamente estes componentes para gerar a visualização na tela. A escolha desses componentes se deu, pois, as bibliotecas padrão do Netbeans (*AWT* e *Swing*) não apresentaram compatibilidade com a biblioteca *draw2D*, de modo que não foi possível adicionar o componente gráfico, gerado com a *draw2D*, dentro de uma interface criada com os componentes da biblioteca *Swing* e *AWT*. O *SWT* está disponível para *download* no endereço http://www.eclipse.org/swt/.

A conexão da aplicação com os SGBD, utiliza a Java Database Connectivity (JDBC), que é um conjunto de classes Java que podem ser usadas no desenvolvimento

de aplicações que necessitam comunicar-se com bancos de dados. Ela possui uma classe denominada *Driver*, que pode ser considerada como uma ponte para a origem do banco de dados (CADENHEAD; LEMAY, 2005). Esta biblioteca pode ser encontrada no website da Sun Microsystens no endereço http://java.sun.com/products/jdbc/download.html/.

6.3 SISTEMA DE TESTES IMPLEMENTADO

O sistema de testes desenvolvido tem como objetivo fazer a extração de informações do dicionário de dados, e a partir deste, realizar a engenharia reversa de um banco de dados relacional, obtendo como resultado final o modelo conceitual. Este processo pode ser mais bem compreendido a partir do diagrama de atividade mostrado na Figura 28.

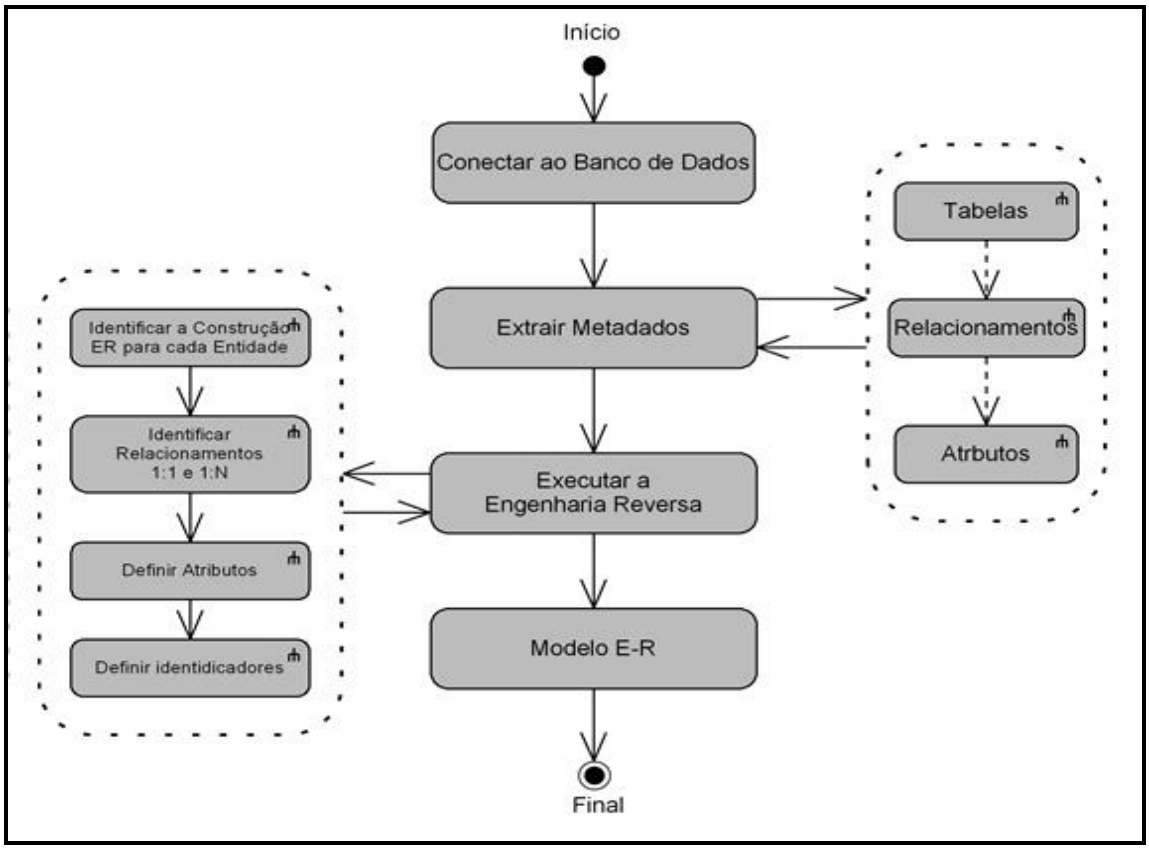


Figura 28: Diagrama de Atividade para a Engenharia Reversa

O diagrama de atividade mostrado na Figura 28 apresenta as etapas executadas pelo sistema para realizar a engenharia reversa. Podem-se destacar a extração de metadados e execução da engenharia reversa como as principais etapas do processo. Essas etapas foram subdivididas em etapas menores, para uma melhor compreensão de todo o processo. A extração dos metadados ocorre em três etapas distintas, executadas separadamente: a extração das tabelas, relacionamentos e atributos. Para cada uma delas é executada uma consulta ao dicionário de dados, e o resultado desta é armazenado em memória. A execução da engenharia reversa foi subdividida em quatro etapas: a identificação da construção E-R correspondente a cada tabela, a identificação dos relacionamentos 1:1 e 1:N, a definição dos atributos e definição dos identificadores de cada entidade, conforme apresentado por Heuser (2001) e descrito no capítulo 4.2.

6.3.1 Extração do Dicionário de Dados

A extração do dicionário de dados se dá por meio de consultas SQL diretamente no banco de dados. Para que isso seja possível, o sistema realiza uma conexão com o banco, de modo que ele possa ter acesso ao dicionário de dados do mesmo.

Primeiramente, deve-se configurar uma conexão ODBC especificando o banco de dados no qual se deseja realizar a engenharia reversa. Para realizar a conexão, foi criada na tela de abertura do sistema, uma interface de conexão para Oracle e Firebird na qual o usuário deve selecionar o SGBD e preencher os campos com o nome do usuário administrador, senha e o nome da conexão ODBC, previamente configurada, conforme a Figura 29. No Oracle, um mesmo arquivo de dados pode armazenar vários

esquemas diferentes, cada esquema é armazenado em um usuário denominado *owner* (proprietário). Para realizar a engenharia reversa de cada esquema de dados no Oracle, é necessário estar conectado com o usuário proprietário deste esquema. No Firebird, cada arquivo de dados armazena um único esquema de dados.

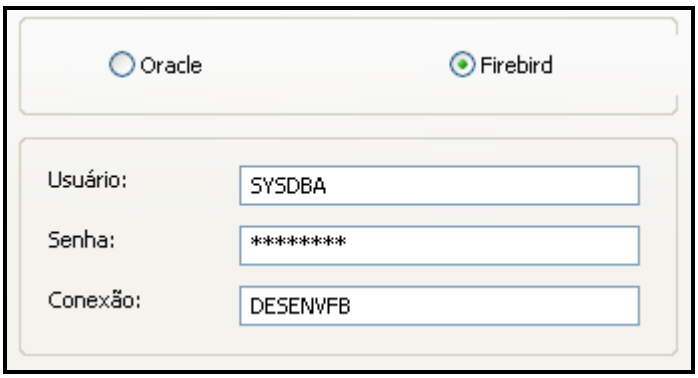


Figura 29. Interface para Conexão com o Banco de Dados

A Figura 29 mostra a interface de entrada dos dados para conexão com o banco, onde se podem verificar na parte superior, as opções Oracle e Firebird, e abaixo, os campos para informar o usuário, senha e a conexão.

A Figura 30 mostra a parte do código em que o sistema atribui os dados de conexão solicitados e realiza a conexão com o banco.

```
public boolean conectar(){
    try {
        conectado = true;
        Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
        String url = "jdbc:odbc:"+nomeConexao;
        conn = DriverManager.getConnection( url, usuario, senha );
    } catch ( ClassNotFoundException e ) {
        JOptionPane.showMessageDialog(null, "Erro: Problemas durant
        conectado = false;
    } catch ( SQLException e ) {
        JOptionPane.showMessageDialog(null, "Erro: Problemas durant
        conectado = false;
    }

    return conectado;
}
```

Figura 30. Parte do código para realizar a conexão com o banco de dados

Depois de conectado ao banco de dados, o sistema deverá mapear o dicionário de dados para as classes de armazenamento. Para isso, o sistema realiza três consultas SQL distintas.

A primeira consulta SQL realizada tem como objetivo fazer o mapeamento de todas as tabelas do esquema de dados. A sintaxe das consultas SQL realizadas para cada SGBD pode ser visualizada na Tabela 4.

Tabela 4. Consultas SQL para listar as Tabelas

Oracle	Firebird
`SELECT TABLE_NAME FROM USER_TABLES;`	`SELECT RDB$RELATION_NAME FROM RDB$RELATIONS WHERE RDB$FLAGS IS NOT NULL;`

No Oracle foi realizada uma consulta na view *user_tables*, que, como apresentado anteriormente, irá exibir apenas as tabelas do esquema de dados pertencentes ao usuário, que está conectado ao banco de dados. Já no Firebird, foi utilizada a tabela *rdb$relations* que, por sua vez, exibe todas as tabelas do banco de dados. Por este motivo, houve a necessidade de adicionar a cláusula *where*, para que a consulta retornasse apenas os registros em que o campo *rdb$flags* fosse diferente de nulo, caso contrário, essa consulta retornaria inclusive as tabelas usadas pelo SGBD para armazenar o dicionário de dados do Firebird entre outras tabelas do SGBD, o que não seria relevante para a engenharia reversa.

Na figura 31 pode ser visualizado a parte do código fonte, em que o sistema faz a leitura das tabelas.

```
/*******************************
 ******* LE AS ENTIDADES ******
 *******************************/
String lsSql =dicionario.getTabelas();
try{
    Statement stmt = conn.createStatement();
    ResultSet results = stmt.executeQuery( lsSql );
    String nomeEntidade = null;

    while ( results.next() ) {
        nomeEntidade = results.getString(1);
        lb_progress.setText("Extraindo Tabelas: "+nomeEntidade+"...");
        entidades.add(new Entidade(nomeEntidade));
    }

    stmt.close();
}catch(SQLException e){
    new MensagemErro(shell,"Erro:"+String.valueOf(e.getErrorCode()),e.getMessage());
}
```

Figura 31. Parte do código que faz a extração das tabelas

O código apresentado na Figura 31 primeiramente busca a consulta correspondente ao SGBD no qual está sendo realizada a engenharia reversa, por meio da função *getTabelas()* da classe *dicionario*. Esta função retorna uma *String* contendo o *script* SQL para o respectivo SGBD. Posteriormente, o sistema executa a busca SQL e percorre os registros retornados pela consulta através da estrutura de repetição *while*. A cada registro que a consulta retorna, o sistema cria um novo objeto da classe *Entidade*, e armazena em um vetor de objetos denominado *entidades*.

A segunda consulta realizada é para a extração dos relacionamentos, como pode ser visualizado na Tabela 5.

Tabela 5. Consultas SQL para listar os relacionamentos

Oracle	Firebird

`select constraint_name,`
`table_name,`
`(select c.table_name`
`from user_constraints c`
`where c.constraint_name =`
`uc.r_constraint_name`
`and rownum=1)`
`from user_constraints uc`
`where constraint_type = 'R';`
	`select fk.rdb$constraint_name,`
	`fk.rdb$relation_name,`
	`(select tb.rdb$relation_name`
	`from rdb$relation_constraints tb`
	`where tb.rdb$constraint_type`
	`in ('PRIMARY KEY','UNIQUE')`
	`and tb.rdb$constraint_name=pk.rdb$const_name_uq)`
	`from rdb$relation_constraints fk,`
	`rdb$ref_constraints pk`
	`where fk.rdb$constraint_type = 'FOREIGN KEY'`
	`and fk.rdb$constraint_name = pk.rdb$constraint_name`

As consultas dos relacionamentos buscam os nomes das referências, nome da tabela que está referenciando e o nome da tabela referenciada, respectivamente, conforme mostrado na Tabela 5. Para realizar a consulta no Oracle, foi utilizada apenas a view *user_constraints*, que retorna o nome da referência e o nome da tabela que a mesma pertence, onde o atributo *constraint_type* seja igual a "**R"**. Para buscar o nome da tabela referenciada, foi necessário uma sub-consulta na própria view *user_constraints*, pois no registro da mesma existia apenas o nome da chave primária que ela referenciava, e não o nome da tabela referenciada. Já para o Firebird, foi necessário fazer uma junção entre as *views rdb$relation_constraints* e *rdb$ref_constraints*, assim como a sub-consulta para buscar o nome da tabela referenciada.

```
/*******************************
 *** LE OS RELACIONAMENTOS ****
 *******************************/
lsSql = dicionario.getRelacionamentos();
try{
    Statement stmt = conn.createStatement();
    ResultSet results = stmt.executeQuery( lsSql );
    String nomeRelacionamento = null;
    String nomeEntidadeOrigem = null,nomeEntidadeDestino=null;

    while ( results.next() ) {
        nomeEntidadeOrigem = results.getString(1);
        nomeRelacionamento = results.getString(2);
        nomeEntidadeDestino= results.getString(3);

        Relacionamento rel = new Relacionamento(nomeRelacionamento);
        rel.setOrigem(nomeEntidadeOrigem, "?");
        rel.setDestino(nomeEntidadeDestino , "?");
        relacionamentos.add(rel);
    }
    conn.commit();
}catch(SQLException e){
    new MensagemErro(progress.getShell(),"Erro:"+String.valueOf(e.getErrorCode
}
```

Figura 32. Parte do código que faz a extração dos relacionamentos

Na Figura 32 é apresentada parte do código fonte, em que o sistema realiza a extração dos relacionamentos. O processo de extração dos relacionamentos é praticamente o mesmo processo realizado para a extração das tabelas, porém, a consulta

realizada é a consulta apresentada na Tabela 5, que retorna três valores: o nome do relacionamento, da entidade de destino e da entidade de origem, a qual pertence o relacionamento. Estes valores são armazenados em objetos do tipo *Relacionamento*, e são adicionados ao vetor *relacionamentos*.

A próxima consulta que o sistema executa é a extração dos atributos das tabelas e dos relacionamentos. Esta consulta deverá retornar os seguintes valores:

a) nome do atributo;

b) se o atributo é ou não chave primária (‘S’ para sim e NULL para não);

c) se o atributo é ou não chave estrangeira (‘S’ para sim e NULL para não);

d) nome da chave estrangeira;

e) nome da tabela referenciada.

Para obter todos esses valores em uma única consulta SQL, foi necessário criar várias subconsultas, como se pode verificar na Tabela 6. Essa consulta é executada uma vez para cada tabela existente no banco de dados e com ela é possível obter todas as informações necessárias para classificar os atributos.

Tabela 6. Consultas SQL para listar os atributos

SGBD	Consulta SQL

ORACLE	`Select column_name,`
	`(select 's' from user_ind_columns i`
	`where i.table_name = t.table_name`
	`and i.column_name = t.column_name`
	`and i.index_name = ( select c.index_name from user_constraints c`
	`where c.table_name = t.table_name`
	`and c.constraint_type = 'p')),`
	`(select 's' from user_cons_columns c`
	`where c.table_name = t.table_name`
	`and c.column_name = t.column_name`
	`and instr(c.constraint_name,'FK_') > 0`
	`and rownum = 1) ,`
	`(select cons.constraint_name`
	`from user_constraints cons, user_cons_columns c`
	`where cons.constraint_name = c.constraint_name`
	`and c.table_name = t.table_name`
	`and c.column_name = t.column_name`
	`and instr(c.constraint_name,'FK_') > 0`
	`and rownum = 1),`
	`(select (select cons2.table_name`
	`from user_constraints cons2`
	`where cons2.constraint_name=cons.r_constraint_name )`
	`from user_constraints cons, user_cons_columns c`
	`where cons.constraint_name = c.constraint_name`
	`and c.table_name = t.table_name`
	`and c.column_name = t.column_name`
	`and instr(c.constraint_name,'FK_') > 0`
	`and rownum = 1)`
	`from user_tab_columns t where t.table_name = :nome_da_tabela`

FIREBIRD	`Select rdb$field_name,`
	`( select 'S' from rdb$index_segments i`
	`where i.rdb$field_name = f.rdb$field_name`
	`and i.rdb$index_name in ( select r.rdb$index_name`
	`from rdb$relation_constraints r`
	`where r.rdb$relation_name = f.rdb$relation_name`
	`and rdb$constraint_type IN ('PRIMARY KEY','UNIQUE'))),`
	`( select 'S' from rdb$index_segments i`
	`where i.rdb$field_name = f.rdb$field_name`
	`and i.rdb$index_name in ( select r.rdb$index_name`
	`from rdb$relation_constraints r`
	`where r.rdb$relation_name = f.rdb$relation_name`
	`and r.rdb$constraint_name in`
	`(select ref.rdb$constraint_name`
	`from rdb$ref_constraints ref ))),`
	`( select rdb$index_name from rdb$index_segments i`
	`where i.rdb$field_name = f.rdb$field_name`
	`and i.rdb$index_name in ( select r.rdb$index_name`
	`from rdb$relation_constraints r`
	`where r.rdb$relation_name = f.rdb$relation_name`
	`and r.rdb$constraint_name in`
	`(select ref.rdb$constraint_name`
	`from rdb$ref_constraints ref))) ,`
	`( select (select rdb$relation_name`
	`from rdb$relation_constraints`
	`where rdb$constraint_type IN ('PRIMARY KEY','UNIQUE')`
	`and rdb$constraint_name = r.rdb$const_name_uq)`
	`from rdb$index_segments i, rdb$ref_constraints r`
	`where i.rdb$field_name = f.rdb$field_name`
	`and i.rdb$index_name = r.rdb$constraint_name`
	`and i.rdb$index_name in (select r.rdb$index_name`
	`from rdb$relation_constraints r`
	`where r.rdb$relation_name = f.rdb$relation_name`
	`and r.rdb$constraint_name in`
	`(select ref.rdb$constraint_name`
	`from rdb$ref_constraints ref)))`
	`from rdb$relation_fields f`
	`where f.rdb$relation_name = :nome_da_tabela`

Na Figura 33 é apresentada uma parte do código fonte para a extração dos atributos, e conforme os valores retornados nos campos PK e FK, esses já são inseridos na entidade a qual pertencem como chave primária, chave estrangeira, chave primária e estrangeira simultaneamente ou simplesmente como um atributo. Se o atributo é uma chave estrangeira, devem ser informados, além de seu nome, o nome da tabela a qual o

mesmo está relacionado (*nomeEntidadeRel*) e o nome do relacionamento (*nomeForeignKey*). As informações *nomeEntidadeRel* e *nomeForeignKey* tem como objetivo identificar se a chave primária da entidade em questão é composta toda ela de uma única chave estrangeira, de várias chaves estrangeiras ou nenhuma.

```
private Entidade buscaAtributos(Entidade asEnt){
    String nomeEntidade =  null,nomeAtributo = null,nomeForeignKey = null,nomeEntidad
    Entidade ent = asEnt;
    nomeEntidade = ent.getNomeEntidade();
    String lsSql = dicionario.getColunas(nomeEntidade);
    try{Statement stmt = conn.createStatement();
        ResultSet results = stmt.executeQuery( lsSql );
        while ( results.next() ) {
            nomeAtributo = results.getString(1);
            PK = results.getString(2);
            FK = results.getString(3);
            nomeForeignKey = results.getString(4);
            nomeEntidadeRel = results.getString(5);
            if (PK!=null){
                if (FK!=null){
                    ent.addAtributoPKFK(nomeAtributo,nomeEntidadeRel,nomeForeignKey);
                }else{
                    ent.addAtributoPK(nomeAtributo);}
            }else{
                if (FK!=null){
                    ent.addAtributoFK(nomeAtributo,nomeEntidadeRel,nomeForeignKey);
                }else{
                   ent.addAtributo(nomeAtributo);
                }
            }
        }
        stmt.close();
    }catch(SQLException e){
```

Figura 33. Parte do código para a extração dos atributos

Para que o sistema pudesse manipular estes dados, foram criadas as classes internas para armazenar e processar essas informações. As principais classes desenvolvidas foram: *Entidade*, *Atributos* e *Relacionamento*. Estas classes são usadas para armazenar os dados obtidos por meio das consultas SQL, apresentadas anteriormente nas Tabelas 4, 5 e 6. O diagrama de classes que representa essas estruturas pode ser visualizado na Figura 34.

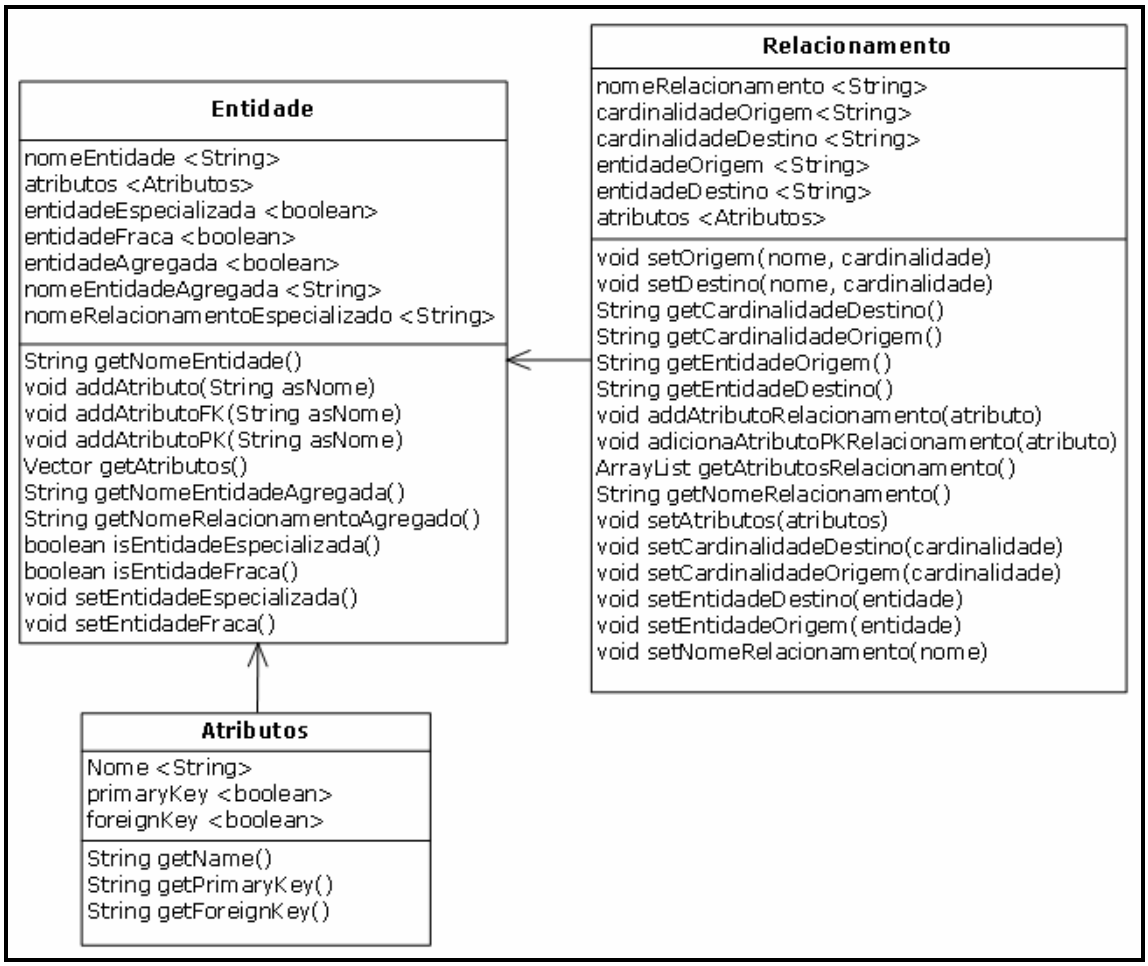


Figura 34. Diagrama de Classes - Estruturas de Armazenamento dos Metadados

A classe *Entidade*, apresentada na Figura 34, tem como objetivo armazenar os dados das tabelas encontradas no esquema de dados. A classe *Relacionamento* armazena os dados das chaves estrangeiras, bem como o nome das tabelas de origem e destino e a cardinalidade do relacionamento. A classe *Atributos* armazena as informações dos atributos de cada tabela e de cada relacionamento. Cada tabela pode possuir vários atributos, assim como cada relacionamento também pode conter vários atributos.

6.3.2 Identificação da Construção E-R Correspondente a cada Tabela

Esta é a etapa em que se inicia o processo da engenharia reversa propriamente dita. Após a extração do dicionário de dados, o sistema deverá fazer o processamento das tabelas do banco de dados classificando-as em entidades, relacionamentos ou entidades especializadas, conforme as regras definidas por Heuser (2001) mostrado no capítulo 4.3.1. Esta etapa aplica o algoritmo apresentado na Figura 35, para identificar o que cada tabela deverá representar.

```
PARA cada tabela (t)
   SE toda chave primária composta de múltiplas chaves estrangeiras
      ENTÃO t representa um relacionamento N:N entre as
            tabelas referenciadas por t;
   SENÃO
      SE toda a chave primária implementa uma única chave estrangeira
         ENTÃO t representa uma entidade especializada;
      SENÃO
         SE a chave primária for composta de uma única chave estrangeira
            porém nem toda a chave primária for uma chave estrangeira
            ENTÃO t representa uma entidade fraca;
         SENÃO
            t representa uma entidade;
         FIM SE;
      FIM SE;
   FIM SE;
FIM PARA;
```

Figura 35. Algoritmo resumido para realizar a engenharia reversa de banco de dados

O algoritmo apresentado na Figura 35 percorre todas as tabelas previamente armazenadas na lista *entidades*, analisando a composição das chaves primária e estrangeira, e irá definir o que cada tabela deve representar no modelo E-R. Se a superchave da tabela em questão for toda ela composta por mais de uma chave estrangeira, ela representará um relacionamento muitos para muitos (N:N). Se toda a superchave for composta de uma única chave estrangeira, então ela representará uma entidade especializada, se apenas parte da superchave for composta de uma única chave estrangeira, essa tabela representará uma entidade fraca, e se nenhuma das situações

acima for verdadeira, essa tabela representará uma entidade forte, conforme a metodologia apresentada no capítulo 4.2.1.

6.3.3 Identificação dos Relacionamentos 1:1 e 1:N

Identificar os tipos e cardinalidade dos relacionamentos é o próximo passo da engenharia reversa. Nesta etapa, o sistema cria os relacionamentos, ligando a entidade de origem e a entidade referenciada, assim como também define a cardinalidade da origem e do destino do relacionamento. O que irá definir a cardinalidade de um relacionamento é a composição da chave primária das entidades relacionadas e quais os campos fazem parte da chave estrangeira. Em alguns casos, poderá haver mais de uma possibilidade, então o sistema deverá interagir com o usuário para que o mesmo possa informar a cardinalidade deste relacionamento, conforme apresentado na Figura 36.

Figura 36. Tela de interação com o usuário

Conforme apresentado no capítulo 4.2.2, não existe uma regra exata para definir a cardinalidade mínima e máxima de alguns tipos de relacionamento. A interface mostrada na Figura 36 é apresentada ao usuário toda vez em que ocorre um relacionamento 1:1 ou 1:N. Nesta interface, o usuário tem a opção de informar a

cardinalidade mínima e máxima do relacionamento, tanto para a entidade de origem quanto para a entidade de destino.

6.3.4 Definição de Atributos e Identificadores das Entidades

A definição dos atributos e dos identificadores é a última etapa realizada na construção do modelo E-R. Porém, no sistema implementado, essa etapa é executada juntamente com a identificação da construção E-R de cada tabela. A cada entidade identificada é realizada uma busca no dicionário de dados, que retorna os atributos e os identificadores das mesmas. Foi criado um laço percorrendo as entidades adicionando uma a uma seus atributos, exceto aqueles atributos que sejam chave estrangeira, pois, tradicionalmente em um modelo E-R, os atributos que são apenas chave estrangeira não devem aparecer dentro da entidade como um atributo, mas sim como um identificador externo.

6.3.5 Apresentação do Modelo E-R Gerado

Ao final da engenharia reversa, o sistema apresenta ao usuário o diagrama E-R do banco de dados especificado. Nas Figuras 37, 38 e 39 são apresentados alguns exemplos de modelos de dados gerados pela ferramenta de testes durante o desenvolvimento da mesma.

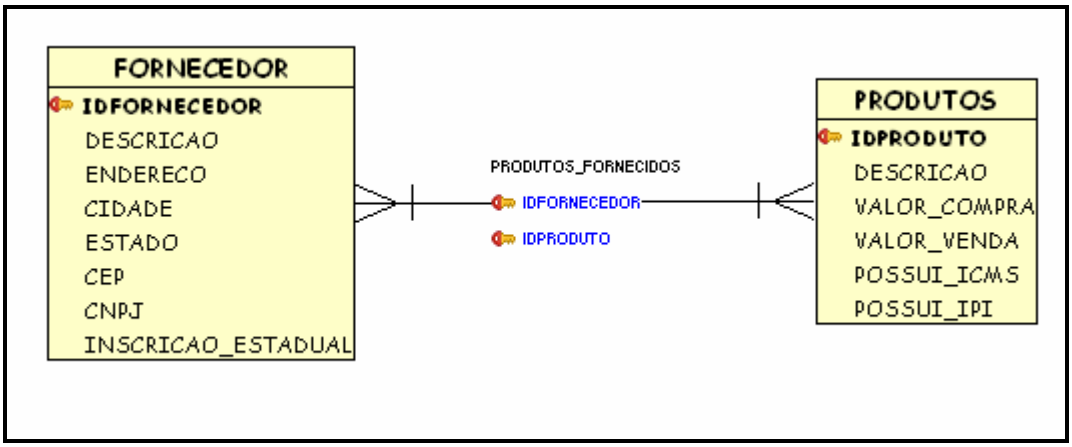


Figura 37. Representação de Relacionamento N:N no Sistema de Testes

A Figura 37 apresenta o resultado da engenharia reversa de um banco de dados, no qual existem três tabelas: fornecedores, produtos e produtos_fornecidos. Nesse exemplo, a tabela produtos_fornecidos representa um relacionamento N:N entre as tabelas fornecedores e produtos, pois toda a sua superchave é composta de mais de uma chave estrangeira.

A representação de um relacionamento especializado é feita por meio de um círculo entre as entidades relacionadas, como pode ser visualizado na Figura 38.

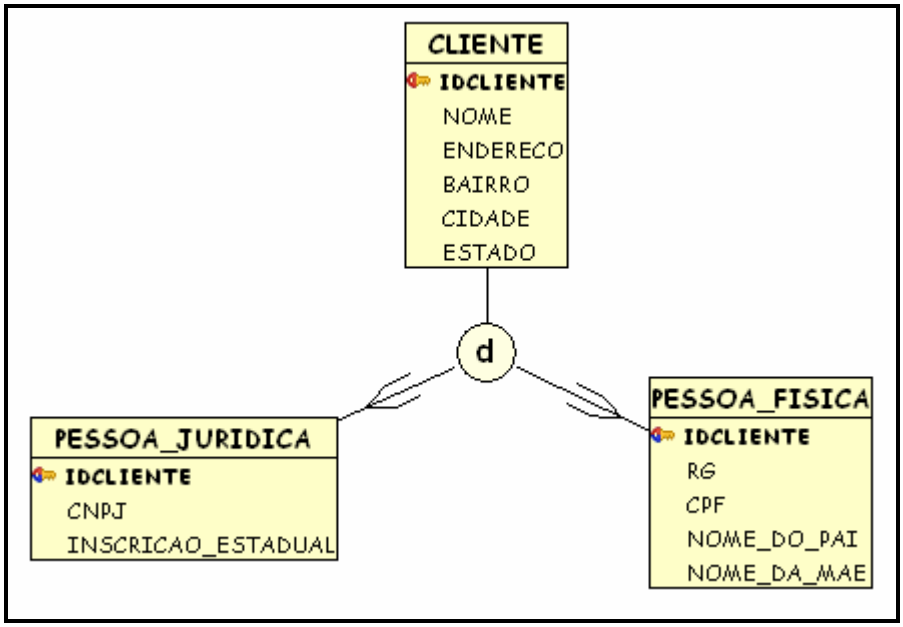


Figura 38. Representação de Entidade Especializada no Sistema de Testes

Para representar uma entidade fraca, o sistema utiliza um retângulo com a borda mais espessa que uma entidade normal (Figura 39). Neste caso, o sistema classificará automaticamente a cardinalidade deste relacionamento como 1:N (um-para-muitos).

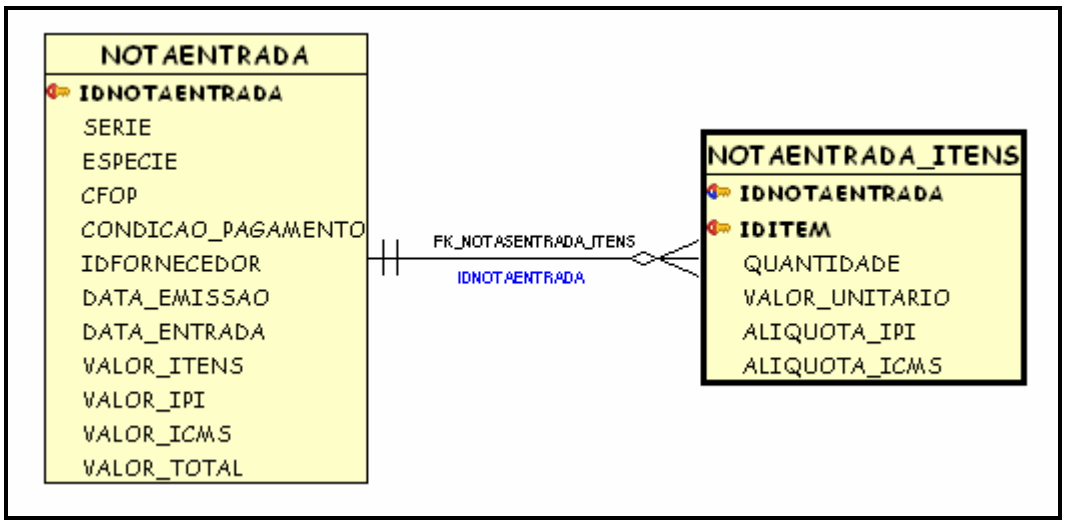


Figura 39. Representação de Entidade Fraca no Sistema de Testes

6.3.6 Interface

O sistema desenvolvido possui uma interface simples e fácil de ser utilizada. Foram criados três menus, nos quais estão disponibilizadas todas as funcionalidades do sistema: menu Arquivo, com as opções de abrir, salvar e imprimir; menu Processos, com as opções de extrair metadados e gerar modelo E-R; e menu Ajuda que exibe informações sobre o sistema. Também foi criado uma barra de ferramentas, logo abaixo do menu, com todas essas opções para facilitar o acesso do usuário, provendo uma melhor usabilidade ao sistema.

Figura 40. Barra de Ferramentas

A Figura 40 mostra a barra de ferramentas, na qual são disponibilizadas todas as funcionalidades que o sistema oferece. A opção (a) permite abrir um arquivo com extensão “*.der” contendo o modelo E-R salvo anteriormente pelo sistema. O botão (b) tem a função de salvar o modelo E-R, armazenando em um arquivo com a extensão *.der. O botão (c) imprime o modelo E-R. O botão (d) tem a função de conectar ao sistema ao banco de dados, conforme os dados de conexão fornecidos pelo usuário e iniciar e extração do dicionário de dados. O botão (e) inicia o processo de engenharia reversa com base nos metadados previamente extraídos e gera o modelo E-R. O botão (f) exibe informações sobre o sistema, e por fim, o botão (g) finaliza o sistema.

Abaixo da barra de tarefas estão localizadas as três abas, onde podem ser visualizados os dados de conexão, os metadados e o diagrama ER, respectivamente. A primeira aba contém os campos necessários para conectar-se ao banco de dados, conforme a Figura 29 apresentada no Capítulo 6.3.1.

A segunda aba é apresentada após a extração dos metadados. Ela exibe uma lista das tabelas na forma de árvore, onde podem visualizadas todas as colunas de cada tabela, bem como suas chaves estrangeiras.

Ao expandir cada tabela contida na árvore clicando no sinal de adição (+), são apresentados os nomes dos atributos e as referências pertencentes à mesma. Ao clicar em cima do nome da tabela, é apresentada uma lista à direita com os nomes dos campos da mesma, entre outras informações, por exemplo, se o atributo é chave estrangeira e/ou chave primária e o nome da tabela que o mesmo referencia, como mostra a Figura 41.

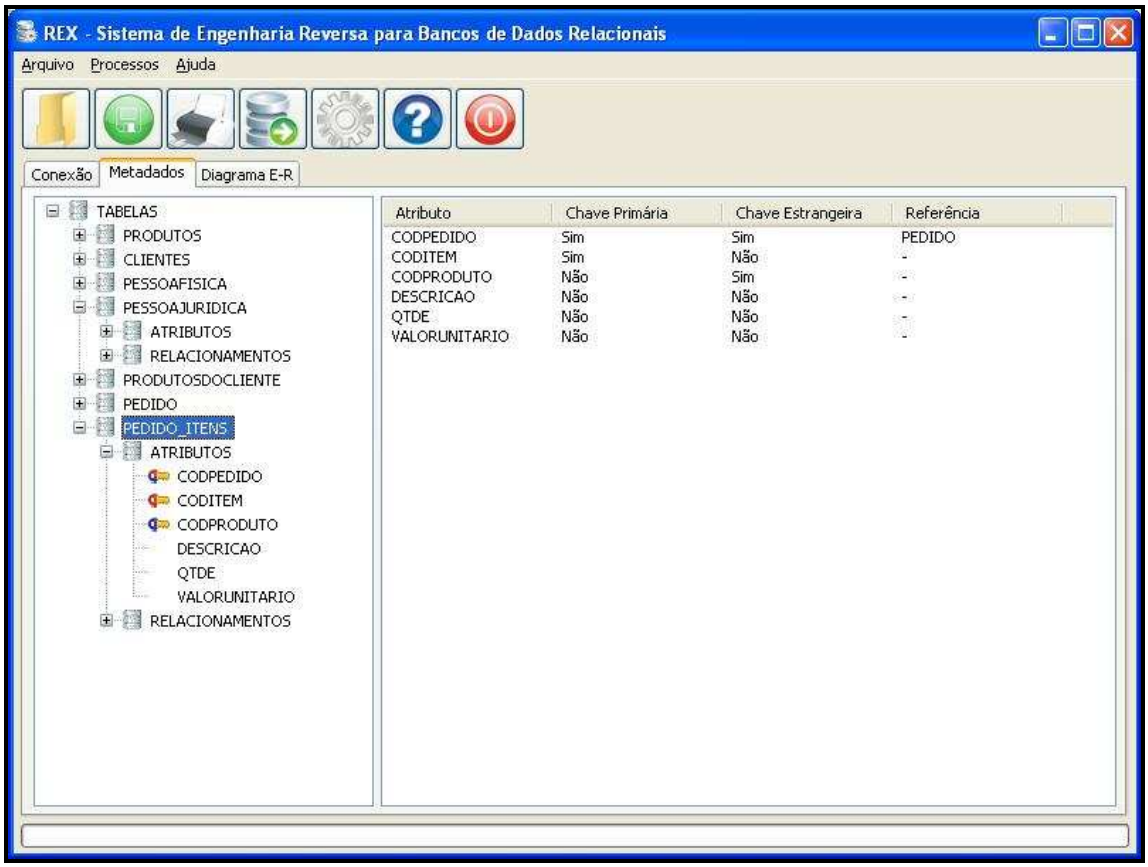


Figura 41. Tela de Apresentação dos Metadados.

Os dados exibidos na Figura 41 são referentes ao esquema físico do banco de dados, ou seja, sem a aplicação do algoritmo para a engenharia reversa.

A terceira aba exibe o modelo de dados conceitual ou modelo E-R, resultante da engenharia reversa, conforme a Figura 42. Este diagrama é exibido após a aplicação do algoritmo para a transformação do modelo lógico em conceitual.

Na parte inferior do sistema é exibida uma barra de progresso, que é utilizada durante a extração do dicionário de dados e a aplicação do algoritmo de engenharia reversa, conforme mostrado na Figura 42. A barra de progresso apresenta ao usuário um *feedback*, mostrando a situação real da execução do algoritmo.

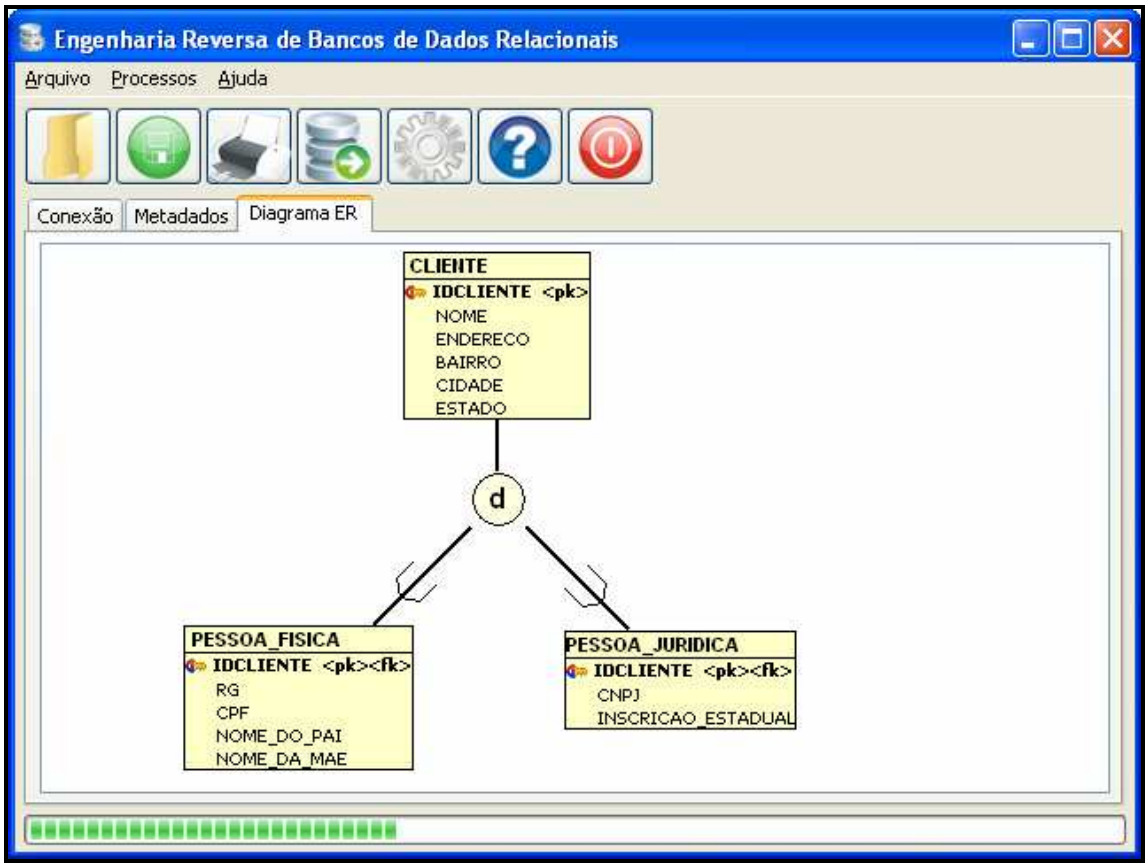


Figura 42. Tela de Apresentação do Diagrama E-R

6.4 ANÁLISE DOS RESULTADOS OBTIDOS

Durante a fase de pesquisa do trabalho surgiram muitas dificuldades. A falta de livros publicados na área de engenharia reversa para bancos de dados foi uma das principais. Esta, por sua vez, foi superada somente através de pesquisas, principalmente de artigos, trabalhos acadêmicos e monografias publicadas, nas quais foram encontrados trabalhos sobre o assunto.

No desenvolvimento da aplicação também foram encontradas algumas dificuldades, sendo as principais: dificuldade para encontrar bibliotecas para modelagem de dados e a incompatibilidade dessas com as bibliotecas padrão do Netbeans: *AWT e Swing*. Por meio de pesquisas na internet e em outros trabalhos acadêmicos, foi encontrada a biblioteca *draw2D*, que permitiu a criação de

componentes gráficos para a modelagem de dados. O problema de compatibilidade foi solucionado com a substituição da biblioteca *Swing*, que é a biblioteca padrão do Netbeans, por outra biblioteca denominada *SWT*, a qual foi descrita no capítulo 6.2. Esta apresentou compatibilidade com a biblioteca *draw2D*, permitindo a visualização dos componentes gráficos de modelagem, dentro da janela da aplicação.

No processo de engenharia reversa também foram encontradas algumas dificuldades, por exemplo, o mapeamento da cardinalidade dos relacionamentos 1:1 e 1:N, descrita no capítulo 4.2.2. Este problema foi solucionado fazendo-se a interação do usuário com o sistema durante a engenharia reversa, permitindo que o analista auxilie na classificação da cardinalidade dos relacionamentos durante o processo de mapeamento. Esta interação permite que o analista possa corrigir erros de modelagem durante a engenharia reversa, obtendo desta forma, um modelo de dados mais elaborado.

Encontrou-se também, dificuldades com o posicionamento dos objetos no modelo de dados, pois, na primeira vez em que é realizada a engenharia reversa, as entidades ficam espalhadas na tela sem uma ordem coerente. Sendo assim, o usuário precisa posicionar os objetos que compõem o diagrama, de forma que ele possa visualizar o modelo E-R da maneira mais compreensível e transparente.

Outro desafio no desenvolvimento foi a exportação do modelo gerado para diferentes tipos de mídia, por exemplo: imagens, XML e PDF entre outras, sendo que a alternativa encontrada para a superação deste foi a serialização dos objetos para arquivo permitindo que o mesmo pudesse ser visualizado e impresso posteriormente sem a necessidade de refazer o processo de engenharia reversa.

Para realizar os testes no sistema, foram utilizados pequenos bancos de dados, desenvolvidos em Firebird pelos alunos da disciplina de Banco de Dados I, do

curso de Ciência da Computação da UNESC. Nos testes executados, a engenharia reversa obteve sucesso na maioria dos bancos de dados utilizados, sendo que em alguns bancos foram encontrados erros no mapeamento da superchave, pois no Firebird, uma superchave pode ser criada como sendo uma *Primary Key*, ou então, apenas como uma *Unique Constraint*. Inicialmente, o sistema considerava como superchave apenas as *Constraints* do tipo *Primary Key*. Portanto, em alguns testes o sistema não funcionou corretamente, fazendo com que as tabelas fossem mapeadas sem a superchave e os relacionamentos fossem classificados de forma incorreta. Para corrigir este problema foi adicionado na cláusula *where* da consulta SQL dos atributos e relacionamentos, o tipo *Unique*, como pode ser visualizado nas Tabelas 5 e 6 do Capítulo 6.3.1.

Como resultado final dos testes obteve-se os diagramas E-R apresentados nos Apêndices B, C e D. No Apêndice B são exibidos os diagramas E-R de um banco de dados para um sistema de administração de cinemas, onde o primeiro diagrama é modelo original gerado por meio da ferramenta de modelagem DB Designer e o segundo é o modelo gerado pela engenharia reversa do sistema de testes, implementado neste trabalho. Do mesmo modo, são apresentados nos Apêndices C e D, os diagramas E-R para um banco de dados de uma empresa de eventos e festas e para um banco de dados para o gerenciamento de vendas.

Após o término dos testes, pode-se concluir que o objetivo geral e os objetivos específicos, deste trabalho, foram satisfatoriamente alcançados, uma vez que a ferramenta de testes desenvolvida realizou a engenharia reversa de bancos de dados em Oracle e Firebird, apresentando como saída o modelo E-R.

CONCLUSÃO

O desenvolvimento de modelos de dados conceituais, durante a concepção de projetos de bancos de dados, é fundamental para garantir a qualidade do software que está sendo desenvolvido, assim como permitir a continuidade de seu desenvolvimento.

O objetivo geral deste trabalho foi alcançado somente a partir da total compreensão dos objetivos específicos. Primeiramente, foi realizado o estudo sobre o desenvolvimento de projetos de bancos de dados, assim como a compreensão dos conceitos de engenharia reversa. Foi necessário também, o estudo da estrutura do dicionário de dados dos SGBD Oracle e Firebird e das ferramentas de modelagem existentes atualmente.

O desenvolvimento do trabalho foi subdividido em etapas para uma melhor organização. As principais etapas foram: levantamento bibliográfico, estudo sobre projetos de bancos de dados, modelo E-R, engenharia reversa, pesquisa do algoritmo a ser aplicado, estudo dos SGBD Oracle e Firebird, modelagem do sistema, implementação prática e testes.

Um dos principais desafios encontrados durante a etapa de levantamento bibliográfico, foi suprir a escassez de material didático na área de engenharia reversa de bancos de dados, através de pesquisas em outros trabalhos acadêmicos, artigos e monografias publicadas.

Durante a etapa de desenvolvimento também foram encontradas algumas dificuldades, por exemplo, a incompatibilidade entre as bibliotecas de componentes visuais *AWT* e *Swing* com a biblioteca de modelagem *draw2D*. Para superá-la, foi necessário um estudo da linguagem de programação Java e das bibliotecas existentes para a criação de interfaces gráficas. Por meio dessa pesquisa foi selecionada a

biblioteca de componentes visuais *SWT*, a qual apresentou ótima compatibilidade com a biblioteca *draw2D*.

Outro desafio na implementação da ferramenta, foi o mapeamento dos relacionamentos 1:1 e 1:N. Como não existe uma regra exata para a identificação desses relacionamentos, foi implementada uma forma de interação com o usuário, para que o mesmo pudesse auxiliar na classificação desses relacionamentos durante o processamento da engenharia reversa.

Durante os testes da ferramenta foram encontrados alguns erros no mapeamento do dicionário de dados, por exemplo, erros nas consultas SQL, que foram imediatamente solucionados.

O sistema de testes desenvolvido permite que os usuários possam obter o modelo E-R de um banco de dados relacional desenvolvido em Oracle ou Firebird de forma semi-automática, ou seja, interagindo com o usuário quando necessário.

O desenvolvimento desta ferramenta de código fonte aberto, poderá contribuir com a comunidade acadêmica no estudo de projetos de bancos de dados e permitirá a continuação desta pesquisa.

Sendo assim, pode-se concluir que os objetivos específicos e consequentemente, o objetivo geral deste trabalho foram alcançados, uma vez que o sistema de testes desenvolvido realizou a engenharia reversa de bancos de dados implementados em Oracle e Firebird, gerando como saída o modelo de dados conceitual.

Considerando os resultados obtidos, podem-se destacar algumas sugestões para trabalhos futuros:

a) desenvolver um algoritmo para permitir o melhor posicionamento dos objetos que compõe o modelo E-R de forma automatizada, facilitando a visualização do mesmo;
b) desenvolver meios de exportação do modelo E-R para outras mídias, por exemplo: PDF, XML, SQL Script, JPEG, BMP entre outras;
c) adaptar a ferramenta para suportar outros SGBD;
d) adicionar outras funcionalidades ao sistema, por exemplo, permitir ao usuário/analista modificar graficamente o modelo E-R, aprimorando-o e posteriormente fazer o caminho inverso à engenharia reversa, que corresponde a geração do esquema lógico do banco de dados.

REFERÊNCIAS

BRAGA, Rosana Teresinha Vaccare**. Padrões de software a partir da engenharia reversa de sistemas legados**. 1998. 132 f. Dissertação (Mestrado em Matemática Computacional). Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, São Carlos.

CADENHEAD, Roger; LEMAY, Laura. **Aprenda em 21 dias Java 2**. 4. ed. Rio de Janeiro: Campus, 2005.

CHIKOFSKI, Eliot J.; CROSS, James H. **Reverse Engineering and Design Recovery**: *a Taxonomy.* IEEE Software. 1990.

DATE, C. J. **Introdução a sistemas de bancos de dados**. 4. ed. Rio de Janeiro: Campus, 2004.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de bancos de dados:** Fundamentos e aplicações. 3. ed. Rio de Janeiro: LTC, 2002.

FABULOUS FORCE DATABASE TOOLS. **DB Designer 4:** Online Documentation Version 1.0.41, 2003. Disponível em: <http://www.fabforce.net/dbdesigner4/doc/index.html>. Acesso em: 10 jun. 2007.

FERREIRA, Janaina L. **Implementação de Engenharia Reversa de Banco de Dados Relacionais na Ferramenta LEaRning**. 2005. 85 f. Monografia (Bacharelado em Sistemas de Informação). Pontifícia Universidade Católica de Minas Gerais, Arcos.

FIREBIRD Foundation. **About Firebird**. Disponível em: <http://www.firebirdsql.org/index.php?op=history >. Acesso em: 10 junho 2008.

HERNANDEZ, Michael J. **Aprenda a projetar seu próprio banco de dados.** São Paulo: Makron Books, 2000.

HEUSER, Carlos Alberto. **Projeto de banco de dados.** 4. ed. Porto Alegre: Sagra Luzzatto, 2001.

KREINES, C.David. **Oracle Data Dictionary Pocket Reference.** Sebastopol, CA : O'Reilly, 2003.

MACHADO**,** Felipe Nery Rodrigues; ABREU, Maurício Pereira de. **Projeto de banco de dados:** uma visão prática. 8. ed. São Paulo: Érica, 2002**.**

NOBRE, Miguel K.; SOUZA, Marcelo Caon. **Ferramenta de apoio a Engenharia Reversa de um Banco de Dados Relacional.** 2005. 54 f. Monografia (Bacharelado em Sistemas de Informação) Universidade Federal de Santa Catarina, Florianópolis.

ORACLE Database. **Oracle's History:** Innovation, Leadership, Results. Disponível em: <http://www.oracle.com/corporate/story.html>. Acesso em: 02 jun. 2007.

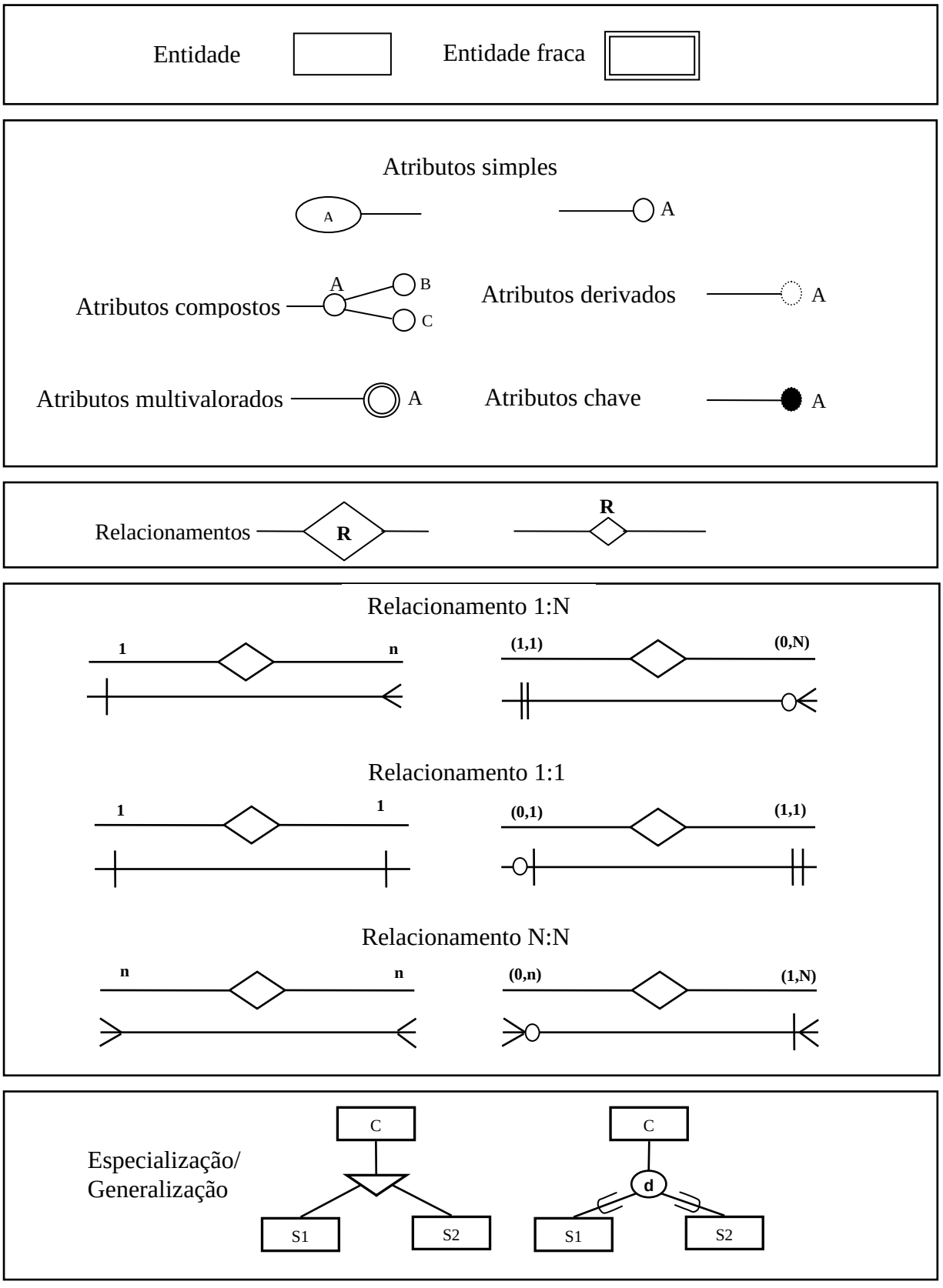
QUEST Software. **Toad Data Modeler**. Disponível em: <http://www.casestudio.com/enu/default.aspx> Acesso em: 03 jun. 2007.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados.** 5. ed. Rio de Janeiro: Elsevier, 2006.

SOUZA, Adriano Luiz de. **Portando Aplicações do Sistema Gerenciador de Banco de Dados Sybase para Firebird.** 2005. 84 f. Monografia (Pós-graduação MBA em gestão de banco de dados). Universidade do Extremo Sul Catarinense, Criciúma.

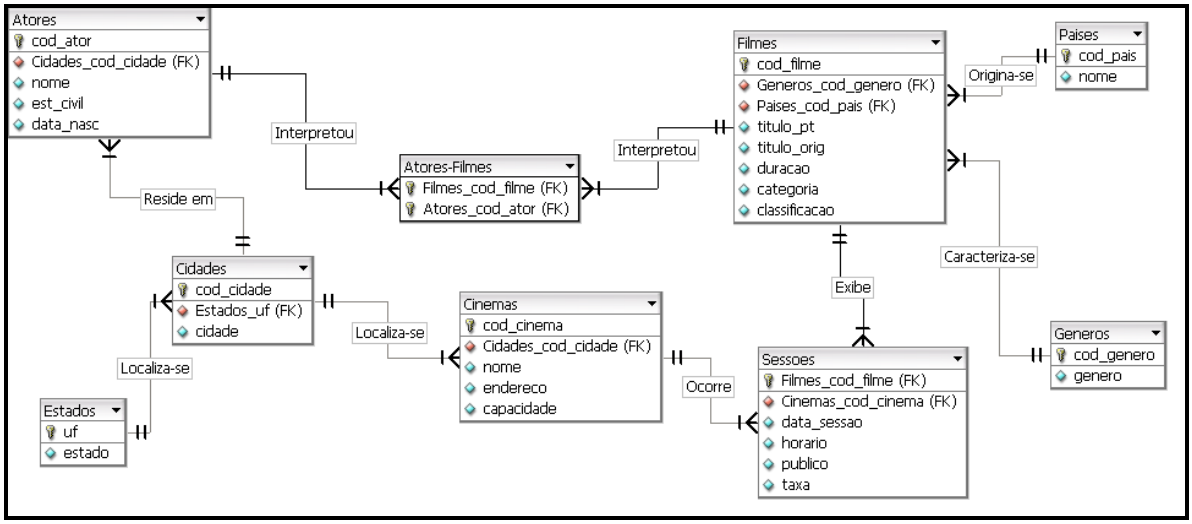
APÊNDICE A

Notações alternativas para criação de Diagramas Entidade-Relacionamento.

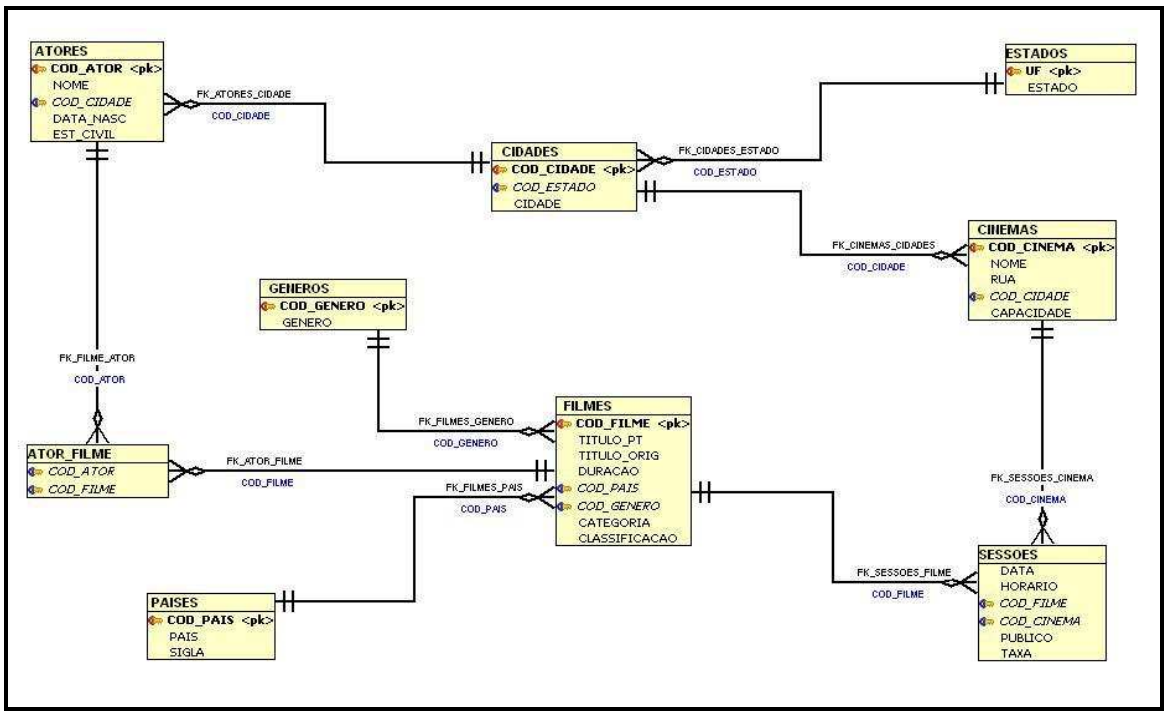


APÊNDICE B

Modelo E-R de um banco de dados para um sistema de administração de cinemas, desenvolvidos pelos alunos da disciplina de Banco de Dados I do curso de Ciência da Computação da UNESC.



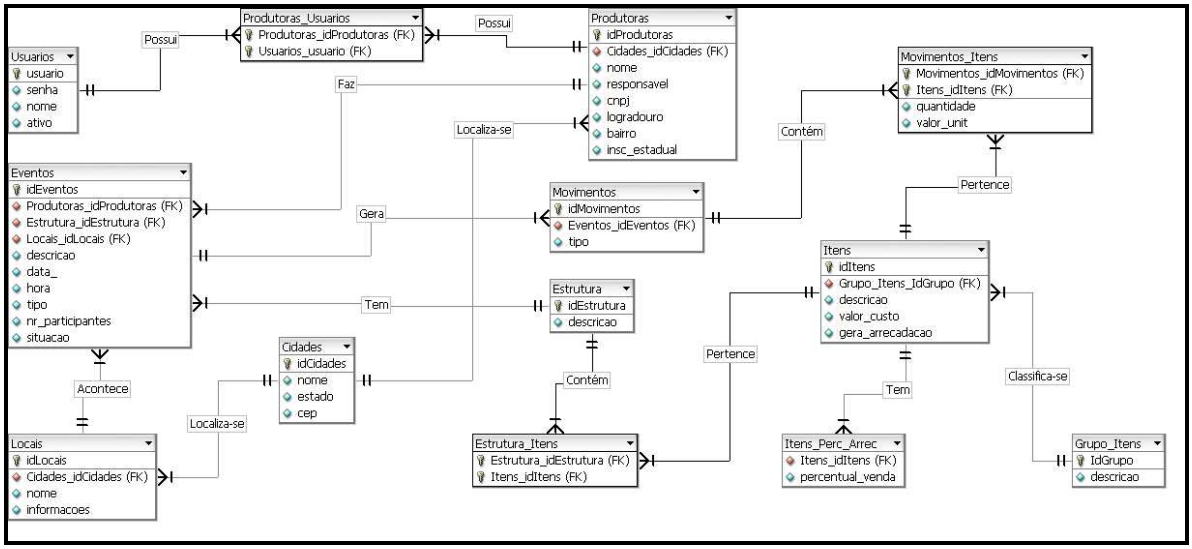
Modelo E-R elaborado no DB Designer



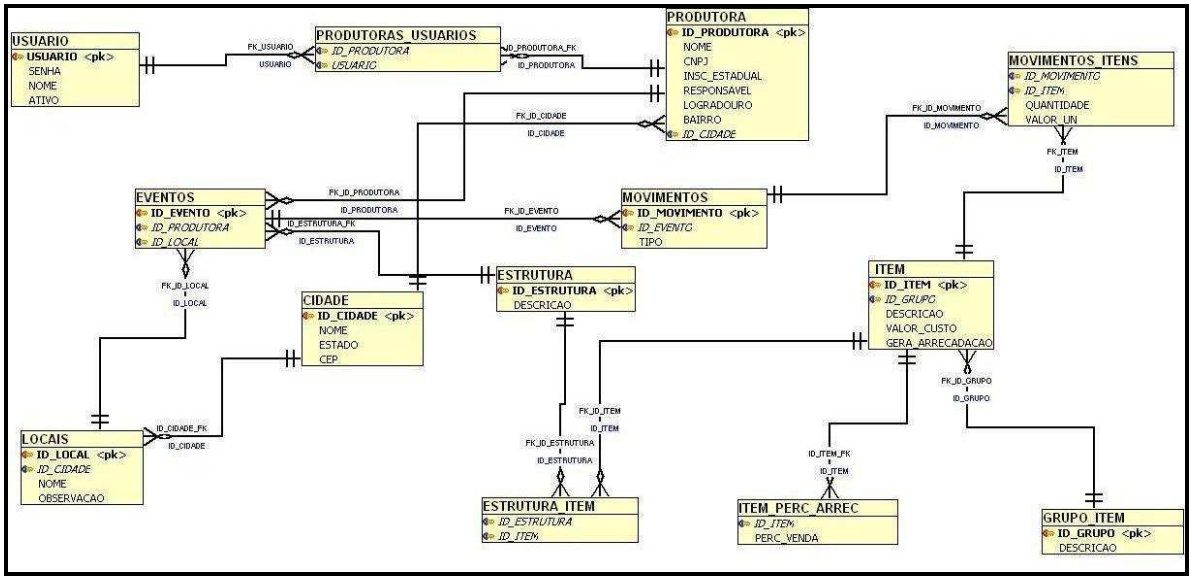
Modelo E-R resultante da engenharia reversa, realizada pelo sistema de testes implementado

APÊNDICE C

Modelo E-R de um banco de dados para um sistema de Eventos e Festas, desenvolvido pelos alunos da Disciplina de Banco de Dados I, do Curso de Ciência da Computação da UNESC.



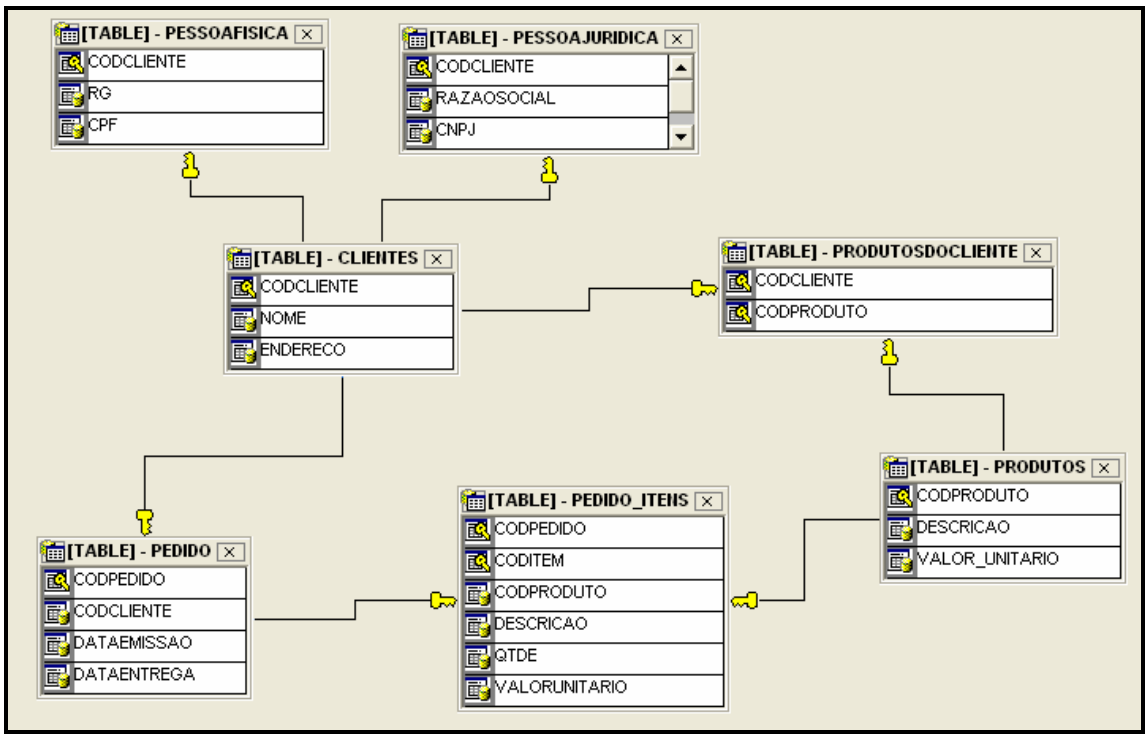
Modelo E-R elaborado no DB Designer



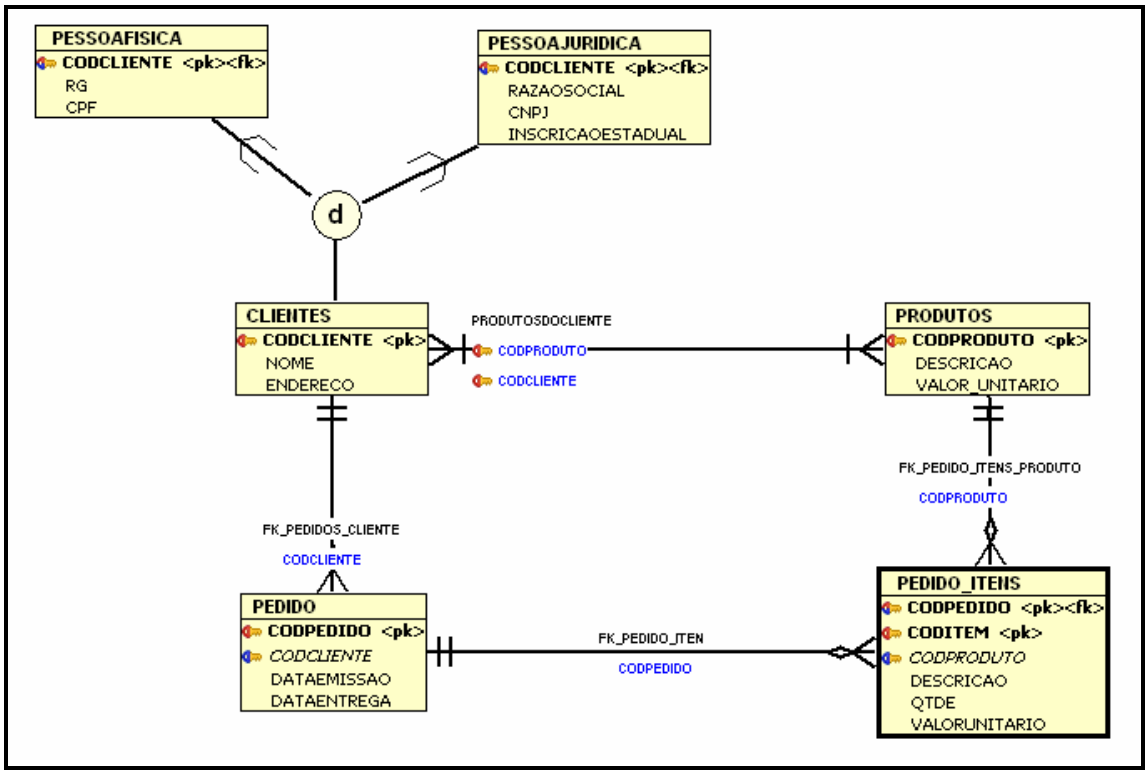
Modelo E-R resultante da engenharia reversa.

APÊNDICE D

Modelo de dados para um sistema de vendas, utilizado como exemplo, durante o desenvolvimento da ferramenta.



Modelo de dados gerado pelo Quick Desk



Modelo E-R resultante da engenharia reversa