

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE INDAIATUBA

GUSTAVO QUIRINO FERREIRA

**Desenvolvimento de um software para modelagem conceitual de
banco de dados**

Indaiatuba
Junho/2013

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE INDAIATUBA

GUSTAVO QUIRINO FERREIRA

**Desenvolvimento de um software para modelagem conceitual de
banco de dados**

Trabalho de Graduação do Curso Superior de
Tecnologia em Banco de Dados apresentado como
requisito para obtenção do título de tecnólogo,
elaborado sob a orientação da professora Maria das
Graças Junqueira Machado Tomazela.

Indaiatuba
Junho/2013

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE INDAIATUBA

GUSTAVO QUIRINO FERREIRA

Banca avaliadora:

Prof ^a Me. Maria das Graças J. M. Tomazela	Orientadora
Prof ^o Carlos César Farias de Souza	Professor convidado
Prof ^o Me. Antonio César de Barros Munari	Profissional da área

Data da defesa: 28/07/2013

AGRADECIMENTOS

À Prof. Mestra Maria das Graças Junqueira Machado Tomazela pela orientação durante todo o período do trabalho.

Ao Prof. Doutor Aldo Nascimento Pontes pelo acompanhamento e correções durante a fase de projeto do trabalho.

Ao Prof. João Manuel pelas ajudas relacionadas à cálculos geométricos utilizados no software.

Ao Prof. Sérgio Donisete Clauss pelo auxílio com dúvidas matemáticas.

RESUMO

No dia a dia, desenvolvedores e administradores de bancos de dados precisam ter conhecimento das estruturas das bases de dados que estão utilizando. Para isso, muitas vezes fazem uso do modelo conceitual de modelagem dos dados, tornando a compreensão do banco de dados mais simples e abstrata. Para realizar a modelagem conceitual de dados, muitas vezes os profissionais fazem uso de ferramentas que não são adequadas para a finalidade, ou que não lhes oferecem todos os recursos ou ainda, não é disponibilizada gratuitamente. Para suprir a falta de ferramentas que atendam à necessidade de modelagem de banco de dados, mais especificamente a modelagem conceitual, esse trabalho tem por objetivo propor e desenvolver uma ferramenta gratuita, que funcione em diversos sistemas operacionais que ofereçam condições técnicas para suportar a ferramenta. Para o desenvolvimento da ferramenta, foi escolhido tecnologias *open-source* e multiplataformas como a linguagem de programação e o formato de arquivo gerado ao salvar o diagrama do modelo conceitual de banco de dados com o intuito de garantir a portabilidade e permitir que o *software* possa ser usado em diversos sistemas operacionais. Também foi aplicado padrões de projetos e boas práticas de programação a fim de garantir legibilidade ao código-fonte e facilitar futuras manutenções por terceiros, já que há uma possibilidade de o código-fonte se tornar aberto futuramente.

Palavras-chave: modelagem conceitual, modelo entidade relacionamento, ferramenta de modelagem, software gratuito.

SUMÁRIO

INTRODUÇÃO

CAPÍTULO I.....	7
Fundamentação teórica.....	7
1. Modelagem conceitual.....	7
1.1.2. Entidade.....	7
1.1.3. Atributo.....	8
1.1.4. Cardinalidade.....	9
1.1.5. Relacionamento.....	10
1.1.6. Generalização/Especialização.....	11
1.1.6.1. Generalização total.....	12
1.1.6.2. Generalização parcial.....	13
1.1.6.3. Generalização compartilhada	13
1.2. Relação entre ferramentas disponíveis no mercado.	13
1.2.1. DBDesigner 4	14
1.2.2. CA ERwin Data Modeler r8.2	15
1.2.3. Xcase	16
1.2.4. Dezign for Databases V7.2.0	18
1.2.5. PowerDesigner	18
1.2.6. Enterprise Architect	19
1.2.7. brModelo	20
1.2.8. Visual Paradigm	21
1.2.8. Dr.CASE 3.50f	23
1.2.9. DbWrench Database Design and Synchronization	25
1.2.10. Astah	25
CAPÍTULO II.....	27
Materiais e métodos.....	27
2. Desenvolvimento.....	27
2.1. Internacionalização.....	29
2.2. Estrutura lógica.....	30
2.3. Estrutura de pacotes.....	31
2.4. Herança e polimorfismo.....	31
2.5. Algoritmo de ligação de atributos.....	32
2.6. Movimentação de componentes.....	34
2.7. Padrão de arquivo gerado pelo software.....	35
2.8. Recurso de desfazer e refazer.....	35
2.9. Algoritmo de ligação de atributo com relacionamentos.....	36
CAPÍTULO III.....	39
Apresentação dos resultados.....	39
3.1. A respeito das características.....	39
1.2. Apresentação do sistema.....	40
CONSIDERAÇÕES FINAIS.....	44
REFERÊNCIAS BIBLIOGRÁFICAS.....	45

INTRODUÇÃO

Profissionais que trabalham com desenvolvimento de software ou projetam banco de dados utilizam no seu dia a dia muitas ferramentas para modelagem de dados. Essas ferramentas auxiliam na criação de modelos conceituais da estrutura da base de dados, modelos lógicos e físicos. Além disso, o profissional também necessita de ferramentas que auxiliam de forma simplificada a manutenção dessas bases de dados.

A maioria das ferramentas de modelagem partem do modelo lógico para o físico como o dbWrench (DBWRENCH.COM, [s.d.]) outras vão do modelo conceitual ao lógico. Algumas ferramentas como o brModelo (CÂNDIDO, 2005) utilizam os três modelos, porém gera o modelo físico na forma padrão da linguagem SQL, ou seja, falta a opção de geração do modelo físico específico para cada banco. Além disso, a maioria das ferramentas não são gratuitas e operam somente na plataforma Windows.

A partir da situação problema, podemos levantar as seguintes questões: Como utilizar os conceitos de modelagem conceitual em uma ferramenta multiplataforma e que seja gratuita?

O objetivo desse trabalho é desenvolver um software que seja capaz de auxiliar o projetista de banco de dados a criar o modelo conceitual de banco de dados relacionais, utilizando uma ferramenta gratuita e multiplataforma.

O trabalho parte da pesquisa entre os grandes estudiosos e escritores da área de banco de dados (ELMASRI; NAVATHE, 2005; SILBERSCHATZ; KORT; SUDARSHAN, 2006), a fim de encontrar as variantes na forma de como é feita a modelagem conceitual. É notório que há uma grande variação entre as formas de diagramas do modelo conceitual e de acordo com Elmasri e Navathe (2005, p. 135), “não há um padrão universal para notações de diagramas E-R, e os diversos livros e softwares de diagramas E-R usam diferentes notações”. Partindo desse ponto, o primeiro passo é identificar as notações mais utilizadas e mais viáveis para um software, visando a maior praticidade para se criar modelos conceituais e maior aproveitamento do espaço na interface gráfica do software.

Após esse estudo, a segunda parte do trabalho é o desenvolvimento do software que permita aplicar esses conceitos e criar modelagens conceituais.

O trabalho está dividido em três capítulos, sendo o primeiro um levantamento teórico sobre as diferentes formas de diagramar uma modelagem conceitual. Serão comparados os

padrões adotados em livros de diferentes autores. Após esse estudo, serão analisadas várias ferramentas de modelagem de dados a fim de explicar o que falta nelas em relação ao software proposto nessa monografia.

O segundo capítulo se baseia em explicar como será desenvolvido o aplicativo, os processos documentais do software e suas funcionalidades.

O terceiro e último capítulo irá mostrar a ferramenta em produção, exemplificando seu uso no meio acadêmico e profissional a fim de mostrar os resultados do trabalho aqui proposto.

CAPÍTULO I

Fundamentação teórica

1. Modelagem conceitual

Na etapa de planejamento de um banco de dados, um fator importante é a modelagem conceitual, na qual o projetista procura criar uma estrutura na forma de diagramas de maneira que simplifique o entendimento do banco para todos os profissionais que estarão envolvidos na criação ou manutenção. De acordo com Silberschatz, Kort e Sudarshan (2006), os modelos conceituais criados na primeira parte do projeto traduzem as necessidades reais da empresa.

Não faz parte dessa monografia, explicar como deve ser feita a modelagem de um banco de dados, no entanto é necessário considerar os padrões de diagramas utilizados por vários autores literários para então, escolher um ou vários modelos a serem utilizados pelo software proposto. A seguir serão detalhados os principais elementos do modelo conceitual e suas diferenças entre autores.

1.1.2. Entidade

Uma entidade é uma abstração de algo do mundo real, como por exemplo, uma pessoa. Uma entidade representa todos os seus atributos.

De acordo com a maioria dos autores literários, o padrão de diagrama para expressá-la seria como a Figura 1.

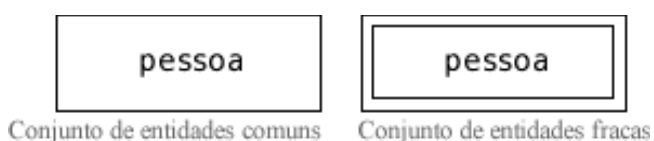
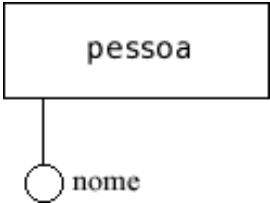
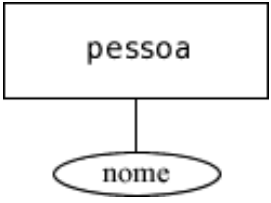
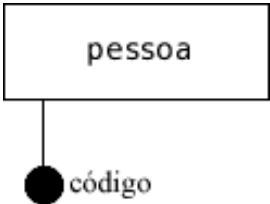

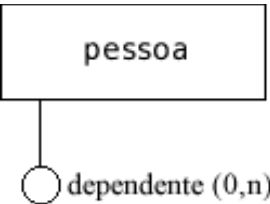
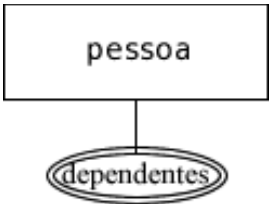

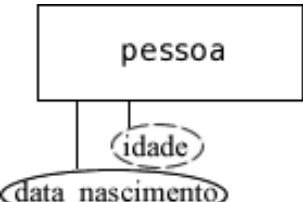


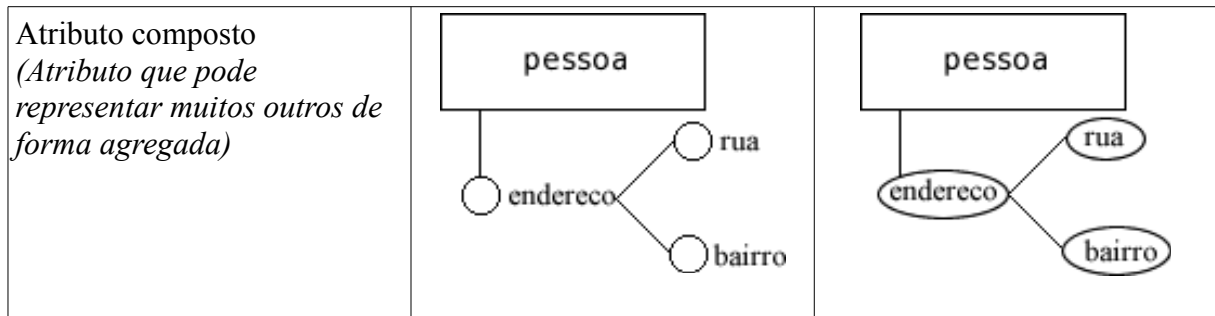
Figura 1 – Tipos de conjunto de entidades

1.1.3. Atributo

Um atributo é uma característica de uma entidade, um valor propriamente dito. Por exemplo, “nome” pode ser um atributo da entidade “pessoa”.

Com relação aos atributos há uma variação de padrões de diagramas entre Heuser(2004) e os demais autores como pode se perceber no Quadro 1

Descrição	Heuser	Maioria dos autores literários
Atributo comum		
Atributo chave (Atributo identificador da entidade)		
Atributo multivalorado (Tipo de atributo que pode representar vários valores, na prática, vira tabela)		
Atributo derivado (Tipo de atributo “calculado”. Deve ser evitado já que há meios de se conseguir o valor que o campo armazenaria)		



Quadro 1 – Relação de modelos de atributos

1.1.4. Cardinalidade

A cardinalidade entre relacionamentos determina, conceitualmente, a quantidade de vezes que uma entidade pode se relacionar com outra. Por exemplo, uma relação entre “produto” e “tipo”. Nesse caso, o diagrama ficaria como na Figura 2.

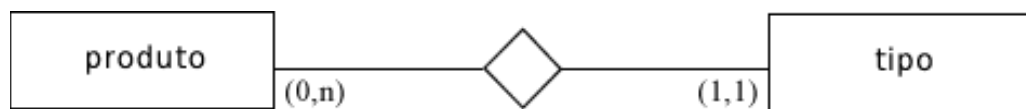
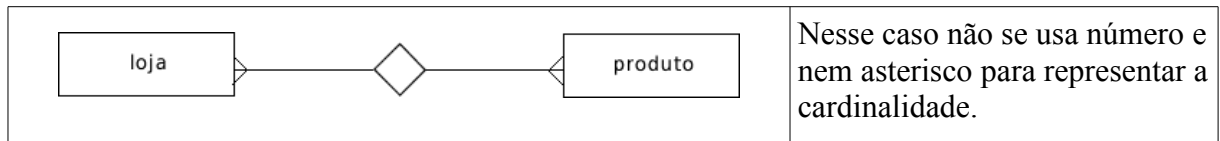


Figura 2 – Exemplo de cardinalidade mínima e máxima

A cardinalidade do lado do tipo significa que um produto tem **no mínimo** um tipo e **no máximo** um. Do lado contrário significa que um tipo pode não ser usado nunca (cardinalidade mínima) ou muitas vezes pelos produtos (cardinalidade máxima).

Existe uma grande variação entre os padrões de diagramas descritos no Quadro 2.

	Nesse modelo, a cardinalidade fica do lado da entidade.
	Como proposto por Navathe (2005, p. 51) e Silberschatz (2006, p. 158), a cardinalidade fica do lado do relacionamento.
	Em alguns modelos, a cardinalidade mínima pode ser omitida.
	A cardinalidade máxima também pode ser representada por um asterisco.



Quadro 2 – Diferentes maneiras de representar cardinalidade.

Como se pode perceber há muitas formas de expressar a cardinalidade e nem foi possível exemplificar todas, pois a maioria dos autores combina as características supracitadas. De acordo com Elmasri e Navathe (2005, p. 135), “não há um padrão universal para notações de diagramas E-R, e os diversos livros e softwares de diagramas E-R usam diferentes notações.”

Alguns autores descrevem a questão de cardinalidade única (1:1) com participação total (dependência de existência) como na Figura 3.

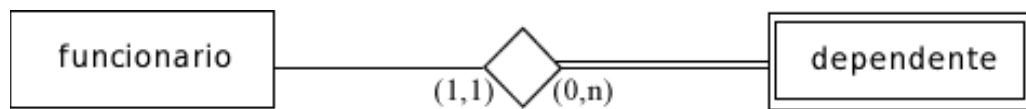


Figura 3 – Participação total e dependência de existência.

Nesse caso observa-se que como o dependente **sempre** terá uma relação com o funcionário e somente uma (1:1), se torna uma entidade fraca, pois só existe por causa da outra e ainda possui uma **participação total**, “enquanto a participação parcial é representada por uma linha única” (ELMASRI; NAVATHE, 2005, p. 48).

1.1.5. Relacionamento

Os relacionamentos são utilizados apenas para interligar os grupos de entidades e, segundo Teorey, “não possuem existência física ou conceitual além de sua dependência das entidades associadas.” (TEOREY; LIGHSTONE; NADEAU, 2009, p. 14).

Nos modelos propostos pelos autores pesquisados, não foram encontrado grandes diferenças, sendo basicamente como nas representações descritas abaixo.

A Figura 4 mostra a representação de um relacionamento ternário, segundo Machado e Abreu (MACHADO; ABREU, 1996, p. 100).

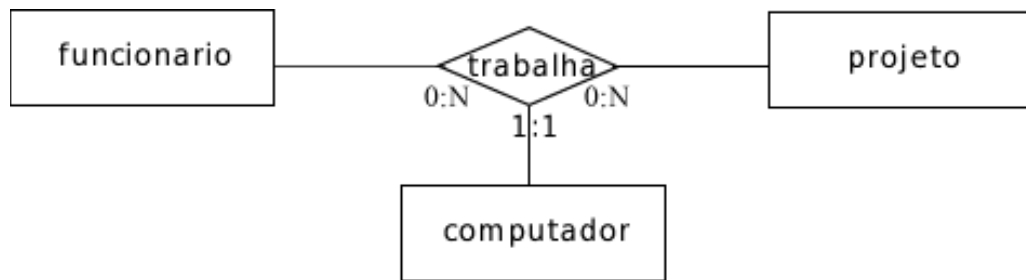


Figura 4 – Relacionamento ternário

Na Figura 5 é apresentada a representação de um relacionamento associativo segundo Teorey (TEOREY; LIGHSTONE; NADEAU, 2009, p. 27).

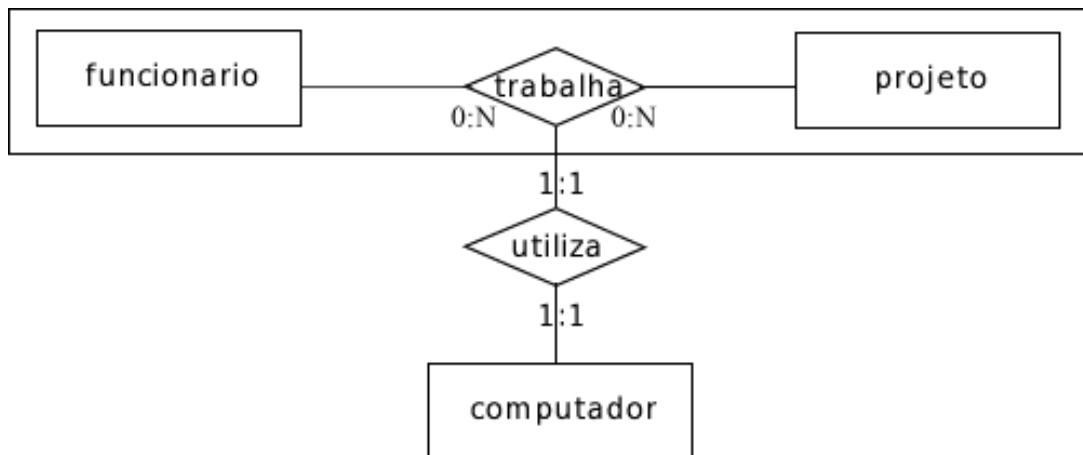


Figura 5 – Relacionamento associativo

A representação de um auto relacionamento é comumente representada como na Figura 6.

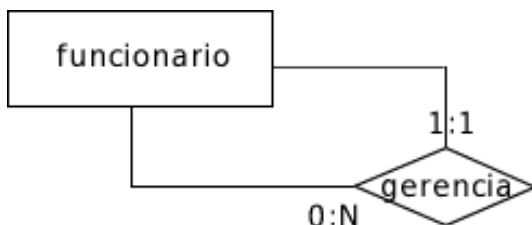


Figura 6 – Auto relacionamento.

1.1.6. Generalização/Especialização

Os recursos de generalização e especialização são utilizados para criar uma hierarquia

em determinados grupos de entidades. Por exemplo, as seguintes entidades: pessoa, motorista e médico. Ambos contêm dados comuns de pessoas, no entanto o médico possui CRM e o motorista possui CNH. Esse foi apenas um exemplo de que entidades que derivam de uma superior podem sofrer alterações. A nomenclatura de generalização ou especialização depende do propósito que foi modelado, pois visualmente é representado pelo mesmo símbolo. Na Figura 7 é possível observar como é diagramada uma generalização/especialização.

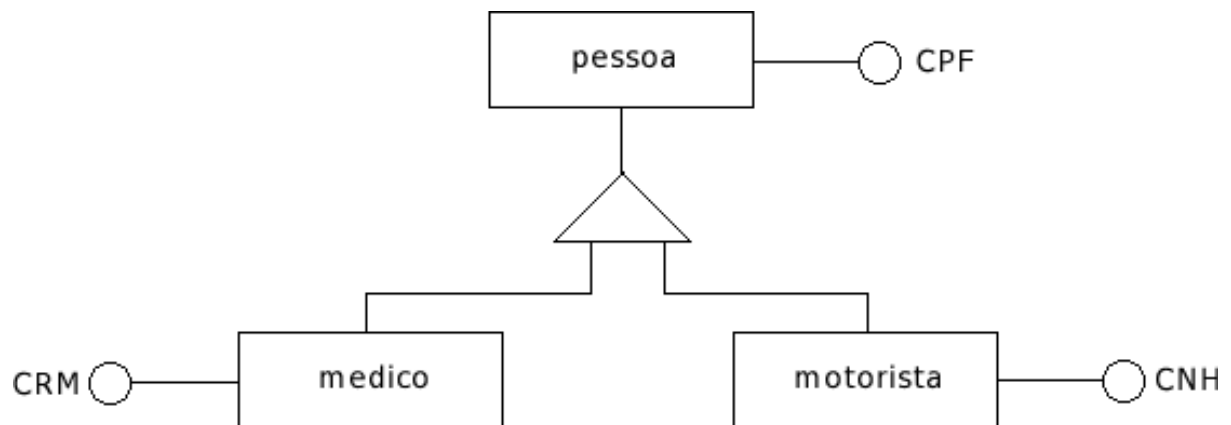


Figura 7 – Exemplo clássico de generalização/Especialização

Existem três tipos de especialização/generalização: total, parcial ou compartilhada. A seguir será apresentada uma breve descrição de cada uma com suas características.

1.1.6.1. Generalização total

Uma especialização/generalização parcial pode ser exemplificada entre as entidades cliente, pessoa física e jurídica.

Como todo cliente é uma pessoa física ou jurídica, significa que sempre terá uma entidade cliente e outra física ou jurídica para representar a mesma pessoa.

Uma generalização/especialização total pode ser representada conforme a Figura 8.



Figura 8 – Generalização total

1.1.6.2. Generalização parcial

No caso da especialização/generalização parcial, significa que nem sempre a entidade geral terá uma especialização, como no visto na Figura 9, onde nem sempre uma pessoa será um médico ou um motorista, fazendo com que não haja uma especialização em alguns casos.

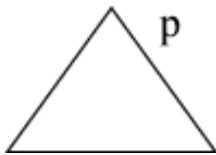


Figura 9 – Generalização parcial

1.1.6.3. Generalização compartilhada

Uma especialização compartilhada significa que a entidade genérica pode se especializar mais de uma vez, por exemplo, um cenário onde uma pessoa possa ser médico e motorista ao mesmo tempo.

No diagrama, essa generalização pode ser representada igual a Figura 10.

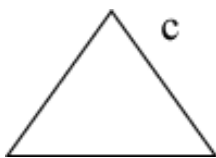


Figura 10 – Generalização compartilhada

1.2. Relação entre ferramentas disponíveis no mercado.

As ferramentas de modelagem encontradas atualmente no mercado não atendem a algumas características desejadas pelos meios acadêmico e profissional. A maioria das ferramentas não contam com o recurso de modelagem conceitual, o que é extremamente utilizado nas instituições de ensino e também por desenvolvedores de software por ter uma representação simplificada do esquema do banco. Algumas ferramentas como, por exemplo o

brModelo(CÂNDIDO, 2005), possuem modelagem conceitual, porém são rígidos em um único sistema operacional. Outras ferramentas possuem muitas funcionalidades tornando o uso inviável para o propósito desejado.

Observando a Figura 11 é possível identificar as principais características procuradas nas ferramentas que foram testadas. Algumas foram encontradas, outras não. A seguir será explanada uma a uma, sempre confrontando com as funcionalidades desejadas (Figura 11) para o projeto proposto nessa monografia.

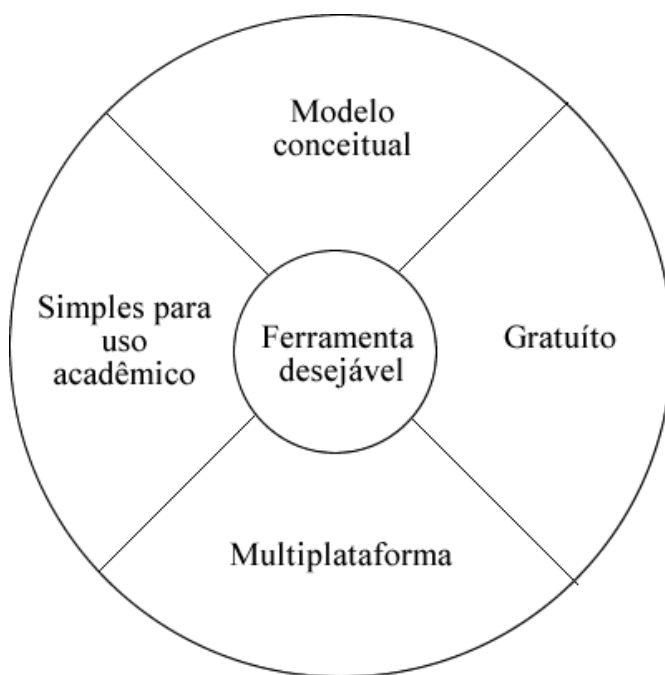


Figura 11 – Características desejadas.

1.2.1. DBDesigner 4

Uma das ferramentas utilizadas para modelagem e manutenção de banco de dados é o DBDesigner.

Em sua última versão, o software é focado na modelagem lógica e não possui opção de modelagem conceitual. As vantagens da ferramenta é o fato de fazer engenharia reversa, criando o modelo a partir de um banco de dados já existente. Apesar disso, a ferramenta é fortemente atrelada ao MySQL como mostrado na Figura 13.

As principais características dessa ferramenta são o fato de ser gratuita, possuir

modelagem lógica, ser multiplataforma e multibanco (apesar de ser fortemente ligada ao MySQL).

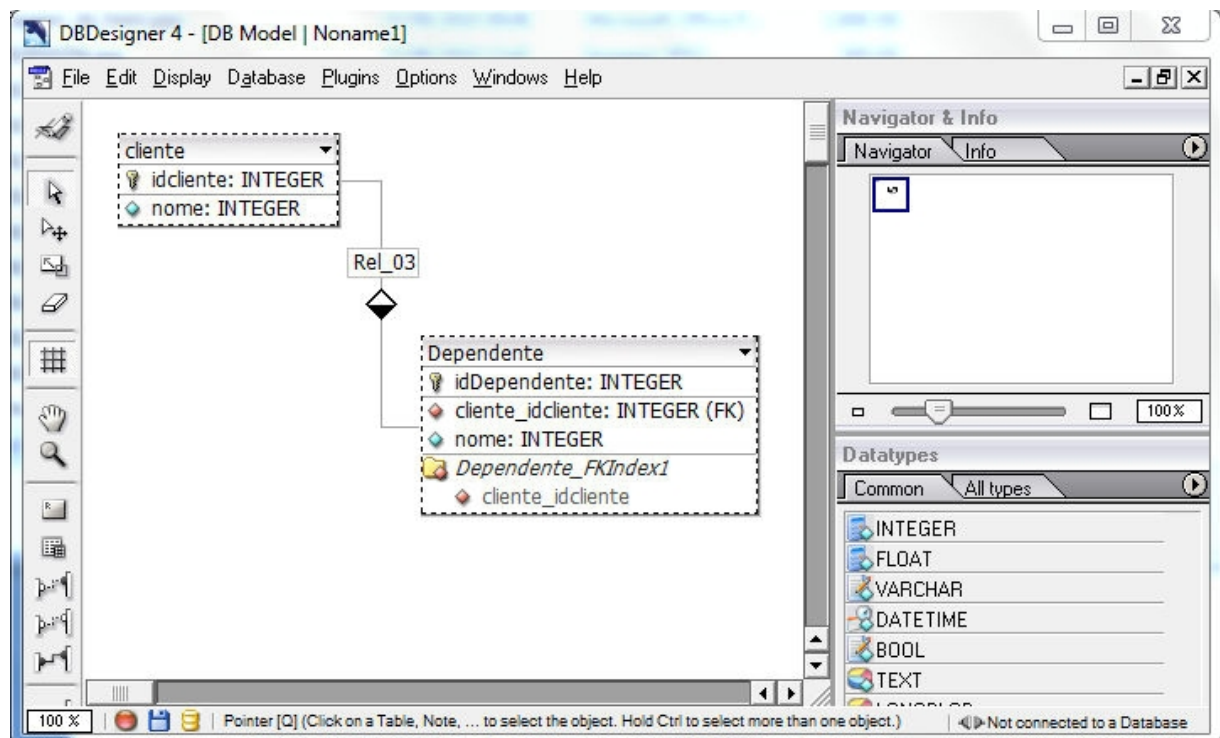


Figura 12 - Modelagem no DBDesigner (WWW.FABFORCE.NET, [s.d.])

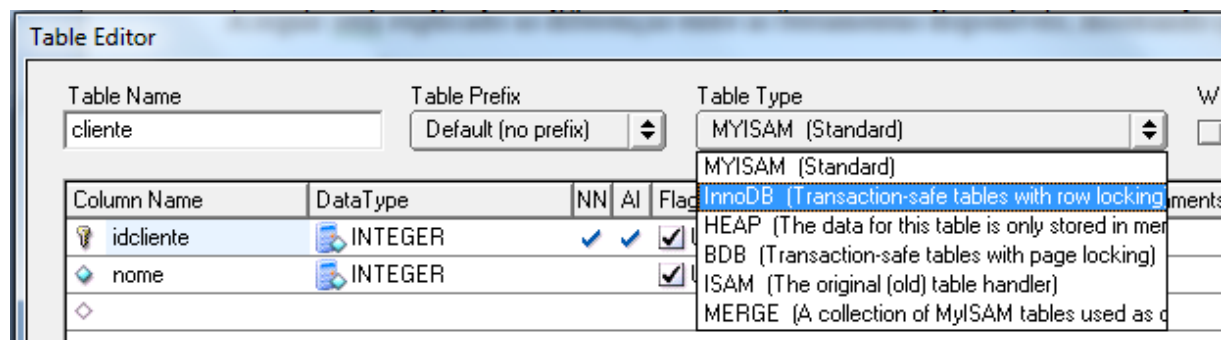


Figura 13 – Tipo de tabela no DBDesigner

1.2.2. CA ERwin Data Modeler r8.2

O CA Erwin Data Modeler faz apenas modelagem lógica e física e possui portabilidade para vários sistemas de banco de dados, no entanto é uma ferramenta paga e só possui versão para Windows.

A interface gráfica do software é repleta de opções devido à quantidade de

funcionalidades que possui, podendo ser desnecessários em um ambiente acadêmico.

Um exemplo de utilização do CA ERwin Data Modeler pode ser visto na Figura 14.

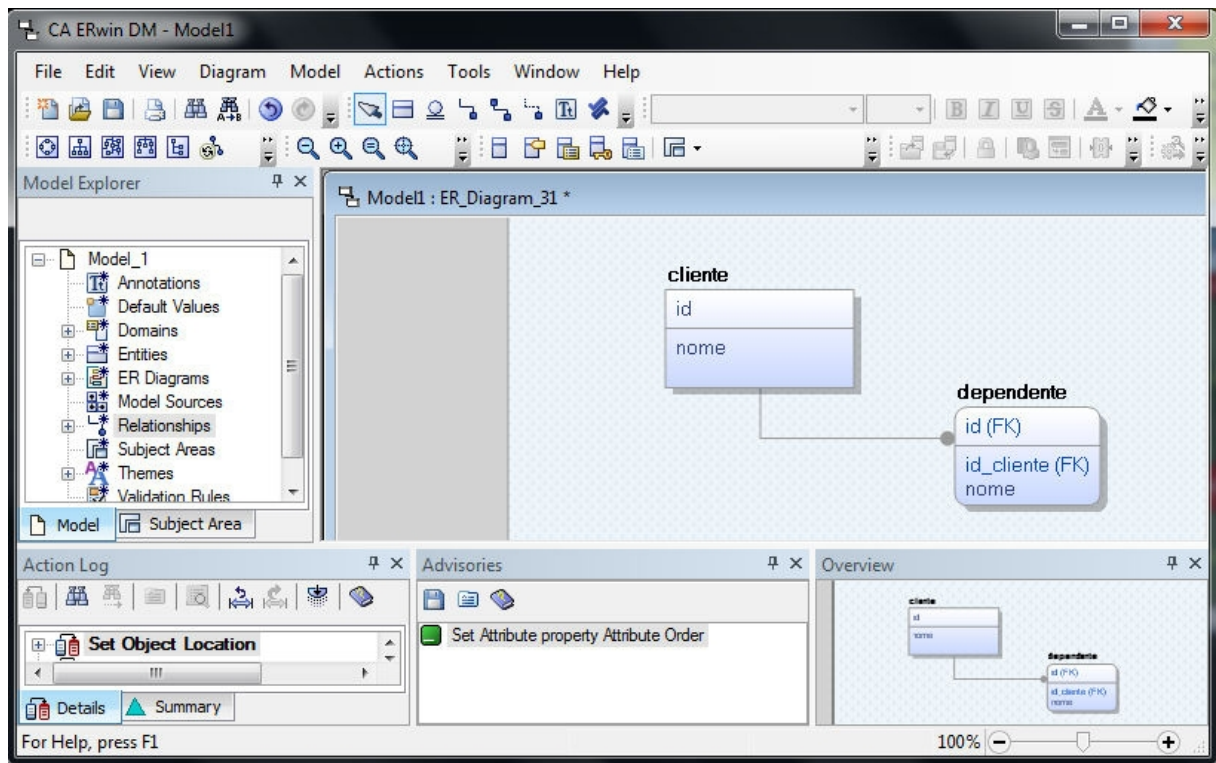


Figura 14 – CA ERwin Data Modeler.(WWW.ERWIN.COM, [s.d.])

1.2.3.Xcase

Assim como a maioria das ferramentas, o Xcase também não faz modelagem conceitual. Muito semelhante ao ERwin Database Modeler, o Xcase é capaz de sincronizar com um banco de dados já implementado e fazer a engenharia reversa.

Um ponto muito positivo da ferramenta é a quantidade de Sistemas Gerenciadores de Bancos de Dados (SGBD's) suportados que podem ser observados na Figura 15.

Um ponto negativo que não atende aos requisitos da ferramenta proposta é fato de não ser multiplataforma e só funcionar em plataforma Windows.

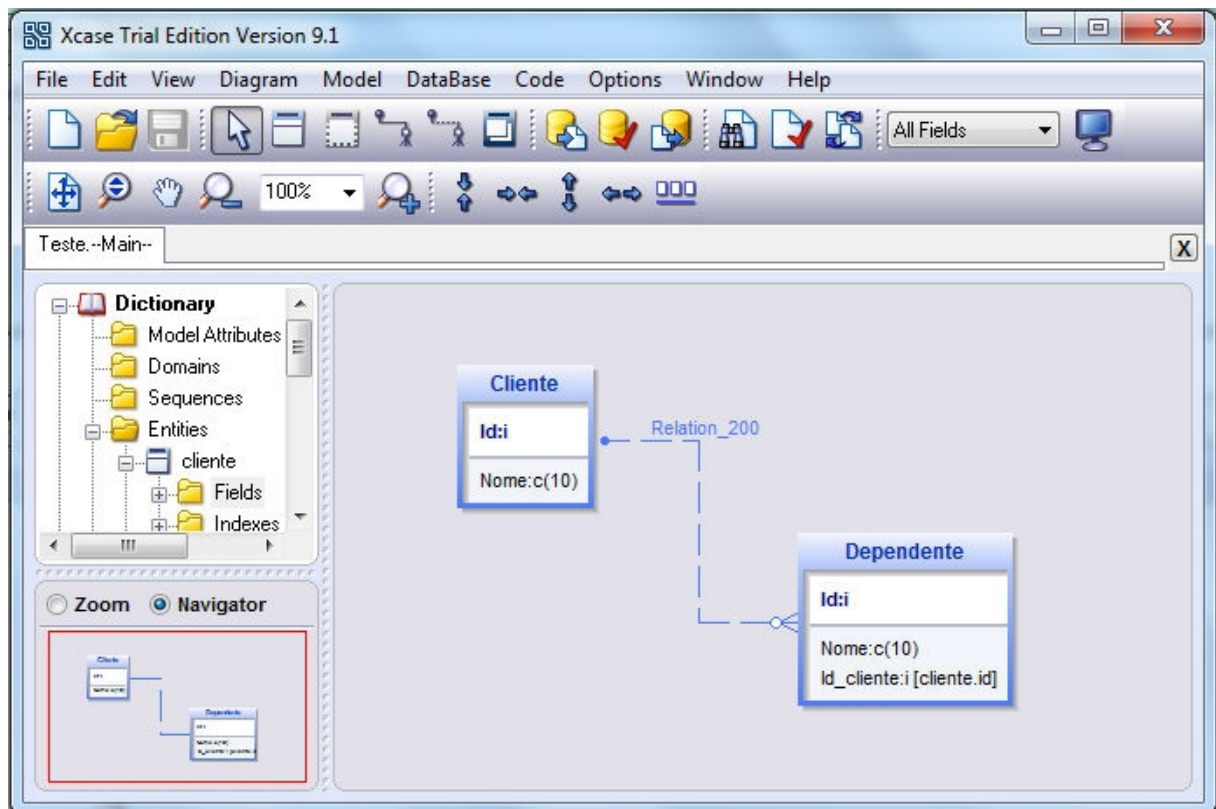


Figura 15 – Modelagem lógica no Xcase(WWW.XCASE.COM, [s.d.])

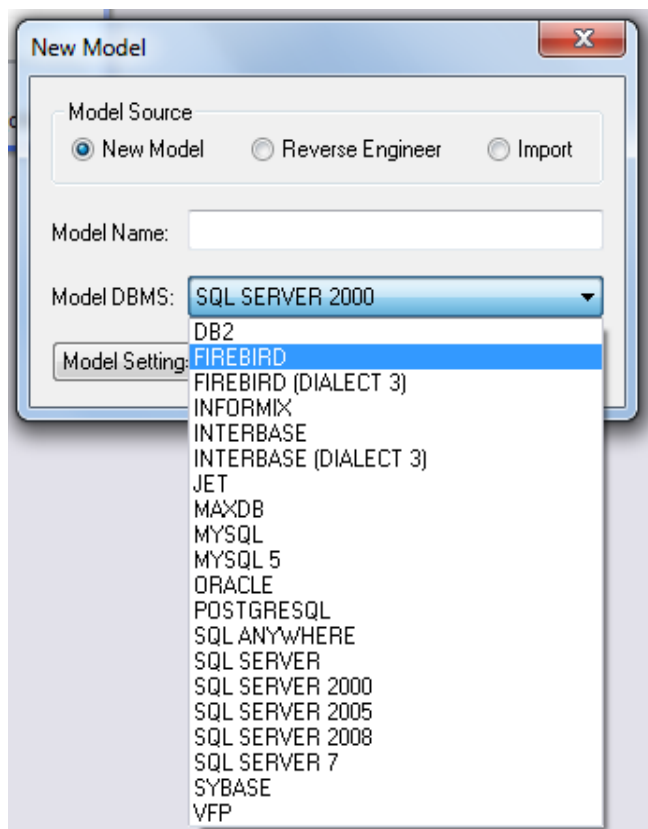


Figura 16 – SGBD's suportados pelo Xcase

1.2.4. Design for Databases V7.2.0

Sem fugir muito das características descritas nas ferramentas anteriores, o DeZign for Databases V7.2.0 também só trabalha com a modelagem lógica e física, porém não faz sincronização com o banco de dados, apenas gera o script.

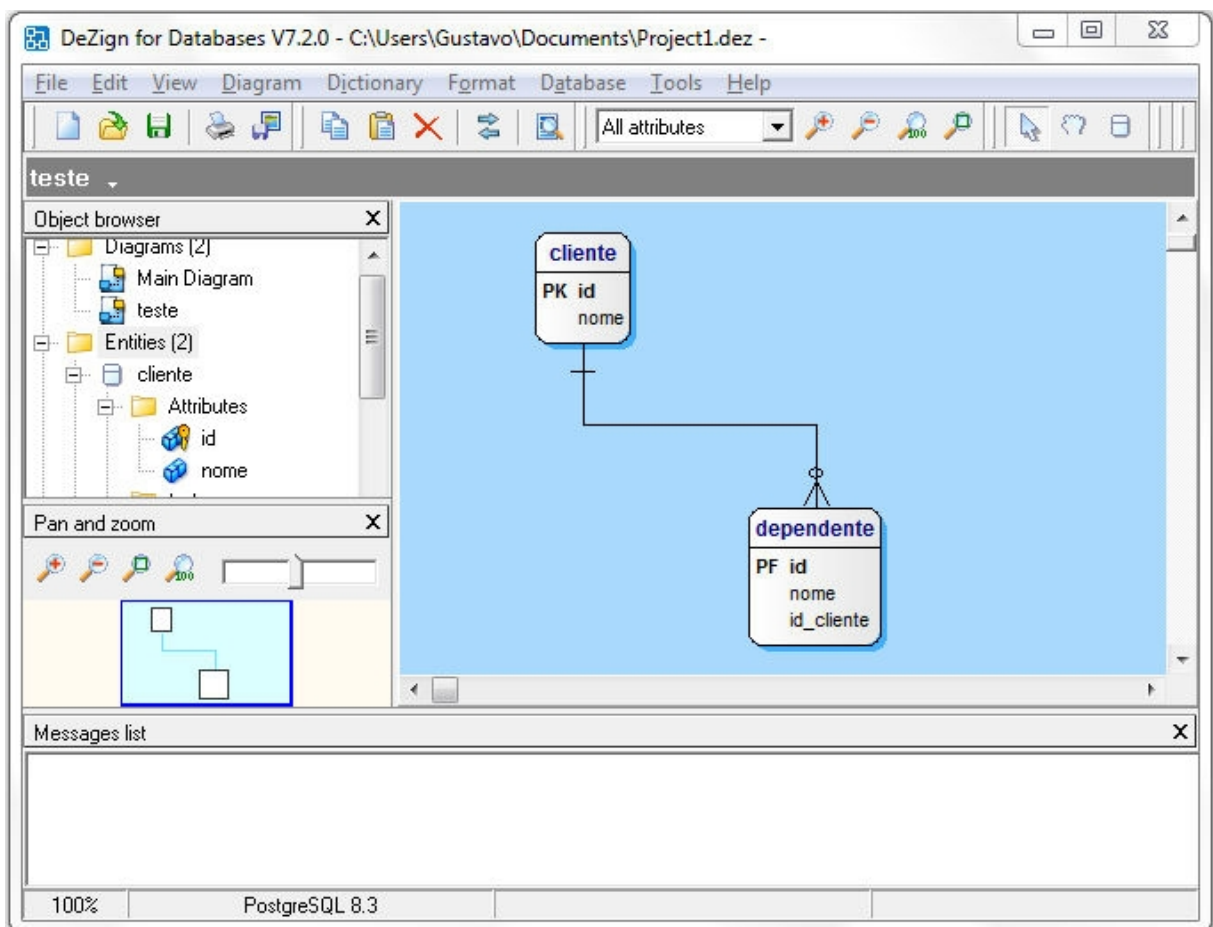


Figura 17 - Modelagem lógica no DeZign for Databases V7.2.0(WWW.DATANAMIC.COM, [s.d.])

1.2.5. PowerDesigner

O PowerDesigner se mostra uma ferramenta bem extensa pelo fato de permitir muitos padrões de diagramas como diagramas multidimensionais e modelos lógicos. A ferramenta também não possui a opção de criação de modelo conceitual.

Outros dois pontos importantes são o fato de ser uma ferramenta paga e só funcionar em plataformas Windows.

Na Figura 17 pode-se observar a ferramenta em funcionamento.

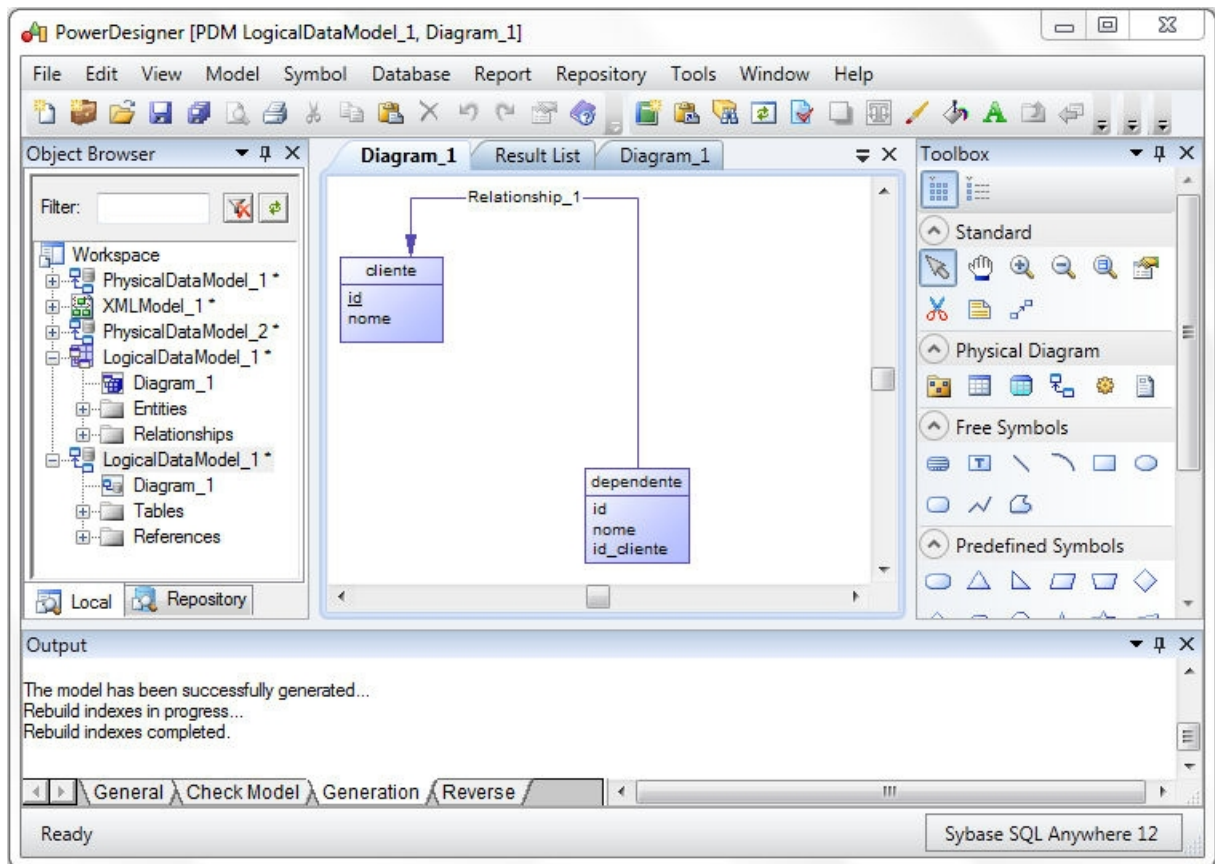


Figura 17– Modelagem lógica no PowerDesigner(WWW.SYBASE.COM.BR, [s.d.])

1.2.6. Enterprise Architect

Nem todas as ferramentas utilizadas para fazer modelos conceituais são próprias para banco de dados, como é o caso do Enterprise Architect que é uma ferramenta para diagramação em geral e UML, porém utilizada para representar conceitualmente as estruturas de dados.

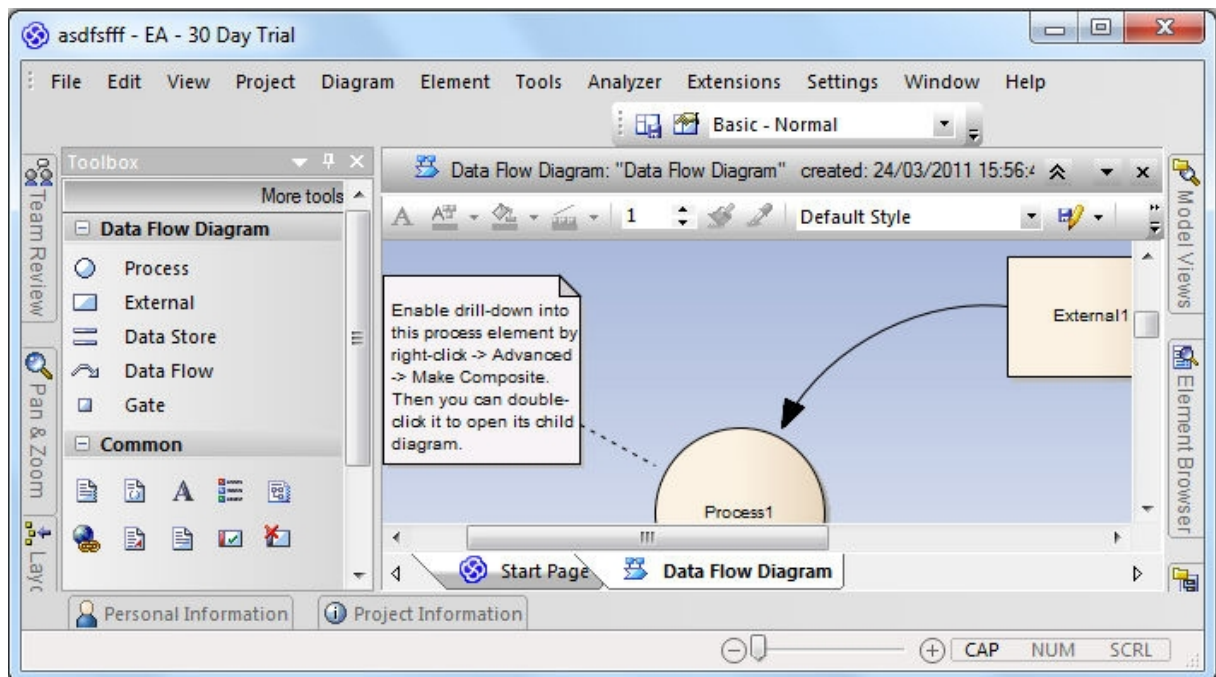


Figura 18 – Modelagem genérica com Enterprise Architect. (WWW.SPARXSYSTEMS.COM.AU, [s.d.])

1.2.7. brModelo

A ferramenta que mais se aproxima do software desejado é o brModelo. O software trabalha com o modelo conceitual, lógico e também o físico. Trata-se de um projeto de dissertação de pós-graduação e possui código fonte livre e gratuito. No entanto o brModelo só funciona na plataforma Windows e não tem uma boa implementação para a modelagem física, gerando um script genérico sem opção de escolher o sistema de banco de dados desejado. Ainda falando da camada física, o brModelo não sincroniza com um banco de dados existente para atualizá-lo ou fazer engenharia reversa.

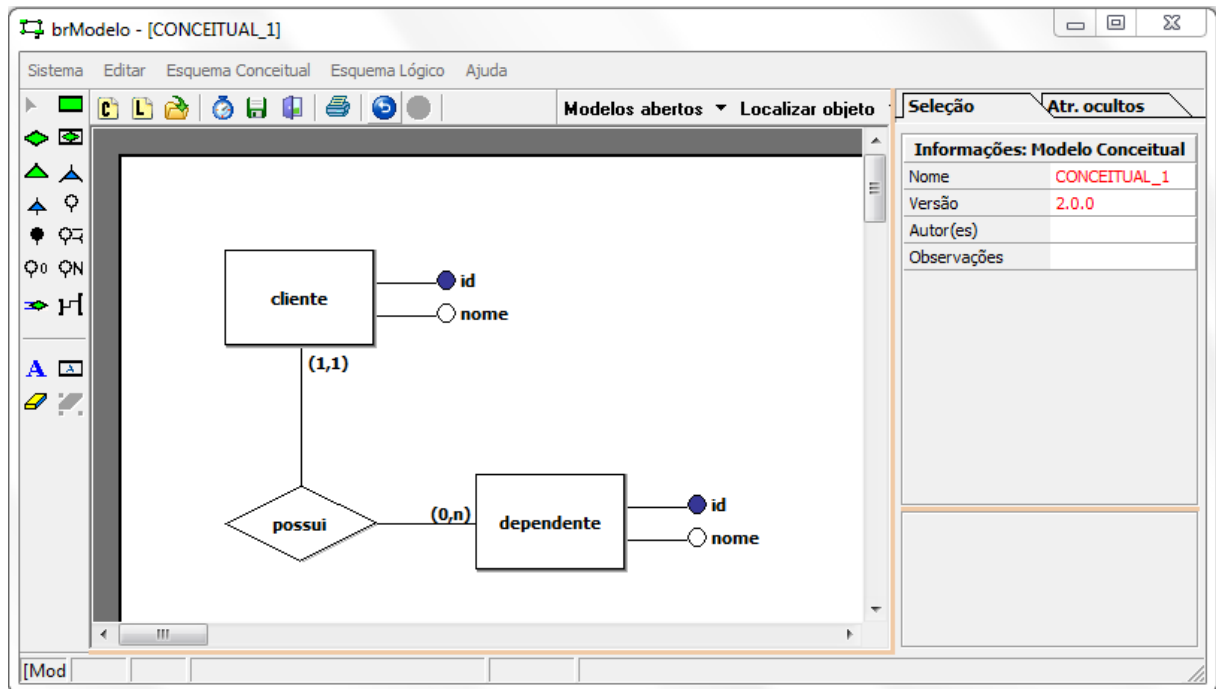


Figura 19 – brModelo, Ferramenta mais adequada atualmente para modelagem conceitual. (CÂNDIDO, 2005)

1.2.8. Visual Paradigm

A ferramenta Visual Paradigm é uma das poucas ferramentas que possuem a opção de trabalhar modelagem conceitual, lógica e física (vide Figura 21), no entanto, o padrão dos diagramas utilizados é muito semelhante ao modelo lógico como observado na Figura 20.

A ferramenta só é disponibilizada para a plataforma Windows e é proprietário, tendo um custo para licenciá-la.

Assim como o CA Erwin Data Modeler, sua interface também é repleta de funcionalidades que dependendo da situação pode ser desnecessária, como por exemplo para uso didático. A interface pode ser observada na Figura 20.

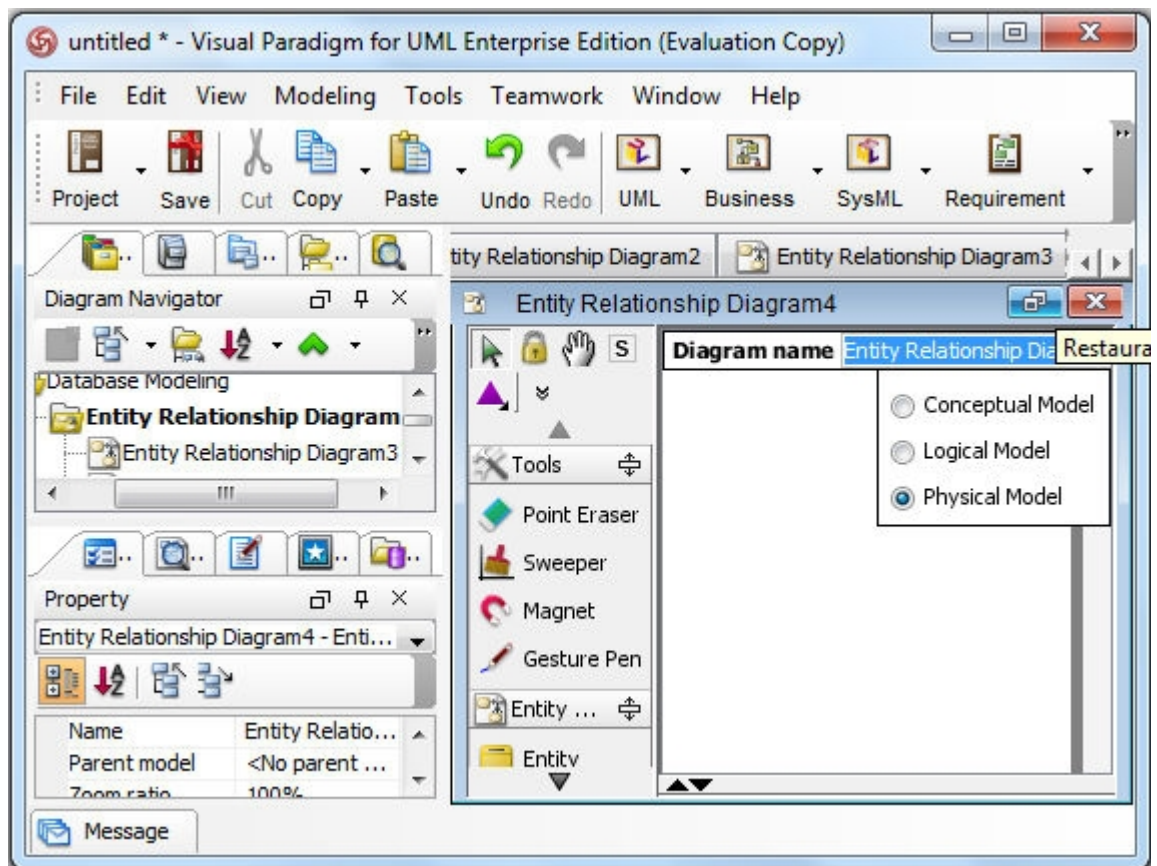


Figura 20 – Escolhendo o tipo de modelo, Visual Paradigm(WWW.VISUAL-PARADIGM.COM, [s.d.])

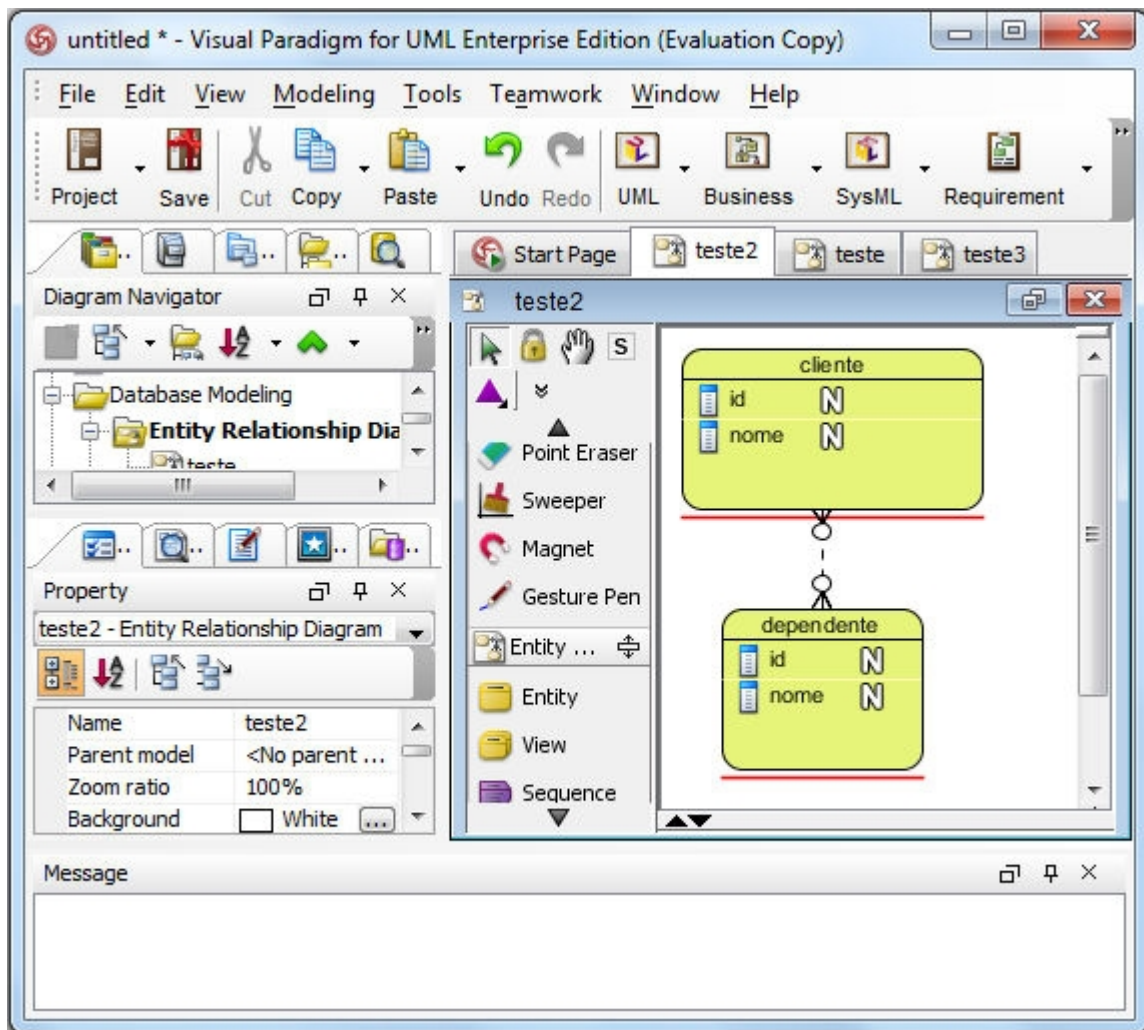


Figura 21 – Diagrama conceitual do Visual Paradigm

1.2.8. Dr.CASE 3.50f

A ferramenta Dr.CASE se mostrou muito próxima do software proposto por essa monografia. Esta ferramenta é capaz de criar modelo conceitual, lógico e físico. Suporta sincronização com o banco (chamado de Engenharia Direta) e engenharia reversa para criar o modelo a partir de um banco de dados existente.

O ponto negativo é que a ferramenta funciona apenas na plataforma Windows e é proprietária, necessitando de uma licença paga para sua utilização.

Algumas diferenças no padrão de diagramas foram encontradas, por exemplo, os atributos das entidades não aparecem no diagrama, apenas são utilizados para gerar o modelo lógico e conceitual.

O Dr.CASE é um projeto aparentemente antigo, o que pode ser confirmado na Figura 23 e apesar de suportar muitos banco de dados, todos são muito antigos e pouco utilizados como Paradox 3.x e MS Access 95.

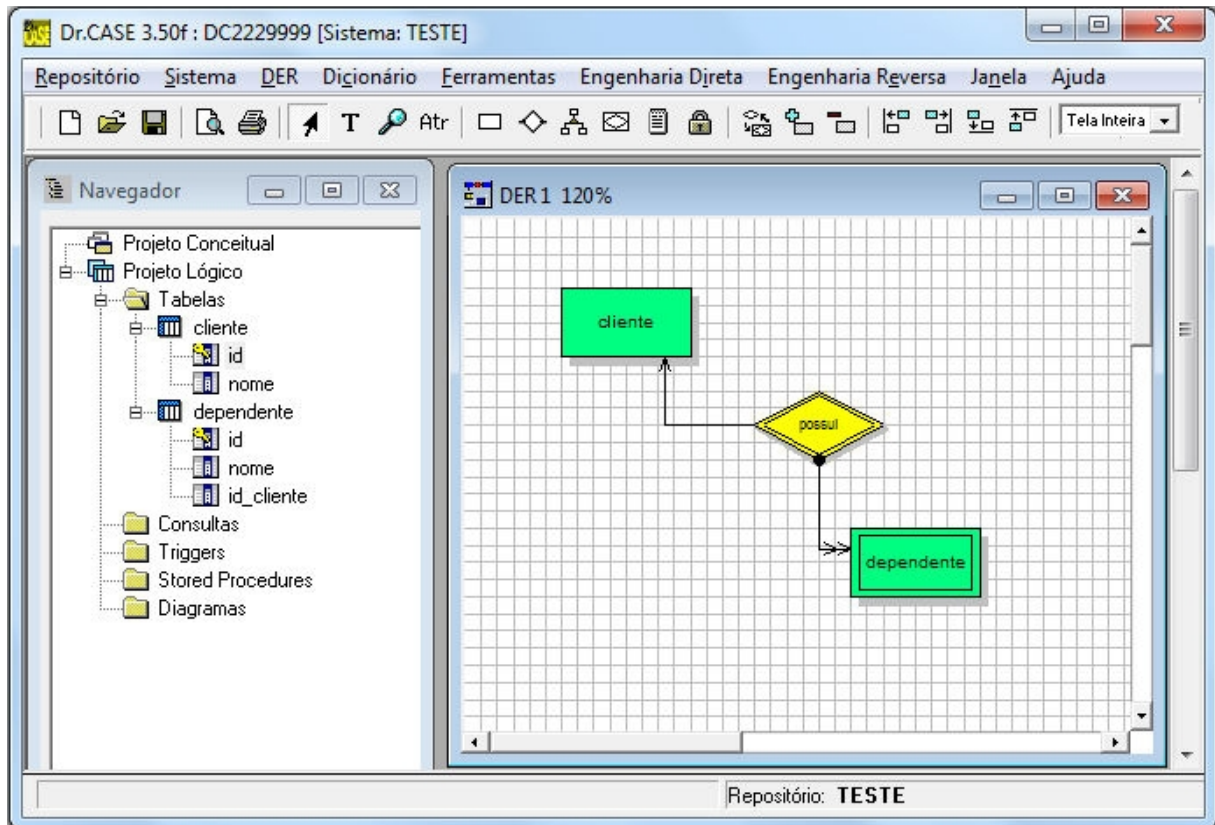


Figura 22 – Modelo conceitual no Dr.CASE (WWW.SQUADRA.COM.BR, [s.d.])

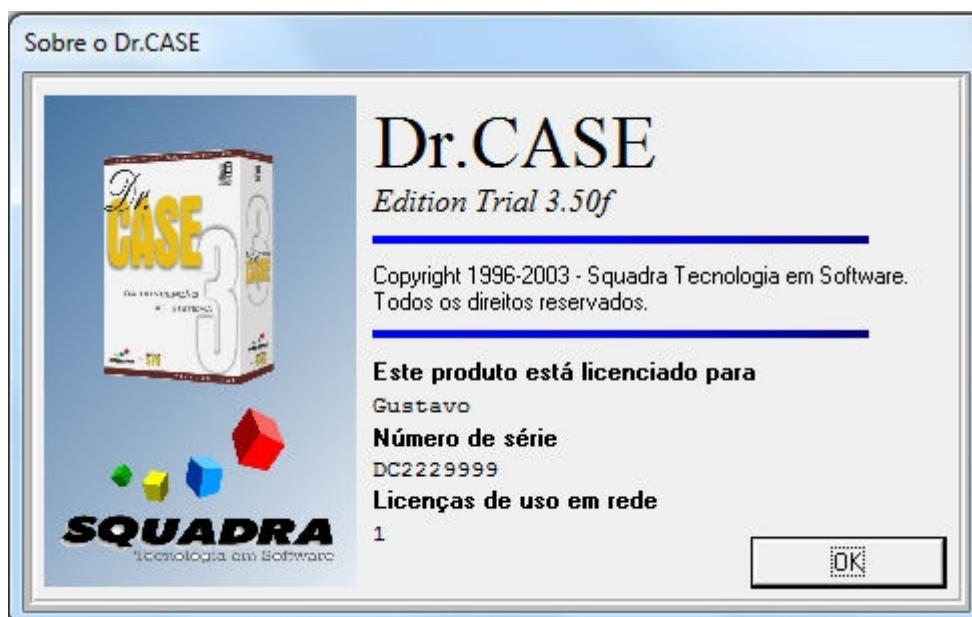


Figura 23 – Tela de “sobre” do Dr.CASE

1.2.9. DbWrench Database Design and Synchronization

A ferramenta DbWrench é muito eficaz na modelagem lógica, sincronização e engenharia reversa, porém possui licença paga. O sistema não trabalha com modelagem conceitual, no entanto é multiplataforma. Sua interface pode ser vista na Figura 24.

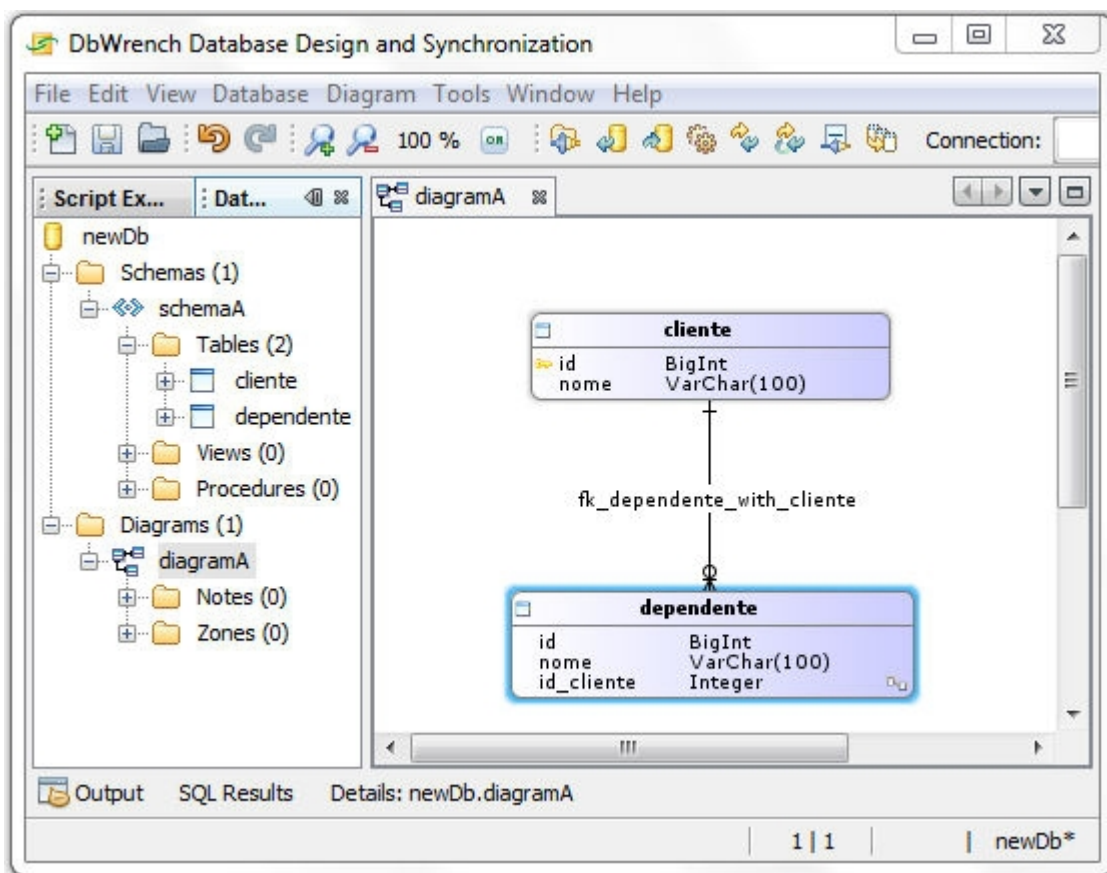


Figura 24 – Modelagem lógica no DbWrench. (DBWRENCH.COM, [s.d.])

1.2.10. Astah

O Astah é uma ferramenta utilizada por desenvolvedores e engenheiros de software, pois carrega um conjunto de funcionalidades muito grande para UML.

Além de trabalhar com os diagramas da UML, o Astah pode trabalhar com diagramas de entidade relacionamento, porém não tem nada para suportar a modelagem conceitual.

É uma ferramenta com licença paga, porém existe a versão gratuita com menos

funcionalidades.

Um dos pontos interessantes do Astah é o fato dele ser multiplataforma, possibilitando que funcione em todas as plataformas que tiverem o Java instalado, ou seja, pode funcionar em plataforma Window, Linux, Mac, Solaris entre outros.

Seu funcionamento pode ser observado na Figura 25.

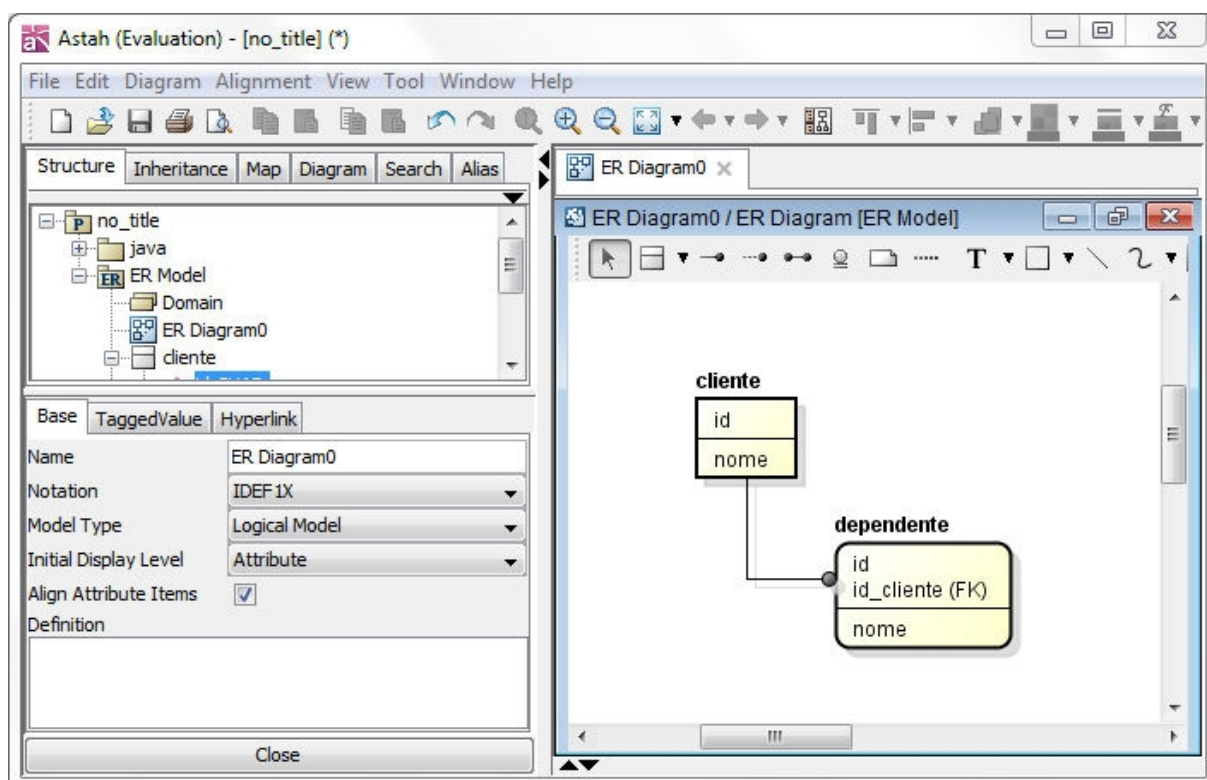


Figura 25 – Diagrama Entidade Relacionamento no Astah (WWW.ASTAH.NET, [s.d.])

CAPÍTULO II

Materiais e métodos

2. Desenvolvimento

O software foi criado para suprir a necessidade de uma ferramenta de modelagem de dados que seja gratuita, multiplataforma, estável e capaz de trabalhar com os diagramas utilizados pelo MER (Modelo Entidade Relacionamento).

A ferramenta foi desenvolvida utilizando a linguagem de programação Java devido à sua grande portabilidade entre sistemas operacionais, com o mínimo de modificações. O desenvolvimento e testes ocorreram sobre a plataforma GNU-Linux, mais especificamente a distribuição Ubuntu.

A IDE (Integrated Development Environment, Ambiente Integrado de desenvolvimento) Netbeans foi utilizada em razão de sua compatibilidade com a linguagem de programação e com o sistema operacional usado. O Netbeans também oferece várias funcionalidades que facilitam a codificação do software, em especial o recurso de *debug* e internacionalização.

O conjunto de softwares utilizados no desenvolvimento foi basicamente constituído de: GNU-Linux (Ubuntu), Netbeans, Java, MySQL e PostgreSQL principalmente pelo fato de serem todos gratuitos e de código-fonte aberto, o que encoraja o combate à pirataria de software.

O software foi desenvolvido usando padrões de projetos (boas práticas de programação) que garantem um código-fonte organizado de fácil manutenção. O principal padrão utilizado foi o MVC (*Model-View-Controller*, ou Modelo-Visualização-Controle). De acordo com Gamma et al (2000), “A abordagem MVC separa Visão e Modelos pelo estabelecimento de um protocolo do tipo inserção/notificação (*subscribe/notify*) entre eles.” como pode-se ver na Figura 26.

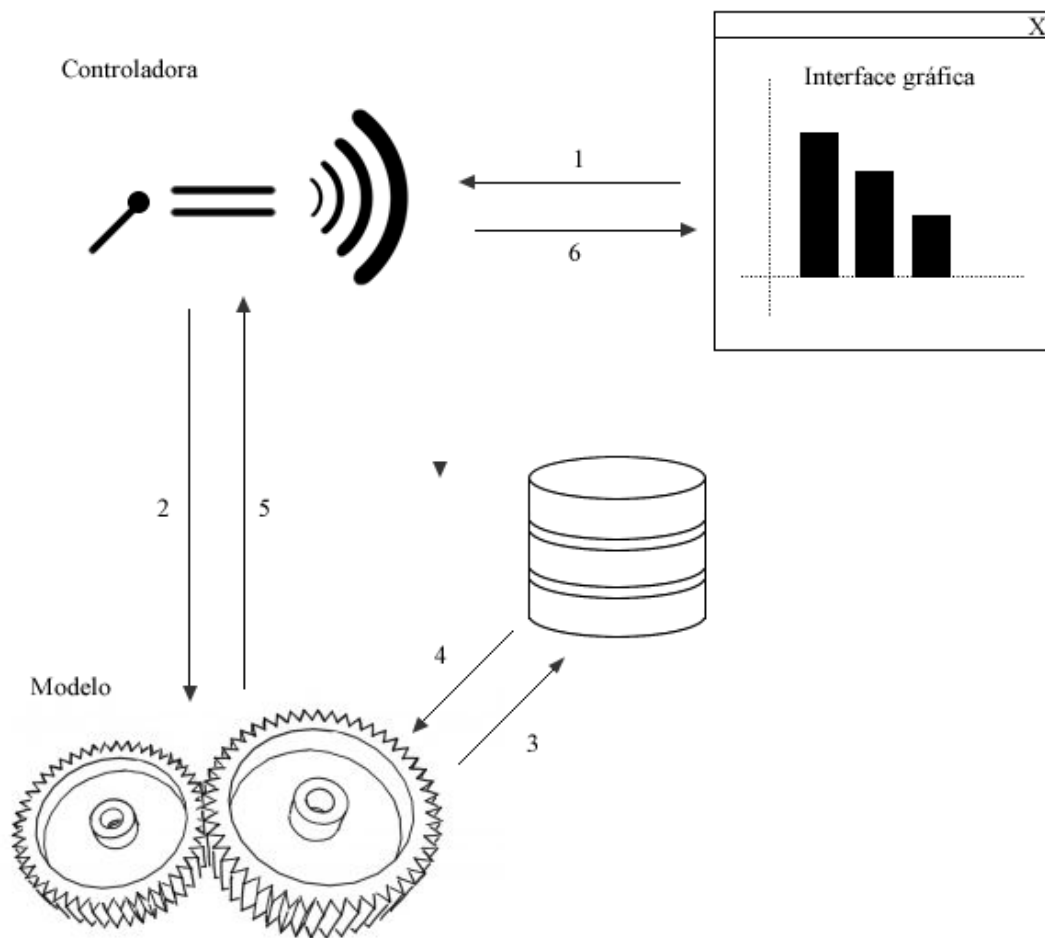


Figura 26 – Analogia ao padrão de desenvolvimento em camadas

De acordo com esse padrão de projeto, o software foi desenvolvido usando as seguintes características de camadas, baseadas na definição de Gamma et al (2000):

Model: Essa camada contém as classes com métodos para manipulação de banco de dados. Também se encontra nessa camada os objetos transientes (Objetos de valor) e classes que contém métodos para processar regras de negócios. Pode-se dizer que a camada *model* é o motor do sistema.

View: Nessa camada se encontra as classes de interface gráfica. É a parte do código-fonte responsável por *renderizar* telas, componentes, imagens, botões, etc. Porém essas interface gráficas não são capazes de interagir com o usuário.

Controller: A camada de controle é a ponte entre a camada *view* e a camada *model*.

Ela é responsável por tratar as interações do usuário e solicitar ao motor da aplicação determinado processamento. Após o término do processamento a *controller* repassa os resultados para que a camada *view* seja atualizada.

2.1. Internacionalização

É importante salientar que o software pretende atingir a maior quantidade possível de idiomas, para isso foram adotadas as técnicas de internacionalização que o Java suporta, utilizando um arquivo de propriedades internas com os valores textuais que são exibidos ao usuário.

Os arquivos de idiomas devem todos possuir a seguinte nomenclatura para nomeação:

language_[idioma]_[PAÍS].properties.

Por exemplo:

language_pt_BR.properties

O interior dos arquivos de tradução devem possuir uma sequência de chaves e valores, como pode-se ver na Figura 27, extraído do software proposto neste trabalho.

```
#Tag utilizado na barra de titulo da janela principal do
sistema
titulo=MK-Model - Build 7-3
#Nome do sistema
nome_sistema=MK-Model

#Notificacoes
salvando=Salvando...
confirmacaoSaida=Deseja realmente sair?
tituloConfirmacaoSaida=Sair
textDialogObservacao=Observação
textDialogHelp=Você pode apertar CTR+[ENTER] para concluir
atributoClicarEntidade=Esse componente não aceita atributos ou
você está tentando colocar o atributo solto na tela.
tituloAtencao=Atenção
tituloErro=Erro
```

Figura 27 – Conteúdo do arquivo de idiomas, versão em português.

Para serem utilizados os arquivos de traduções, deve-se usar um objeto da classe *ResourceBundle* que fica estático para a instância inteira do software como apresentado na Figura 28. Para utilizar os literais, basta consultar pela chave no objeto de tradução, como

visto na Figura 29.

```
public static final ResourceBundle lang =
ResourceBundle.getBundle("br.com.gqferreira.mkmodel.languages.
language", new Locale("pt", "BR"));
```

Figura 28 – Forma de aplicar um arquivo de idiomas

```
String x = lang.getString("confirmacaoSaida");
```

Figura 29 – Forma de capturar o conteúdo literário do arquivo de tradução.

2.2. Estrutura lógica

O sistema possui uma estrutura altamente escalável que permite sua ampliação com uma certa simplicidade. Nessa estrutura, existe uma hierarquia entre os módulos do software que são os de modelagem conceitual, modelagem lógica e física. Os recursos são acoplados tanto graficamente como na codificação de baixo nível, permitindo-se alternar entre os módulos apenas substituindo os objetos acoplados. Essa estrutura pode ser observada na Figura 30.

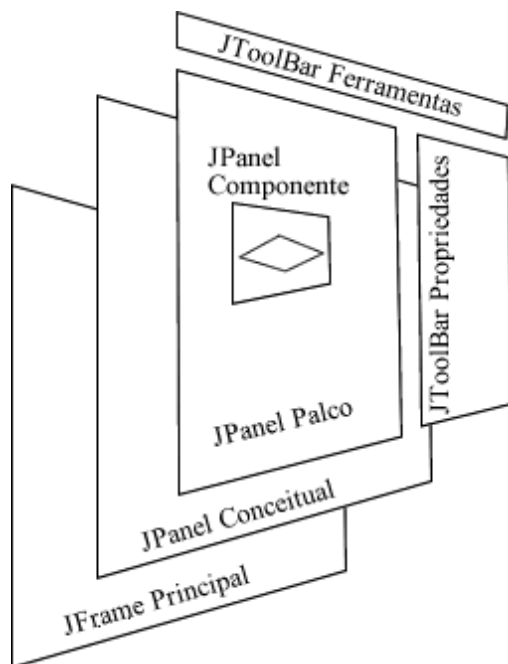


Figura 30 – Estrutura lógica da codificação.

2.3. Estrutura de pacotes

O software seguiu um padrão rígido de nomeação de pacotes, usado pela convenção de boas práticas (<http://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html>). De acordo com a convenção, os pacotes não devem conter letras maiúsculas. A estrutura do projeto pode ser vista na Figura 31.

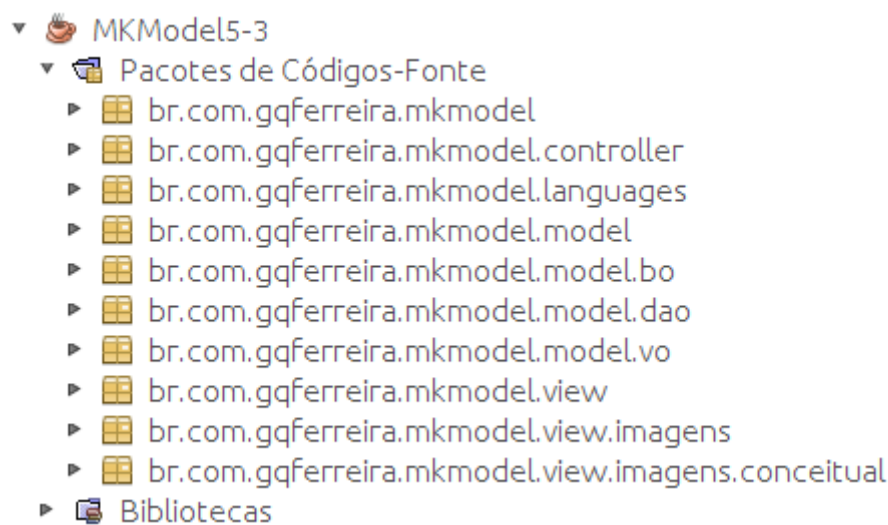


Figura 31 – Estrutura de pacotes.

Seguindo ainda a convenção, os pacotes devem começar com o domínio do site da empresa ou do dono do projeto.

Como pode haver vários projetos de uma mesma empresa/desenvolvedor, a nomenclatura de pacotes precisa ter a região ou o nome do projeto após o domínio do site. Como sugerido também na convenção de boas práticas.

2.4. Herança e polimorfismo

O software foi criado utilizando práticas de orientação a objetos, para isso foi criado classes que herdam de classes pais. Todos os componentes (entidade, relacionamento, atributos, etc) utilizados na modelagem conceitual são, em sua forma mais primitiva, objetos da classe JPanel, no entanto foram especializadas para implementar comportamentos

adicionais. Na Figura 32 podemos ver a estrutura hierárquica das classes responsáveis por criar atributos.

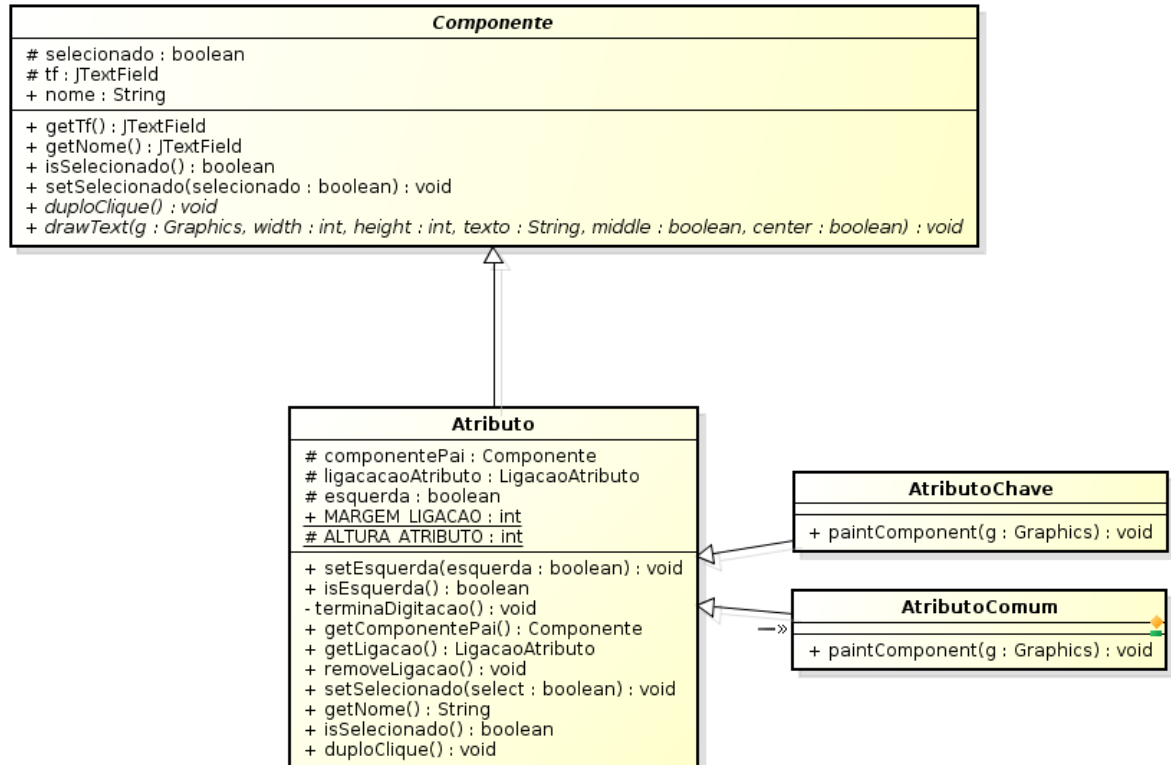


Figura 32 – Herança de classes

Como pode-se ver nesse diagrama de classes, a classe “Componente” é abstrata e apenas define métodos e atributos que devem ser posteriormente implementados pelas classes filhas. Alguns desses métodos são abstratos e outros não, o que torna a classe inteira abstrata já que tendo ao menos um método abstrato, toda a classe se torna abstrata.

As assinaturas das classes podem ser vistas na Figura 33:

```

public abstract class Componente extends JPanel
public class Atributo extends Componente
public class AtributoComum extends Atributo
  
```

Figura 33 – Assinatura das classes descritas na Figura 32

2.5. Algoritmo de ligação de atributos

O modelo conceitual exige que os atributos estejam ligados visualmente a uma entidade. No software desenvolvido foi criada uma classe responsável pelo desenho da linha

de ligação entre o atributo e a entidade. Essa classe é extensa e inclui dois grandes métodos responsáveis por resolver os algoritmos de cálculos de tamanhos e posições das linhas. As linhas são desenhadas dentro de um retângulo de tamanho reduzido, evitando que toda a tela seja atualizada a cada vez que o desenho da linha precise ser alterado, ganhando assim, desempenho durante a utilização das ligações.

O primeiro método da classe que desenha a linha é responsável por calcular o tamanho do retângulo que será usado para posteriormente receber o desenho da linha. O algoritmo faz várias validações com relação à posição do atributo e da entidade para determinar a posição e tamanho do retângulo. As possíveis regiões em que o retângulo pode ocupar está representado na Figura 34.

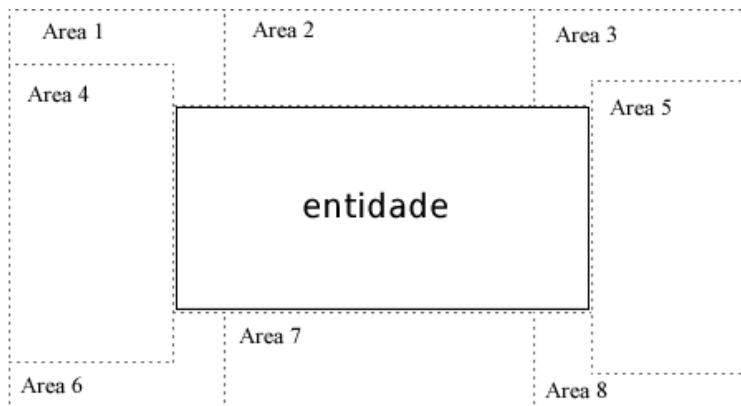


Figura 34 – Áreas para ligações de atributos

O segundo algoritmo é responsável por desenhar a linha dentro do retângulo disponível. Para isso o método realiza mais cálculos para determinar onde haverá a quebra de linha e posição das conexões com a entidade. O esquema completo de ligações, atributos e entidade pode ser visto na Figura 35.

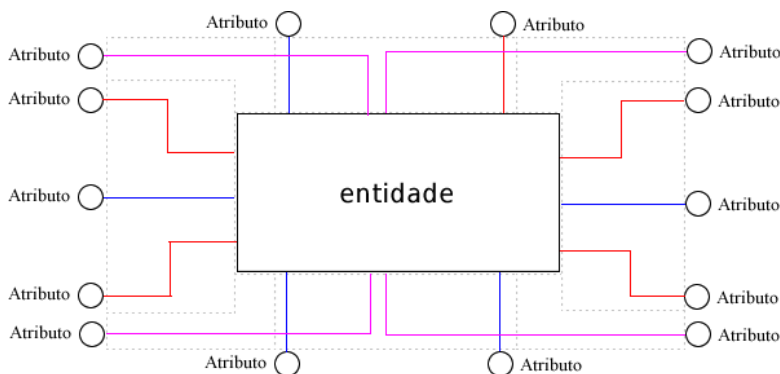


Figura 35 – Ligações de atributos com a entidade.

2.6. Movimentação de componentes

Uma das preocupações no desenvolvimento do software foi garantir o desempenho e evitar travamentos e lentidão quando o usuário desejar mover os componentes na tela. O fato que poderia prejudicar o desempenho são os constantes cálculos usados para posicionar os componentes. Para resolver esse problema, foi desenvolvida uma técnica para que todos os componentes sejam arrastados sem refazer os cálculos. Quando o usuário inicia o processo para arrastar os componentes selecionados, o sistema move todos os componentes para um painel transparente, esse painel transparente fica com a representação gráfica dos componentes e somente ele é arrastado. Quando o usuário termina de arrastar, o sistema move os componentes de volta ao painel. Parte do processo pode ser observado nas figuras Figura 36, Figura 37, Figura 38.

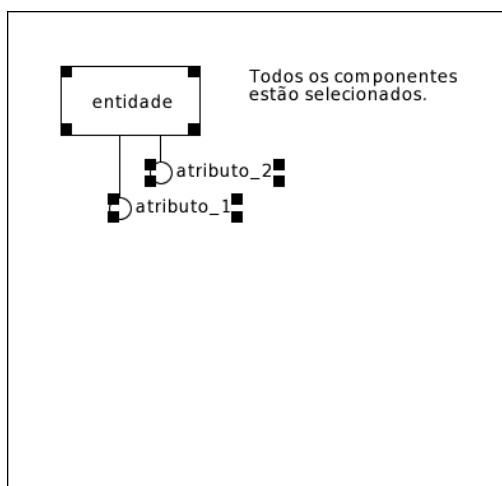


Figura 36 – Selecionando os componentes.

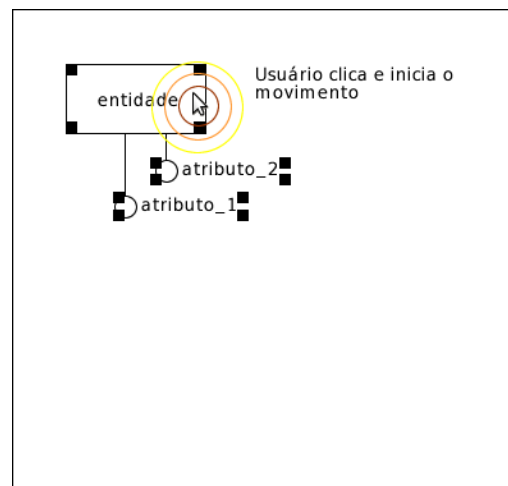


Figura 37 – Clicando sobre a seleção

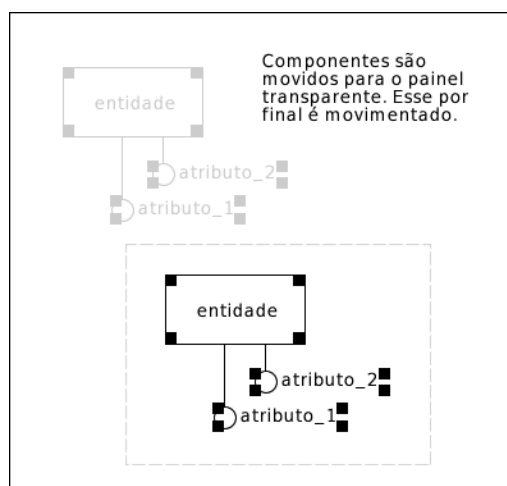


Figura 38 – Arrastando a seleção

2.7. Padrão de arquivo gerado pelo software

O software utiliza o conceito de que os arquivos produzidos ao salvar o projeto devem possuir um formato aberto, pois dessa forma garante que o arquivo possa ser aberto por outra ferramenta que compreenda os padrões desse arquivo. Essa é uma prática geralmente utilizada pela maioria dos softwares livres.

O arquivo é um pacote do tipo zip com vários outros arquivos internos no formato xml. Apesar do tipo dos arquivos serem comuns, as extensões são renomeadas. O esquema completo pode ser observado na Figura 39.

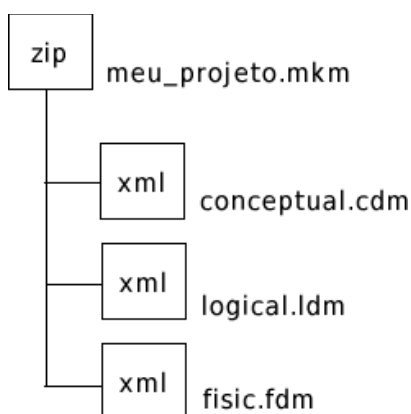


Figura 39 – Estrutura do arquivo gerado pela ferramenta.

2.8. Recurso de desfazer e refazer

O sistema implementa a opção de refazer ou desfazer ações, para isso são utilizados os atalhos de [CTR+Z] para desfazer e [CTR+Y] para refazer. Para que todas as ações do usuário possam ser revertidas, foi utilizados algoritmos de pilha para permitir o empilhamento dos estados do software à medida que vão ocorrendo modificações dos dados por parte do usuário.

Toda vez que o usuário faz uma modificação no modelo, alterando as informações do diagrama, uma cópia dos dados é salva na pilha 'positiva' para manter o histórico de modificações. Esse procedimento pode ser visto na Figura 40.

Quando o usuário solicita que a última ação seja desfeita, o estado atual dos dados é salvo na pilha 'negativa', então é desempilhado o último estado da pilha 'positiva' e torna-se o estado atual, conforme pode ser observado na Figura 41.

Se o usuário desejar refazer, ou seja, voltar ao estado antes de ter solicitado desfazer, o processo compreende a ação de voltar o estado atual para a pilha 'positiva' e desempilhar o último estado da pilha 'negativa' para tornar o estado atual. Observe a Figura 42.

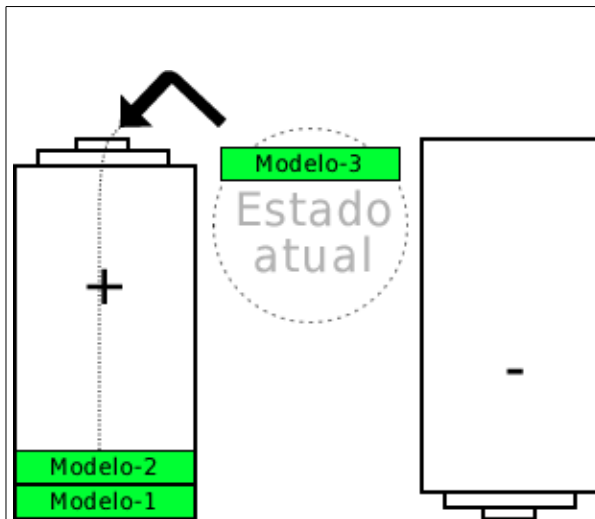


Figura 40 – Empilhando

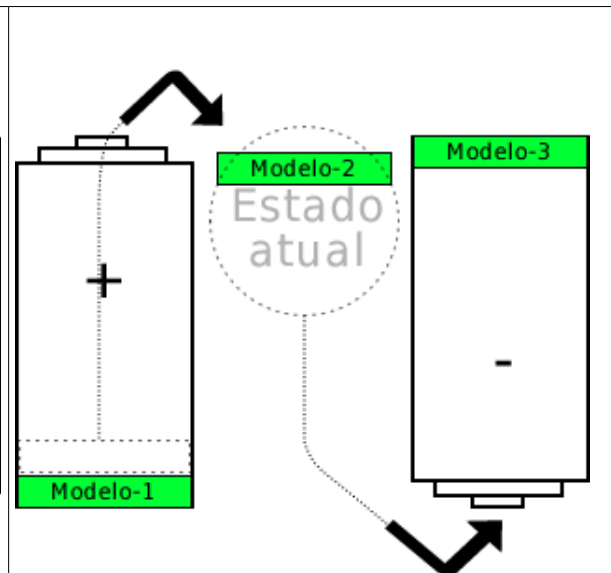


Figura 41 – Desempilhando

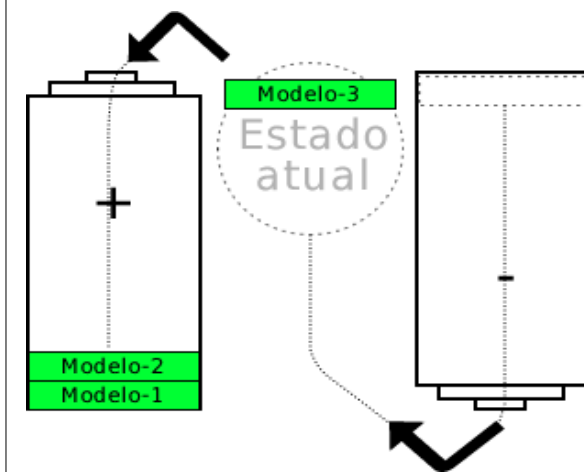


Figura 42 – Reempilhando

2.9. Algoritmo de ligação de atributo com relacionamentos

No diagrama de Entidade Relacionamento os atributos podem estar ligados a um relacionamento quando este representar um relacionamento de muitos para muitos. O processo de cálculo para definir as regras de como ficarão posicionados as linhas que ligam os atributos e o relacionamento são feitos por uma classe específica. Como pode-se ver na Figura

43, há somente quatro áreas ao redor do relacionamento que são passíveis de acomodar os atributos. Os pontos receptores da ligação no relacionamento mantêm uma distância de vinte *pixels* em relação ao vértice do polígono, como também pode ser visto destacado na Figura 43.

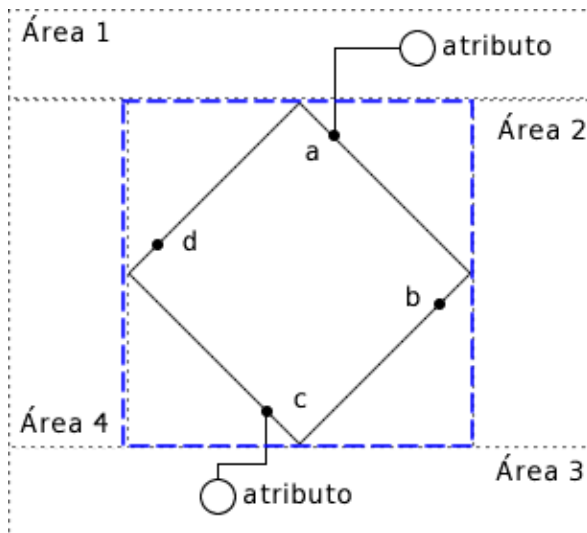


Figura 43 – Áreas para ligação com atributo.

O losango pode ser redimensionado pelo usuário, criando uma figura com tamanhos variados. O algoritmo precisa calcular o tamanho da linha que excede o limite da área, o limite é representado pela linha azul tracejada na Figura 43. Para a realização desse cálculo foi utilizado o teorema de Tales utilizando o conceito de proporcionalidade, como é apresentado na Figura 44.

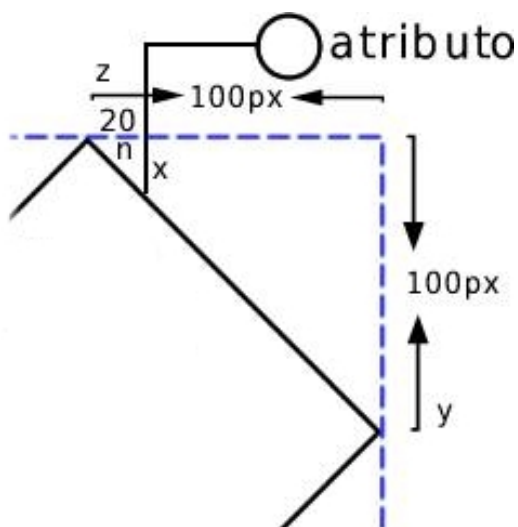


Figura 44 – Triângulos utilizados nos cálculos

A linha entre o atributo e o relacionamento sempre irá cortar a vinte pixels do vértice na reta z. Com essa informação e as demais medidas, é necessário calcular o tamanho da linha x. Para isso aplica-se os cálculos da Figura 45:

$$\begin{aligned}\frac{100}{100} &= \frac{20}{x} \\ 100x &= 20 \cdot 100 \\ 100x &= 2000 \\ x &= \frac{2000}{100} \\ x &= \mathbf{20}\end{aligned}$$

Figura 45 – Teorema de Tales aplicado.

De acordo com os cálculos, podemos concluir que o tamanho da reta x é de vinte *pixels*.

Os cálculos utilizados para posicionar as linhas de ligação entre atributo e relacionamentos são diferentes dos utilizados para ligar atributo à entidade ou para ligar entidade à relacionamento, aproveitando-se apenas alguns princípios.

CAPÍTULO III

Apresentação dos resultados

3.1. A respeito das características

O software desenvolvido como objetivo dessa monografia foi nomeado como Tupiniquim. Esse nome faz uma menção ao Brasil ao se referir a uma tribo indígena característica do Brasil que habitou em grande quantidade a costa do país, no sul da Bahia e litoral do estado de São Paulo.

Após a finalização do desenvolvimento do software foi possível compará-lo com as ferramentas pesquisadas anteriormente, descritas no Capítulo 1. Pode-se observar, no Quadro 3, um resumo dessa comparação.

Quadro 3 – Comparação entre o Tupiniquim e as demais ferramentas pesquisadas

Sistema	Multiplataforma	Gratuito	Modelagem conceitual
DBDesigner	Sim	Sim	Não
CA Erwin	Não	Não	Não
XCase	Não	Não	Não
Dezign	Não	Não	Não
Power Designer	Não	Não	Não
Enterprise Architect	Não	Não	Não
brModelo	Não	Sim	Sim
Visual Paradigm	Não	Não	Sim
Dr.CASE	Não	Não	Sim
DbWrench	Sim	Não	Não
Astah	Sim	Sim	Não
Tupiniquim	Sim	Sim	Sim

1.2. Apresentação do sistema

Na Figura 46 pode ser observado a tela principal do sistema. No caso, há uma modelagem exemplificada na figura, demonstrando alguns dos componentes da modelagem conceitual.

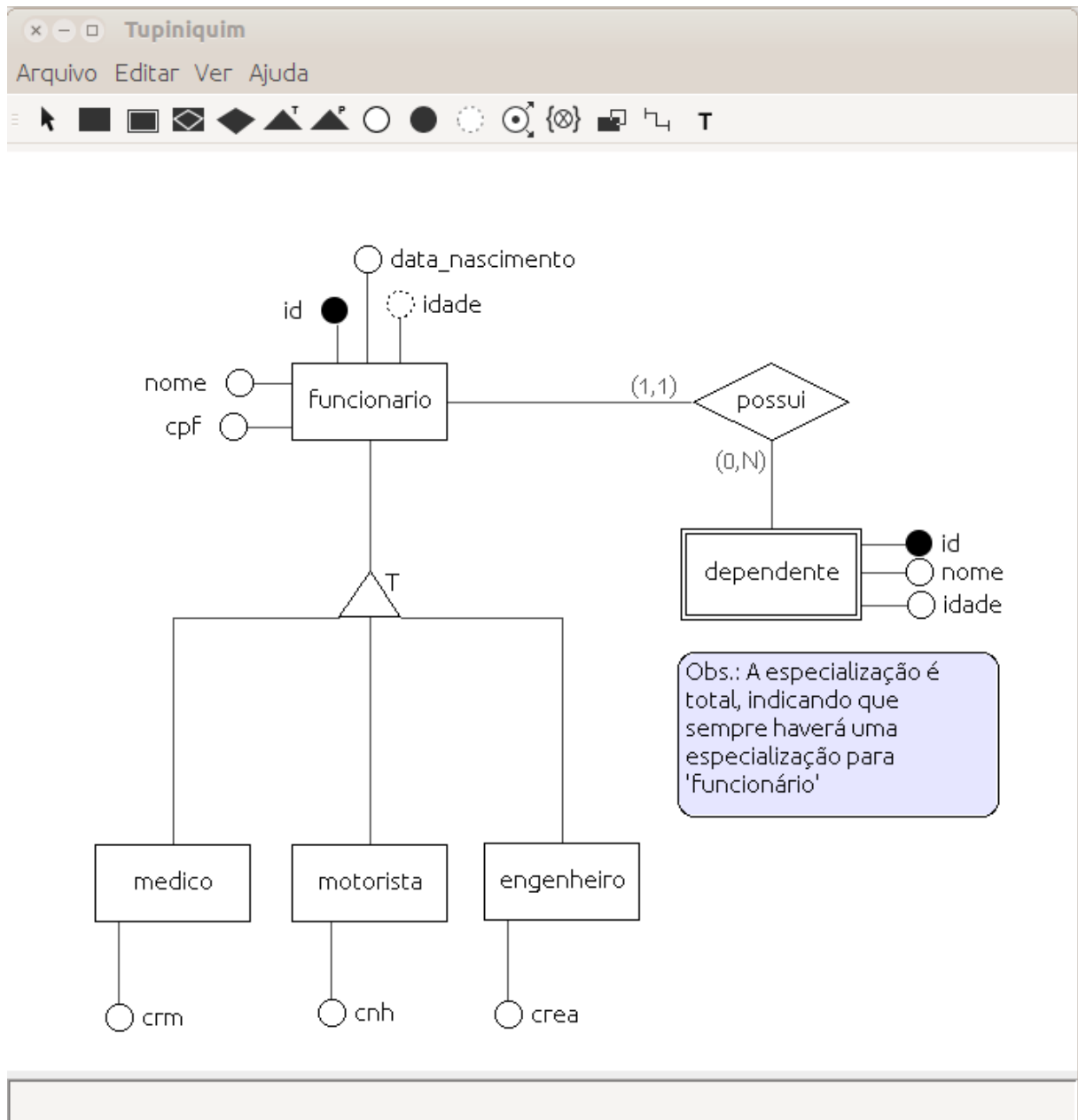


Figura 46 – Tela principal

O menu Arquivo permite as funcionalidades de abrir um arquivo existente, criar um novo arquivo, abrir um arquivo que tenha sido utilizado recentemente, salvar o modelo atual, escolher outro local para salvar o atual modelo (salvar como) e exportar o modelo em uma imagem. Além disso disponibiliza a opção para sair do sistema. Essas opções são exibidas na Figura 47.

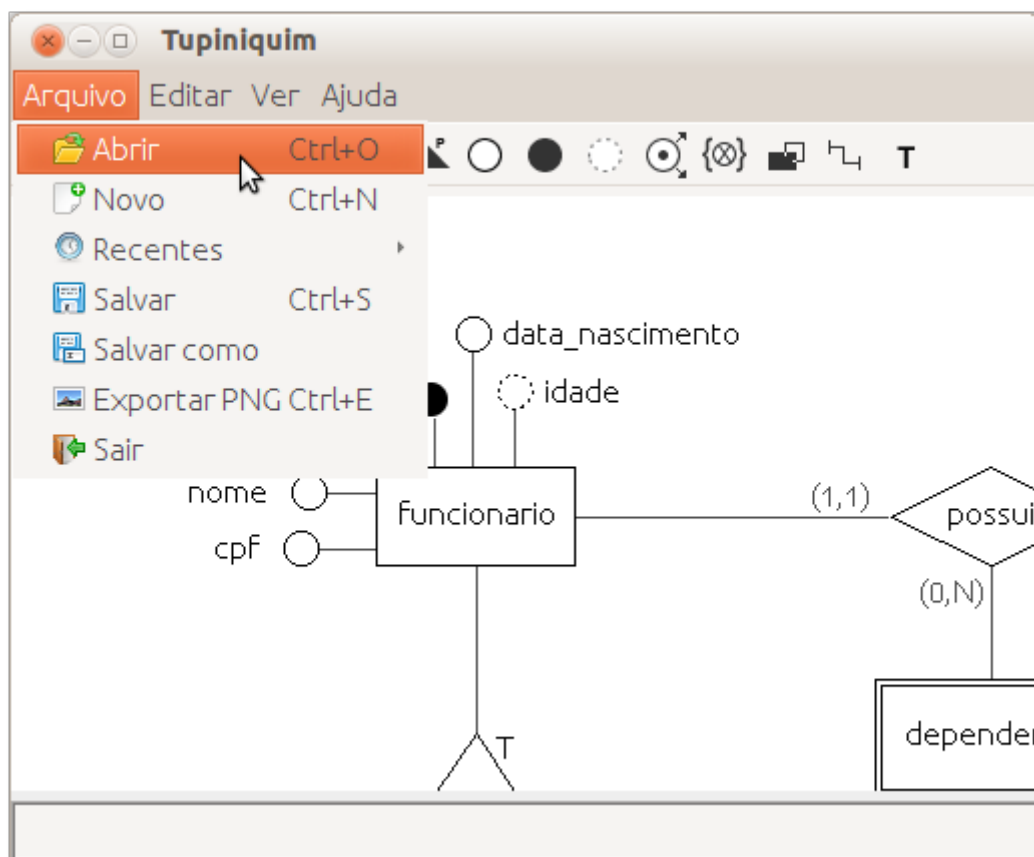


Figura 47 – Menu arquivo do sistema.

O sistema também possui uma barra de ferramentas que pode ser destacado da tela principal, se tornando uma janela independente que pode, até mesmo, ser reposicionada em outra área da janela principal. A barra de ferramentas destacada pode ser observada na Figura 48.

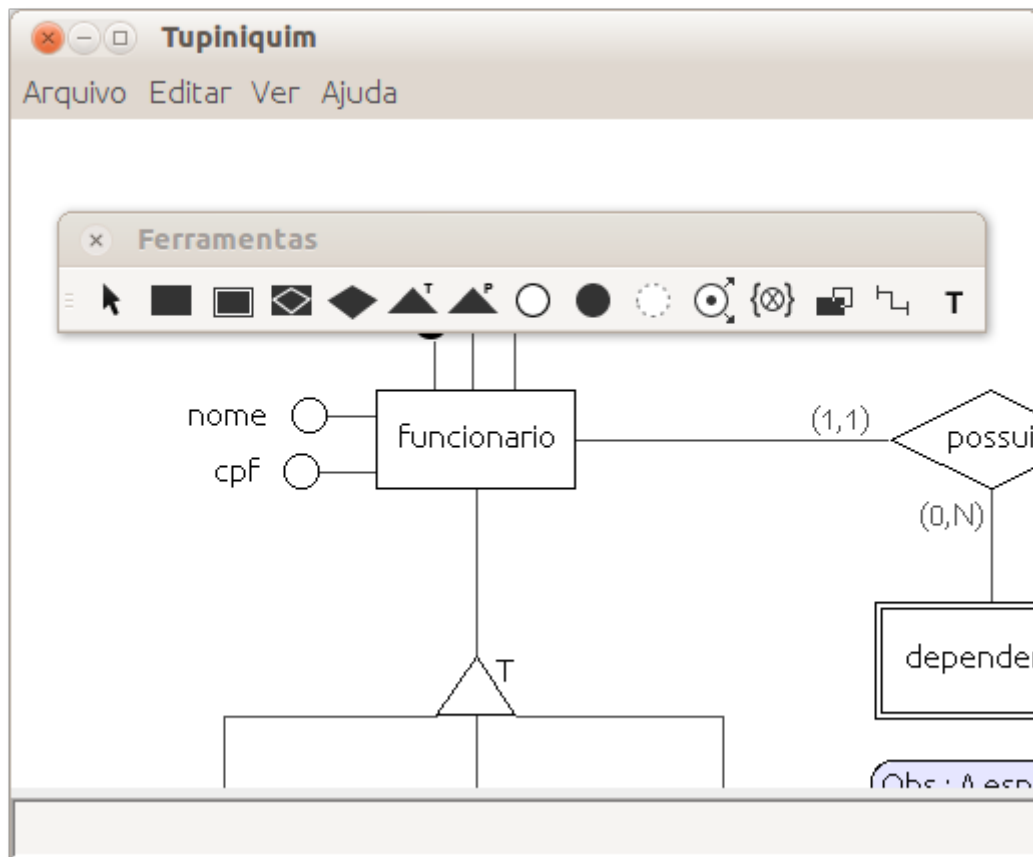


Figura 48 – Barra de ferramentas.

O sistema contém o recurso de adicionar blocos de anotações ao diagrama, permitindo um maior esclarecimento do modelo. Essa característica é apresentada na Figura 49.

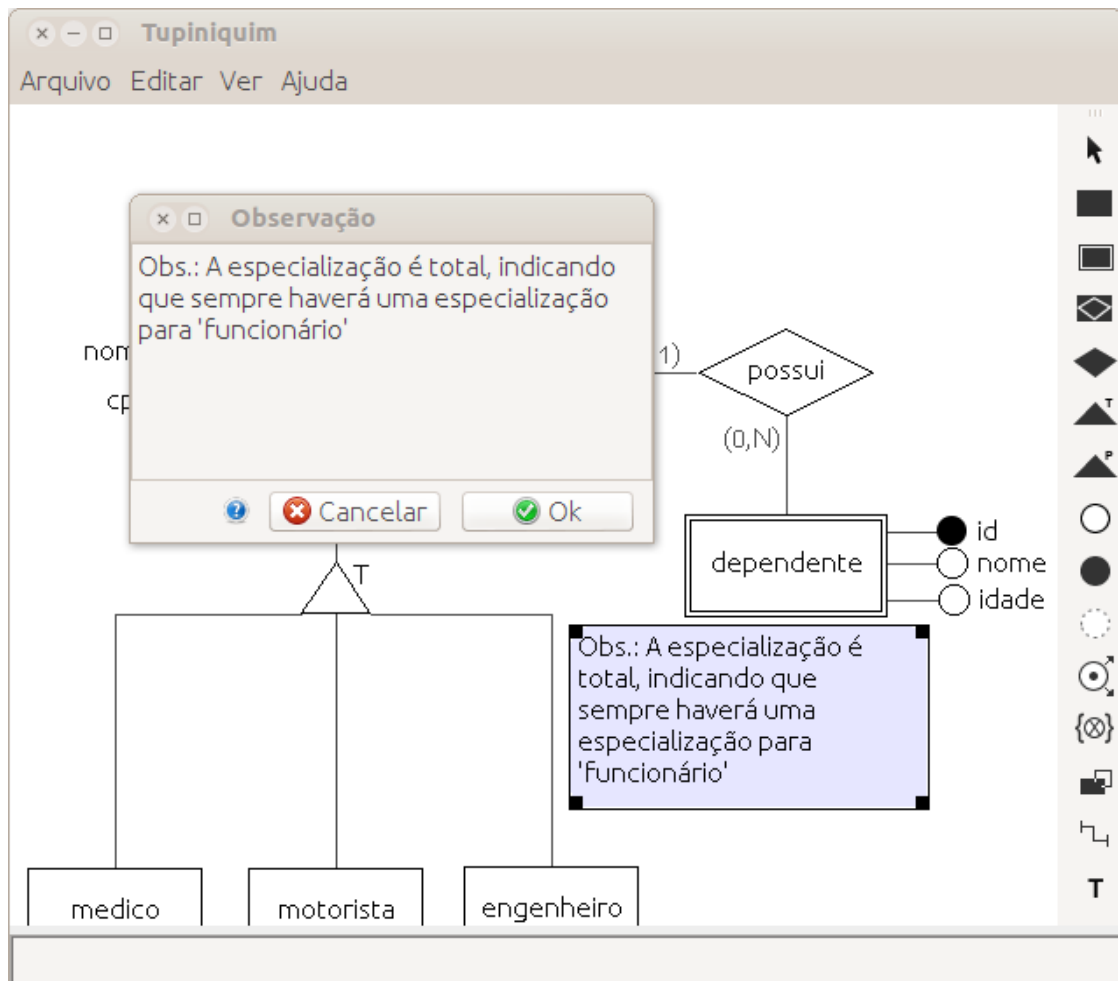


Figura 49 – Blocos de anotações.

CONSIDERAÇÕES FINAIS

Nesse trabalho partiu-se do princípio de que muitos profissionais de banco de dados ou mesmo desenvolvedores precisam no seu dia a dia compreender a estrutura de bancos de dados. Foi constatado que o modelo conceitual cumpre muito bem esse papel por se tratar de uma abstração muito simplificada da estrutura do banco, permitindo entendê-lo de uma forma abrangente.

Foi observada a necessidade que há entre os profissionais em encontrar uma ferramenta que realize modelagem conceitual e que seja um software multiplataforma capaz de executar sobre qualquer sistema operacional ou na maioria deles.

O licenciamento de softwares foi outra questão levantada já que também não se encontra softwares que sejam gratuitos com essas características. A partir disso surgiu a questão de como utilizar os conceitos de modelagem conceitual em uma ferramenta multiplataforma e que seja gratuita?

Com base nisso foi elaborado um projeto para o desenvolvimento de um software para suprir essas necessidades, criando um software que seja gratuito, multiplataforma e que auxilie a criação de modelos conceituais.

Durante o projeto houve várias dificuldades técnicas pelo fato do software trabalhar muito com desenhos vetoriais, o que exige uma grande quantidade de cálculos e domínios da linguagem para criar os diagramas.

Como muito das ferramentas pesquisadas eram estrangeiras, foi considerado o fato de que o software proposto por essa monografia pudesse ser utilizado por profissionais que falem outro idioma, para isso foi desenvolvido a internacionalização do software, o que permite que hoje o software possa ser traduzido para qualquer idioma com o mínimo de esforço.

O software será disponibilizado gratuitamente para download na internet, beneficiando administradores de banco de dados, gerentes de TI, programadores e muitos outros profissionais que possuem contato com banco de dados. Para projetos futuros, espera-se desenvolver outros módulos para trabalhar com modelagem lógica e física de banco de dados, além de traduzir para muitos outros idiomas e correção de erros e melhorias.

REFERÊNCIAS BIBLIOGRÁFICAS

CÂNDIDO, C. H. **www.sis4.com**. Disponível em: <www.sis4.com/brmodelo/Default.aspx>. Acesso em: 31 out. 2012.

DBWRENCH.COM. **DBwrench**. Disponível em: <<http://dbwrench.com>>. Acesso em: 4 nov. 2012.

ELMASRI, R.; NAVATHE, R. B. **Sistemas de banco de dados**. 4º. ed. [s.l.] Editora Pearson, 2005.

GAMMA, E. et al. **Padrões de Projetos**. Porto Alegre: Bookman, 2000.

HEUSER, C. A. **Projeto de Banco de Dados**. 5º. ed. Porto Alegre: Sagra Luzzatto, 2004.

MACHADO, F.; ABREU, M. **Projeto de banco de dados**. 5º. ed. São Paulo: Érica, 1996.

SILBERSHATZ, A.; KORT, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 5º. ed. [s.l.] Elsevier Editora Ltda, 2006.

TEOREY, T.; LIGHSTONE, S.; NADEAU, T. **Projeto e modelagem de bancos de dados**. 4º. ed. [s.l.] Elsevier Editora Ltda, 2009.

WWW.ASTAH.NET. **Astah**. Disponível em: <<http://astah.net/download>>. Acesso em: 5 nov. 2012.

WWW.DATANAMIC.COM. **Dezign**. Disponível em: <<http://www.datanamic.com/dezign/>>. Acesso em: 5 nov. 2012.

WWW.ERWIN.COM. **Erwin**.

WWW.FABFORCE.NET. **Dbdesigner4**. Disponível em: <<http://www.fabforce.net/dbdesigner4/>>. Acesso em: 5 nov. 2012.

WWW.SPARXSYSTEMS.COM.AU. **Enterprise Architect**.

WWW.SQUADRA.COM.BR. **Squadra**. Disponível em:
<<http://www.squadra.com.br/website/produtos/ferramentas/cadastro/cadastroDC.html>>.
Acesso em: 5 nov. 2012.

WWW.SYBASE.COM.BR. **Powerdesigner**. Disponível em:
<<http://www.sybase.com.br/products/modelingdevelopment/powerdesigner>>. Acesso em: 5
nov. 2012.

WWW.VISUAL-PARADIGM.COM. **Visual-paradigm**.

WWW.XCASE.COM. **XCase**.