Universidade Federal de Pernambuco

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

# Uma Ferramenta Case para Modelagem de Bases de Dados Objeto-Relacionais

**Aluno:** Rubem Rodrigues Moreira Bisneto {rrmb at cin.ufpe.br}
**Orientador:** Fernando da Fonseca de Souza {fdfd at cin.ufpe.br}

Dezembro, 2011

Universidade Federal de Pernambuco

CENTRO DE INFORMÁTICA

**Rubem Rodrigues Moreira Bisneto**

# Uma Ferramenta Case para Modelagem de Bases de Dados Objeto-Relacionais

*Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.*

***Orientador:*** *Prof. Dr. Fernando da Fonseca de Souza*

Recife, dezembro de 2011

*“Aqueles, a quem os deuses amam*
*crescem jovens”.*
***(Oscar Wilde)***

***A quem sempre me apoia, vóvi.***

# Agradecimentos

Agradeço a Deus, apesar da minha pouca fé tenho certeza que muita coisa foi facilitada por uma ajuda extra. A minha família que sempre esteve presente comigo quando a coisa ficou séria de verdade. Agradeço a minha mãe que sempre sofreu comigo e tenho certeza que se soubesse programar em alto nível me acompanharia em todas as noites e finais de semana “perdidos” graças aos projetos do CIn. Agradeço ao meu pai pelo suporte e apoio forte. Às minhas irmãs que sempre compartilharam o cargo difícil de filho dos meus pais. À vóvi que me faz encher a boca e bater no peito pra dizer que sou criado por vó. E aos meus padrinhos que servem de referência para meu crescimento e por terem acolhido não só a mim, mas também a meus amigos quando estávamos na Espanha.

Ao professor Fernando da Fonseca por ter compartilhado seu tempo comigo e esse trabalho.

Aos meus amigos dos diversos grupos, aos que passaram pela minha vida e seguiram rumos diferentes e aos que continuam ao meu lado. O grupo da faculdade que sofreu tanto comigo, especialmente a Carol minha dupla eterna em sofrimento e Jota que não me deixou desistir do curso no quarto período. A toda nação Broka que sucumbe às constantes investidas feministas, regida atualmente por uma regência trina permanente. E também aos exilados na Pega Ninguém Lândia. Aos meus amigos da antiga universidade de sarcasmo presentes nos tempos de Pokémon, tíbia, *dominalcóol* e caminhos de cadeira. E aos meus amigos do GGE que sofreram comigo no vestibular, me desviando das festas e de Porto Seguro (Nerds).

Às mulheres da minha vida, Nana e Patrícia. E à menor e mais chata japonesa do mundo, minha rocha forte. Não menos importante ao meu tutor Dorus por ter me emprestado sua força. “Nenor, nenor”.

# Resumo

Os SGBD Relacionais substituíram os sistemas de armazenamento em Arquivos Plenos. Utilizando-se de armazenamento em tabelas e utilizando comandos em inglês para inserir e manipular dados, esses sistemas dominaram rapidamente o mercado, permanecendo líderes ainda hoje.

Pelo grande sucesso, uma série de ferramentas foram desenvolvidas para auxiliar diversos aspectos de gerência de bases de dados. As funcionalidades providas por estas ferramentas estendem-se de funcionalidades simples que permitem realizar apenas diagramação do modelo da base, à ferramentas de visualização de tabelas, podendo-se modificar as mesmas sem utilizar a sintaxe SQL.

Porém, algumas limitações apresentam-se na manipulação de tipos complexos de dados. Surge o paradigma Orientado a Objetos que aponta algumas desvantagens no paradigma Relacional. Esse novo paradigma sofre com muitas desvantagens também, como o alto custo de buscas. A solução encontrada pelos grandes fabricantes de bases de dados foi estender as bases relacionais a conceitos de orientação a objeto. É um novo paradigma que mescla o melhor dos dois mundos, o paradigma Objeto-Relacional.

Apesar de bastante promissoras, essas novas bases sofrem com a ausência de ferramentas CASE específicas para modelagem e gerenciamento. Esse trabalho propõe a criação de uma ferramenta CASE que dê suporte ao processo de modelagem e construção inicial de uma nova base. Essa ferramenta provê uma forma de criar um diagrama ER Estendido que pode ser convertido em código de linguagem de quarta geração, por exemplo, PL/SQL, seguindo o paradigma Objeto-Relacional.

**Palavras-chave**: *ER Estendido, Orientado a Objeto, Relacional, Objeto-Relacional, Ferramentas CASE, SGBD, SGBDOR*

# Abstract

The Relational DBMS has replaced storage on Plain Archive systems. Using tables for storing data and making use of English language commands to insert and manipulate data, these systems had taken a quick ascension and had taken over the market, maintaining their leadership until today.

For their great success a large variety of tools were developed to help a bunch of aspects in management of databases. Their functionality go from simple tools that allow diagram drawing of the base model, to more complex tools of table viewing which allows the change of data without a command in SQL syntax.

Certain limitations appear in handling more complex data types. The Object Oriented paradigm comes into being to show some disadvantages from the Relational Paradigm. But this new paradigm also has a lot of disadvantages such as the high cost to perform queries. The resolution proposed by the greater database vendors was to extend the relational database to Object Oriented concepts. The Object-relational paradigm, thus, mixes the best features of those two worlds.

Although very promising, this new system suffers from the lack of specific CASE tools for modeling and maintenance. This work proposes the creation of a CASE tool that supports the modeling process and the initial building of a new database. This tool provides a new way of creating Extended ER diagrams that can be converted lately in fourth generation language code such as PL/SQL following the Object -Relational- paradigm.

**Key-Words**: *Extended ER, Object Oriented, Relational, Object - Relational-, CASE Tools, DBMS, ORDBMS*

# Lista de Figuras

# Lista de Quadros

# Sumário

# Capítulo 1 - Introdução

Esse capítulo descreve os motivos para a criação da ferramenta, os objetivos deste trabalho e apresenta a estrutura da monografia. Na seção 1.1 será discutido o contexto ao qual a ferramenta deve enquadrar-se. A motivação surge logo em seguida na seção 1.2. Analisando esses dois tópicos a seção 1.3 descreve os objetivos da ferramenta. Para finalizar na seção 1.4 está exposta a estrutura na qual está organizado este trabalho.

## 1.1 Contextualização

Gestores do mundo corporativo estão sempre acessando dados sobre seus negócios para tomada de decisões de forma rápida e eficiente. Os sistemas de banco de dados surgem em meados da década de 70 (Sanches, 2005), porém começam a substituir o sistema de arquivo pleno, somente, alguns anos depois. Desde então são a principal forma de armazenar e recuperar dados.

Atualmente, o sistema de banco de dados mais utilizado é o sistema relacional (Feitosa, Jesus, Machado, & Alencar, 2006). Trata-se de um conceito abstrato que define maneiras de armazenar, manipular e recuperar dados estruturados unicamente na forma de tabelas (Sanches, 2005). Uma variedade de ferramentas está prontamente disponível ao administrador da base. Porém, quando a complexidade dos dados aumenta, o esquema relacional deixa a desejar em alguns aspectos. Aplicações complexas como bases de dados para telecomunicações, sistemas de informação geográfica e multimídia manipulam objetos não triviais que exigem sistemas de armazenamento mais complexos que os SGBD Relacionais (Nahouraii & Petry, 1991).

Surgem então os sistemas de bancos de dados orientados a objeto. Apesar da promessa de prover suporte para bases mais complexas e elementos não triviais, enfrentaram problemas como a otimização de consultas e a dificuldade de ingresso em um mercado dominado pelos sistemas de bancos de dados Relacionais (Edelweiss & Galante). Porém, sua chegada fez acelerar o processo de evolução dos sistemas relacionais. Com o advento de SQL3 (Eisenberg & Melton, 2001), várias operações são criadas, baseadas em SGBD relacionais, a fim de prover suporte a algumas características de orientação a objetos. Herança, encapsulamento de operações e construtores de tipos são algumas das extensões implementadas por esses sistemas (Devarakonda, 2001). Essas bases são então chamadas de bases Objeto-Relacional e

mesclam o conceito de orientação a objeto em um sistema de bases consolidado no mercado, livre das limitações enfrentadas pelas bases puramente orientadas a objeto.

## 1.2 Motivação

As bases objeto-relacionais são relativamente novas no mercado. Por isso a variedade de ferramentas desenvolvidas para auxiliar o processo de criação é bastante pequena e as ferramentas existentes provêm pouquíssimo suporte específico. Na maioria dos casos, o administrador de base de dados deve utilizar-se de uma ferramenta para criação de modelos gerais como o Microsoft Visio (Visual Studio 2010).

Esse problema é facilmente contornado por administradores experientes de bancos de dados, utilizando-se diagramas UML de classe, podendo facilmente adaptar um modelo. Porém, no âmbito acadêmico pode dificultar a ação dos iniciantes que sentem falta de uma ferramenta mais específica para modelagem e entendimento da base. O artigo do doutor Wang (Wang, 2001) discute vantagens em deixar bem claro os conceitos de Bases Objeto-Relacionais aos estudantes:

- Demonstrar algumas limitações das bases relacionais;
- Permitir aos estudantes resolver problemas mais complexos em suas carreiras futuras; e
- Aproveitar-se de uma das maiores vantagens de se estender o modelo relacional, que é o reuso.

Entre outros motivos, uma ferramenta que ilustre o conceito de modelagem e criação de código básico, ajudará uma maior fixação dos conceitos apresentados no nível acadêmico.

## 1.3 Objetivos

O objetivo do Trabalho de Graduação aqui proposto é criar uma ferramenta CASE específica para dar suporte à modelagem de bancos de dados objeto-relacionais. Essa ferramenta deverá prover suporte para criação de um Modelo ER Estendido que mais tarde deverá ser mapeado em uma estrutura Objeto-Relacional, através da geração automática de scripts em linguagem de quarta geração, tendo como exemplo PL/SQL, para Oracle (Oracle, 2011). Com interface simples e intuitiva, essa ferramenta deve poupar o tempo gasto por administradores com tarefas desnecessárias, como especificar o esquema gerado por uma ferramenta generalista.

Um segundo objetivo, mas não menos importante, é a utilização futura do programa aqui apresentado no auxílio do aprendizado de bases objeto-relacionais, particularmente na disciplina *Gerenciamento de Dados e Informações* da Universidade Federal de Pernambuco.

## 1.4 Estrutura do Documento

Os demais capítulos desse trabalho seguem apresentados de acordo com tópicos específicos e são organizados da maneira que se segue.

No Capítulo 2 será apresentada uma visão geral de sistemas de bancos de dados, dando ênfase às bases de dados Objeto-Relacional.

O Capitulo 3 destacará a importância das ferramentas *CASE* no processo de construção de software.Também analisará algumas das ferramentas existentes utilizadas em âmbito comercial e âmbito acadêmico.

O Capítulo 4 é destinado a descrever a especificação e o processo de implementação do projeto. Serão apresentados casos de uso, requisitos informais e uma arquitetura simplificada do projeto de software.

O Capítulo 5 apresenta um estudo de caso para demonstrar o uso da ferramenta em um determinado domínio de problema.

No Capítulo 6 serão, então, expostas as conclusões, destacando as contribuições e limitações deste trabalho. Por fim, sugestões de projetos futuros serão apresentadas baseadas nas limitações do mesmo.

Finalmente, serão descritas as referências utilizadas para o desenvolvimento deste trabalho.

# Capítulo 2 -Fundamentação Teórica

Este capítulo aborda os principais conceitos necessários para o entendimento do trabalho detalhado neste documento, além de discorrer sobre os principais estudos correlatos. Na Seção 2.1, é apresentado um breve histórico dos sistemas de bases de dados, onde cada sessão cria um link com a próxima destacando a necessidade de evolução desses sistemas. A referida sessão contemplará aspectos pertinentes tanto à comunidade acadêmica, quanto à comunidade comercial. Ela terá como destaque principal, no entanto, as bases de dados Objeto-Relacionais. A sessão 2.2 destacará, então, sistemas de bases de dados OR, destacando suas facilidades e demonstrando porque ao final prevaleceram em relação às bases orientadas a objeto. As funcionalidades OR descritas nessa sessão serão específicas para o sistema de gerenciamento de banco de dados (SGBD) líder do mercado, Oracle (Oracle, 2011). Na sessão 2.3 será discutida uma solução de mapeamento Relacional em Objeto-Relacional. Por fim, na Seção 2.4, conclusões acerca do que foi exposto neste capítulo são apresentadas.

## 2.1 Histórico dos Sistemas de Bases de Dados

Reunir dados sempre foi vital para o mundo corporativo. Era entendido que todos esses dados, que representavam desde simples transações comerciais a históricos de recursos humanos (RH), deveriam ser armazenados. Até meados de 60 tais dados eram mantidos aleatoriamente em arquivos (Boscarioli, Bezerra, Benedicto, & Delmiro, 2006) e as empresas começavam a achar custoso empregar grande número de pessoas para armazenar e indexar esses arquivos (Sanches, 2005). Foi no ano de 1970 que um pesquisador da IBM, Ted Codd publicou o artigo “A Relational Model of Data for Large Shared Data Banks” no qual visionava um sistema onde o usuário seria capaz de criar e acessar informações contidas em tabelas através de comandos em inglês (Ted Codd).

### 2.1.1 Sistemas de Bases Dados Relacionais

Apesar do trabalho de Ted Codd, a IBM não implementou de imediato a solução. As idéias de Codd tiveram vida em 1976 pela Honeywell Information Systems INC (Sanches, 2005). No entanto, é somente nos anos 80 que através do Oracle 2 (Features Introduced in the Various Server Releases), a Oracle constrói um sistema

baseado nos padrões SQL (Features Introduced in the Various Server Releases).

Uma base relacional é composta de n relações agregadas em tabelas bidimensionais que armazenam tuplas. Cada linha é nomeada um registro e cada coluna é composta por um tipo especial e representa um atributo.

Os SGBD utilizam-se de linguagem de consulta estruturada de alto nível, SQL. Essa linguagem é tão robusta que possuí comandos para definição de dados, consulta e tratamento de restrições (Elmasri & Navathe, 2004). As consultas podem ser simples, referenciando apenas uma tabela, a consultas multi-tabelas envolvendo operadores específicos de junção a aninhamento (Devarakonda, 2001). Essas bases permitem acesso rápido aos dados e grande capacidade de armazenamento (Relational Database). Esses benefícios aliados ao padrão SQL, que permite fácil migração de uma base a outra, tornaram esses sistemas de bases de dados, praticamente, obrigatórios.

Tanto em âmbito acadêmico quanto comercial, as bases relacionais ainda são o tipo de base mais empregada (Feitosa, Jesus, Machado, & Alencar, 2006). São exemplos de SGBD, o Oracle (Oracle) e o Microsoft SQL Server (SQL Server 2008).

### 2.1.2 Sistemas de Bases de Dados Orientadas a Objeto

Apesar de toda a revolução e impacto causado pelos SGBD relacionais, há algumas limitações nos mesmos, quando algumas aplicações mais complexas precisam ser projetadas e implementadas, como bancos que armazenam informações geográficas e multimídia. O que acontece de fato é que essas novas aplicações têm requisitos e características peculiares como estruturas complexas para objetos e transações de longa duração (Chaudri & Zicari, 2001).

Os Sistemas de Gerenciamento de Banco de Dados Orientados a Objetos (SGBDOO) utilizam-se de um modelo de dados que dá suporte a características de orientação a objetos, tais quais: polimorfismo, herança, encapsulamento e abstração de tipos (Elmasri & Navathe, 2004). Contam com identificadores únicos de objetos e aproveitam-se do poder desse paradigma para dar excelente poder de programabilidade no nível de banco de dados (Devarakonda, 2001).

Os dados nesse tipo de base são gerenciados por dois conjuntos de relações, uma que descreve as inter-relações entre itens de dados e outra descrevendo as relações abstratas (herança) (Rao, 1994). Ao encapsular os métodos de cada conjunto, cria-se uma forte ligação direta entre aplicação e base de dados o que significa menos código, mais estruturas de dados naturais e melhor manutenção e re-usabilidade (Nahouraii &

Petry, 1991).

Para funcionamento é necessário que (Devarakonda, 2001):

- Exista um modelo de objetos, que consiste nos tipos de dados e construtores;
- Uma linguagem de definição de objeto (ODL), que é independente de linguagens de programação e tem como função dar suporte a construtores semânticos;
- Uma linguagem de consultas (OQL), similar a SQL, porém com funções adicionais tais como manipulação de objetos complexos e implementar polimorfismo; e
- Interfaces para linguagem de programação.

Tudo isso é mantido pelo padrão ODMG 3.0 (ODMG).

No entanto, SGBDOO estão longe de ocupar o lugar que os sistemas se bases de dados relacionais ocupam. Algumas das razões para a não popularidade de adoção dessas bases estão ligadas à cultura empresarial, onde devido a custos, tempo e desvantagem perante o número de ferramentas, há hesitação de mudança tão brusca (Weidler, 2004). O pobre desempenho, problemas de escalabilidade e dificuldade em otimização de consultas (Elmasri & Navathe, 2004) foram os principais responsáveis por esse tipo de base não vir a consolidar-se consolidar no mercado.

### 2.1.3 Sistemas de Bases de Dados Objeto-Relacionais

Então, a essa altura, sistemas baseados no modelo relacional já eram amplamente empregados. Descartar as diversas aplicações seria inviável, porém era necessário utilizar-se de alguns dos benefícios propostos pelo modelo OO, como por exemplo o suporte para tipos ricos . Os Sistemas de Gerenciamento de Banco de Dados Objeto-Relacionais (SGBDOR) representam uma progressão natural baseada nos benefícios da combinação entre uma base de dados relacional com base de dados orientada a objeto (Fortier, 1999). Sendo mais específico, a implementação física é feita na forma relacional enquanto a semântica de aplicação é modelada e representada por objetos (Feitosa, Jesus, Machado, & Alencar, 2006). É a partir de SQL3 (Eisenberg & Melton, 2001) que essa implementação sobre sistemas de bancos relacionais foi possível. Entre outros aspectos, SQL3 dá suporte a tratamento de objetos, consultas recursivas e instruções de programação (Weidler, 2004).

Para auxiliar ainda mais o crescimento dos SGBDOR, muitos fabricantes de

sistemas de bases de dados, preocupados com as fraquezas "recém-descobertas" das bases do modelo relacional, começam a adotar essa tecnologia. E são justamente os líderes de mercado, Oracle, Informix e IBM (Wang, Implementation of Object-Relational DBMSs, 2001).

Uma estimativa arriscada, realizada pelo Dr. Michael Stonebraker, chefe de tecnologia na Informix, prediz que pouco a pouco os antigos sistemas relacionais vão migrar para esse novo paradigma. Um exemplo utilizado é de uma companhia de seguros que já tem uma base bastante carregada, mas a partir de determinado período quer adicionar fotos de acidentes e também coordenadas geográficas. A solução proposta é a simples extensão para o modelo objeto-relacional (Stonebreaker)

Uma melhor comparação entre as bases abordadas é mostrada no Quadro 2.1, adaptada de (McClure, Agosto/97).

Quadro 2. 1 Comparações entre as Bases

| **Critério** | **RDBMS** | **OODBMS** | **ORDBMS** |
|---|---|---|---|
| Padrão | SQL 2 em diante | ODMG (Versão atual em 3.0) | SQL 3 em diante |
| Suporte à orientação a objeto | Não dá suporte, e trás certa dificuldade em mapeamento | Dá suporte extensivamente | Suporte limitado, porém satisfatório |
| Usabilidade | Fácil | Bom para programadores, algum acesso SQL a usuários finais | Fácil uso, exceto por algumas extensões |
| Suporte para relações complexas | Não dá suporte a tipos abstratos | Suporte a grande variedade de tipos e dados com relações internas complexas | Suporte tipos abstratos e relações complexas |
| Desempenho | Muito boa | Relativamente menos | Desempenho acima do esperado |
| Maturidade do produto | Relativamente velho e bem maduro | Conceito de alguns anos nem tão maduro | Conceito novo, mas amadurecendo depressa |

| | | | |
|---|---|---|---|
| Uso de SQL | Suporte extenso | OQL é similar a SQL, mas carrega funções adicionais como objetos complexos e OO. | Começa a partir do SQL 3, amplo suporte |
| Vantagens | Facilidade de otimização de consultas e desempenho | Pode manipular todo tipo de aplicações complexas e reuso de código | Consultas em aplicações complexas e lidar com largas aplicações |
| Desvantagens | Inabilidade de lidar com aplicações complexas | Baixo desempenho pela dificuldade de otimização de consultas | Alto custo de implementação |

## 2.2 Funcionalidades de SGBDOR

Serão abordadas nos tópicos seguintes as funcionalidades de bases Objeto-Relacionais que serão implementadas na ferramenta case produzida neste trabalho. A metodologia utilizada será partir do conceito mais básico e seguir evoluindo a sintaxe de um objeto a ser representado no SFBDOR Oracle. Quando necessário, imagens demonstrando um exemplo prático serão utilizadas como ilustração. Ao final de cada sessão, a sintaxe terá novos campos adicionados.

### 2.2.1 Tipos Abstratos de Dados - TAD

Novas aplicações de bases de dados requerem que se armazenem objetos complexos. Trata-se de objetos mais próximos da realidade, com um conjunto bem maior de atributos. Em muitos casos, torna-se difícil definir tal complexidade através de somente um registro (Rao, 1994). Os tipos abstratos são a resposta para a implementação desses objetos, os quais são análogos às *Structs* da linguagem de programação C (Structs C). São compostos por:

- Nome (Identificador do tipo);
- Atributos (Tipos primitivos, tipos definidos por usuarios e também coleções de tipos); e
- Métodos (*Functions* e *Procedures* que rodam em tempo de compilação,

definidos dentro ou fora do TAD; e Métodos que rodam em tempo de execução). Logo, tem-se, inicialmente, uma sintaxe de criação de TAD incompleta:

```
CREATE [OR REPLACE] TYPE new_type AS OBJECT(
   [lista de atributos]
   [lista de métodos]
);
```

Figura 2. 1 Sintaxe de Create Type Básica

### 2.2.2 Coleções de Objetos

No modelo relacional, os atributos multivalorados deveriam ser manipulados para formar uma nova tabela (Elmasri & Navathe, 2004). Logo se em uma tabela qualquer quatro (4) de seus elementos fossem multivalorados, após sua normalização ter-se-ía cinco (5) tabelas. E, somente para recuperar os dados, um desenvolvedor realizaria quatro (4) junções.

Em um modelo de objetos isso não seria necessário, já que todos os elementos estão encapsulados dentro do objeto. Foram criadas algumas estruturas para lidar com esse tipo de atributo e relacionamentos 1:m, entre eles pode-se destacar: *set*, *list*, *multiset* e *arrays* (Baptista, 2009). Dessa forma, as consultas são extremamente simplificadas, evitando o uso de junções.

A partir do Oracle 8 (Features Introduced in the Various Server Releases), uma das formas de manipular esses tipos é uma estrutura *array* de tamanho variável, o *VARRAY* (Elmasri & Navathe, 2004). O *VARRAY* possui tamanho definido; é composto de um conjunto de elementos de índices ordenados de mesmo tipo e é armazenado na mesma estrutura da tabela (Fonseca, Neto, Souza, & Dourado, 2007). Contém duas propriedades:

- *Count* - Número atual de elementos; e
- *Limit* - Número máximo de elementos pré-definido pelo usuário.

Segue a sintaxe de criação de um *VARRAY*:

```
CREATE [OR REPLACE] NomeVarray AS VARRAY(Tamanho)
 OF TipoDeObjeto;
```

Figura 2. 2 Sintaxe de Criação Varray

Outra forma implementada pelo Oracle para essa solução foi a utilização de tabelas aninhadas. Ambas as estruturas possuem funções semelhantes com poucas diferenças. As tabelas aninhadas são armazenadas em uma tabela à parte, por isso não possuem limite superior para números de elementos; cada item individual pode ser recuperado e ainda podem ser definidos índices adicionais para acelerar o acesso aos dados, o que pode melhorar a eficiência de consultas (Chaudri & Zicari, 2001). Sua sintaxe de criação é a seguinte (Elmasri & Navathe, 2004):

```
CREATE [OR REPLACE] NomeNestedTable AS TABLE
 OF TipoDeObjeto;
```

Figura 2. 3 Sintaxe de Criação Nested Table

É importante destacar que tanto os *VARRAYS* como as tabelas aninhadas são compostas por um e apenas um tipo de elemento, seja ele complexo ou simples. Cabe ao usuário decidir qual dos dois utilizar. Os *VARRAYS* são eficientes quando é necessário acessar um elemento pelo índice e preservar as ordens dos elementos. Já as *NESTED TABLES* são mais recomendadas quando se precisa de mais flexibilidade e quer aumentar-se a eficiência de consultas sobre coleções (Varrays e Nested)

A Figura 3.1 ilustra o funcionamento com um contra-exemplo de como utilizar os dois tipos de estruturas descritas nessa sessão. Após a criação do objeto que será armazenado nessas estruturas decide-se por utilizar um *Varray* <1> ou *Nested Table* <2>. Em seguida, ao criar o objeto que encapsulará essa propriedade, declara-se um atributo de um desses dois tipos.

```
CREATE TYPE tipo_varios AS OBJECT (
elemento CHAR(10)
);

<1>
CREATE TYPE varray_Varios as VARRAY(5)
of tipo_varios;
<2>
CREATE TYPE nested_Varios as TABLE
of tipo_varios;

CREATE TYPE uso_Varios as OBJECT (
nome_Pessoa VARCHAR(30),
[elementoV_varios varray_Varios] <1>
[elementoT_varios nested_Varios] <2>
);
```

Figura 2. 4 Exemplo de Uso Nested e Varray

### 2.2.3 Tabelas/ Identificadores de Objeto /Referências

Os TAD são apenas moldes de objetos, logo não há armazenamento dos mesmos nas bases de dados. Para isso, é necessário criar ao menos uma tabela do tipo em questão. Essas tabelas provêm uma visão relacional dos tipos abstratos. Nelas, cada linha representa uma instância de objeto. Por estender sistemas de bancos relacionais, essas tabelas permitem inserção, remoção, busca e alteração dos seus elementos, tal qual como *triggers* e *constraints* (Baptista, 2009).

A declaração de tabelas é idêntica à declaração de uma tabela aninhada como exibida na Figura 2.3, mas, diferente de uma *nested table*, cada tabela pode conter outros tipos como colunas, basta declará-las de forma idêntica a uma declaração de tabela de SGBDR.

Uma tabela definida com base em um TAD difere das tabelas convencionais do modelo relacional sob os seguintes aspectos (Edelweiss & Galante):

- Cada linha da tabela contém um OID que é um identificador único para o objeto; e
- As linhas podem ser referenciadas por outros objetos do banco de dados.

Os OID são automaticamente atribuídos pelo SGBDOR quando um objeto é armazenado em uma tabela. Eles servem como ponteiro lógico, ou seja uma referência a colunas em outras tabelas (Chaudri & Zicari, 2001). Normalmente o OID não é visto

pelos usuários, mas em alguns sistemas pode ser bastante útil tratá-lo como um atributo normal (Fonseca, Neto, Souza, & Dourado, 2007).

No modelo objeto-relacional, um atributo pode referenciar diretamente a outro objeto. O domínio desse atributo consiste em um ponteiro lógico para um objeto de um determinado tipo, desde que esse objeto possua uma OID. Dessa forma, só é possível referenciar a objetos armazenados em tabelas (Feitosa, Jesus, Machado, & Alencar, 2006). Difere-se de uma chave estrangeira, pois não é uma comparação de valores identificadores, e sim, uma referência direta à tabela em questão. Também difere no quesito composição, já que chaves estrangeiras podem ser compostas e referências não (Neves, Rocha, & Segundo, 2008). A sintaxe para declarar um atributo do tipo Referência em um TAD é descrita na figura 2.5:

```
<nomeAtributo> REF <tipoObjeto>
```

Figura 2. 5 Sintaxe de REF

Quando mais de uma tabela pertence a um mesmo tipo, pode ser interessante para o administrador da base de dados restringir o escopo de referências, para isso o Oracle conta com o comando SCOPE IS. Quando utilizado no processo de criação de uma tabela, o atributo estará diretamente ligado com a tabela que sucede à declaração desse comando (Baptista, 2009). Existem outros conceitos interessantes de referência como o operador WITH ROWID que serve para garantir a integridade referencial e o apontador de referência -> que obtém um atributo específico de um atributo referência (Elmasri & Navathe, 2004).

### 2.2.4 Herança

O conceito de herança é talvez a ferramenta mais poderosa dos SGBDOR. É através dela que novos tipos podem especializar tipos já existentes (Fonseca, Neto, Souza, & Dourado, 2007). É importante destacar que o Oracle só habilita suporte para herança simples. Dessa forma, tem-se apenas um supertipo e vários subtipos (Boscarioli, Bezerra, Benedicto, & Delmiro, 2006). Onde na definição de um subtipo, observam-se as seguintes características (Baptista, 2009):

- Herdam todos os atributos e métodos do tipo ancestral;
- Podem adicionar novos métodos e atributos; e
- Podem sobrescrever métodos do supertipo;

A fim de definir um subtipo no Oracle, deve-se atentar ao supertipo. Por *default*, quando se cria um tipo abstrato ele é FINAL. Dessa forma não pode ser estendido a um subtipo. O conceito de final também pode ser aplicado aos métodos, mas ao contrário dos tipos, os métodos por default são NOT FINAL (Boscarioli, Bezerra, Benedicto, & Delmiro, 2006). A fim de sobrescrever um método, função ou *procedure* de um supertipo deve-se utilizar a palavra reservada OVERRIDING e verificar se há permissão de acesso (Al). Em seguida utilizando-se da palavra reservada UNDER cria-se um subtipo especializado. A Figura 2.6 ilustra com clareza esse processo.

```
CREATE TYPE tp_Extensivel as OBJECT (
atrSuper INTEGER,
MEMBER PROCEDURE superProc,
FINAL MEMBER PROCEDURE finalProc

) NOT FINAL;


CREATE TYPE tp_Estendido UNDER tp_Extensivel(
atrSub INTEGER,
MEMBER PROCEDURE subProc,
OVERRIDING MEMBER PROCEDURE superProc
);
```

Figura 2. 6 Exemplo de Extenção de Tipos

É importante destacar que o conceito de herança também é aplicável às tabelas. Os tipos das tabelas filhas devem ser subtipos da tabela pai. Todas as tuplas das tabelas filhas estão implícitas na tabela pai. Mas, a fim de melhorar a eficiência do sistema de bancos de dados, muitas vezes são replicados os atributos herdados dentro das tabelas filhas, o que facilita bastante as consultas, já que todos os atributos fariam parte de uma única tabela (Elmasri & Navathe, 2004).

Se não houvesse esse conceito de herança, os administradores de bases de dados deveriam encadear explicitamente as tabelas de subclasses à suas superclasses através de chaves primárias, tal qual definir restrições para garantir restrições referenciais e de cardinalidade (Feitosa, Jesus, Machado, & Alencar, 2006) (Neves, Rocha, & Segundo, 2008).

### 2.2.5 Métodos

Além de *procedures* e *functions* que não estão associados a nenhum objeto

específico da base de dados, os SGBDOR contam com os métodos. Estes são rotinas que podem acessar todos os atributos desse tipo e só podem ser definidos dentro de um tipo abstrato de dados (Chaudri & Zicari, 2001) . São facilmente diferenciáveis de um procedimento ou função graças ao uso de parênteses (mesmo que não ocorram parâmetros). Métodos podem ser (Boscarioli, Bezerra, Benedicto, & Delmiro, 2006):

- MEMBER - que são os mais comuns; implementam as operações das instâncias do tipo e são invocados pela sintaxe objeto.método();
- STATIC - invocados apenas nos tipos de dados e não nas instâncias, logo devem ser utilizados em operações globais ao tipo. São chamados da seguinte maneira: tipo.método();
- Construtor - Sempre é criado implicitamente ao criar um tipo de objeto. Seu nome deve sempre ser igual ao nome do tipo e pode haver mais de um (polimorfismo); e
- MAP ou ORDER - basicamente servem para ordenação. Não é possível utilizar ambos: uma vez que se escolha um, o outro está automaticamente excluído. Enquanto o método Order necessita de um parâmetro e o utiliza para comparar com o objeto atual, retornando um inteiro, o método Map não exige parâmetro e compara vários objetos por um ou mais dos seus atributos (Fonseca, Fidalgo, & Carolina).

Ao final dessa seção podemos especificar uma sintaxe mais complexa para criação de um TAD, representada pela figura 2.7.

```
CREATE [OR REPLACE] TYPE new_type
[ UNDER supertype_name ]
AS OBJECT(
[lista de atributos]

[ [ NOT ] INSTANTIABLE ]

[[ NOT ] FINAL]

[ --REF ref-options
    --1. Definido pelo usuário
        [REF USING tipo]
    --2. Derivada
        [REF (nomeAtributo,...)]
    --3. Gerada pelo Sistema
        [REF IS SYSTEM GENERATED]

 ]
|
[lista de metódos]
);
```

Figura 2. 7 Sintaxe Complexa de Criação de TAD

## 2.4 Mapeamento Objeto-Relacional

A fim de adicionar as características de orientação a objeto nas bases relacionais e de fato garantir o funcionamento de uma base Objeto-Relacional, é necessário uma tradução. Essa tradução é, portanto o mapeamento Objeto-Relacional (Fonseca, Neto, Souza, & Dourado, 2007). Vários autores publicaram diferentes modelos de mapeamento, tanto abordagens específicas quanto abordagens mais genéricas. O administrador da base deve escolher de forma criteriosa o tipo de modelo que mais o convém (Chaudri & Zicari, 2001).

Um modelo de mapeamento baseado em (Vara, Vela, Cavero, & Marcos, 2007) e (Boscarioli, Bezerra, Benedicto, & Delmiro, 2006) é proposto no Quadro 2.2. Onde a coluna Mapeamento OR descreve como deve ser mapeado o tipo de estrutura em questão, caso mais de uma opção seja possível, um número identifica cada uma.

Quadro 2. 2 Modelo de Mapeamento OR

| **Tipo de Estrutura** | **Mapeamento OR** |
|---|---|
| Entidade | Criação de Tipo e Tabela com OID como chave única. |
| Atributo Simples | Atributo do Tipo e coluna em tabela. |
| Atributo Composto | Cria-se um Tipo e o aninha com o Tipo da entidade ao qual pertence. |
| Atributo Multivalorado | Cria-se um *Varray ou Nested Table* que será aninhado com o Tipo ao qual pertence. |
| Herança | Cria-se apenas as tabelas filhas repetindo todos os atributos da tabela pai, dessa forma evita-se junções desnecessárias o que pode contribuir para consultas mais rápidas. |
| Relacionamento 1x1 | Cria-se as duas tabelas onde uma das duas contém uma referência para a outra tabela. |
| Relacionamento 1xN | 1. Na tabela representada pelo tipo com aridade **N** é inclusa uma referência para um objeto do tipo representado pela aridade **1**; **ou**<br>2. Transformar a tabela do tipo de aridade **N** em uma tabela aninhada na tabela de objetos do tipo de aridade **1** |
| Relacionamento MxN | Cria-se uma nova tabela com referências às chaves primárias das duas tabelas originais. |

## 2.5 Conclusão

Nesse capítulo foi realizado um breve estudo sobre sistemas de bases de dados desde os primórdios, quando ainda eram simples sistemas de arquivos. Ted Codd,

funcionário da IBM revolucionou o modo como os dados deveriam ser armazenados (Sanches, 2005). Foram discutidos conceitos importantes sobre os SGBD relacionais. Tanto os aspectos positivos, que ainda o fazem ser o principal tipo de base utilizado, quanto os aspectos negativos que fizeram com que pesquisadores procurassem outras formas de armazenar dados de forma mais complexa que simples tabelas.

Em seguida foram abordados os Sistemas de Bases de Dados Orientados a Objetos que surgiram para lidar com tipos complexos de dados. Mas alguns problemas como fraco desempenho e baixa escalabilidade fizeram com que esses sistemas não se consolidassem.

Na seqüência, são focalizados os Sistemas de Bases de Dados Objeto-Relacionais que mesclam conceitos de orientação a objeto com sistemas relacionais. São destacados vários de seus conceitos adotados pelo Oracle, incluindo as sintaxes de linguagem PL/SQL. Por fim, é proposto um modelo de mapeamento Objeto-Relacional. Modelo este que será utilizado pela ferramenta apresentada no capítulo 4.

No próximo capítulo serão abordadas ferramentas CASE que podem auxiliar o processo de criação de uma Base de Dados Objeto-Relacional.

# Capítulo 3 -Ferramentas CASE para Desenvolvimento de Modelos

Este capítulo discorre sobre ferramentas CASE. Inicialmente, na Seção 3.1 será feita uma definição formal dessas ferramentas. A Seção 3.2 analisa 3(três) ferramentas utilizadas para desenvolvimento de modelos ER estendidos. Esses modelos possibilitam uma criação mais rápida de bases Objeto-Relacionais. Essas ferramentas serão sobrepostas segundo tópicos específicos para que se possa descobrir as maiores limitações comuns. Ao final do capítulo, na Seção 3.3, será feita uma conclusão do que se foi discutido.

## 3.1 Definição

Desde o início da indústria de computadores, a indústria de software procura soluções para problemas de desenvolvimento. Foi da segunda metade da década de 80 ao início dos anos 90 que a pesquisa tornou-se focada em Engenharia de Software Automatizada (Vessey, Janeiro/95). Ferramentas CASE (*Computer-Aided Software/Systems Engineering*) tratam-se de produtos focados em dar suporte a uma ou mais tecnologias em um contexto de desenvolvimento de software, seja esse desenvolvimento seguindo método estruturado ou orientado a objetos. São criadas com o intuito de melhorar a qualidade dos sistemas desenvolvidos, além de aumentar a velocidade de atividades de modelagem e desenvolvimento (Iivari, Outubro/96). Podem ser divididas em dois tipos (Jarzabek & Huang, Agosto/98):

1. *Upper*-CASE que focam em análise do sistema e design lógico de fases; e
2. *Lower*-CASE que focam na construção de sistemas de software

Ferramentas CASE típicas oferecem editores gráficos que ajudam desenvolvedores a modelar aspectos diversos de sistemas de software. Trazem conhecimento de regras de método, o que faz com que as próprias ferramentas sejam capazes de desenhar um diagrama de dados correto ou como gerar esquemas relacionais a partir de diagramas entidade-relacionamento (Jarzabek & Huang, Agosto/98). Tal conhecimento permite às ferramentas case automatizar algumas transações em etapas de desenvolvimento de software.

Nesse trabalho foi criada uma ferramenta CASE que permite ao usuário criar um modelo ER estendido (comum a diferentes fabricantes de Sistemas de Bases de Dados) utilizando-se de editores gráficos simples. A partir do modelo gerado, é possível gerar

um script de criação de tipos e tabelas em sintaxe ORACLE para uma base de dados Objeto-Relacional.

## 3.2 Análise de Concorrentes

Esta seção analisa três ferramentas CASE que podem ser utilizadas para realizar modelagem de bases de dados. Serão destacados os pontos fortes e fracos de cada uma.

### 3.2.1 BR Modelo

BrModelo (brModelo) é uma ferramenta freeware e de código aberto para geração de esquemas conceituais e lógicos de bases relacionais. Toda sua implementação foi baseada na notação de Peter Chen (Chen, 1976) com alterações introduzidas pelo Dr. Carlos Alberto Heuser (Heuser, 2008). Trata-se de uma ferramenta de fácil manipulação, implementada em Bordland Delphi (Bordland Delphi), amplamente utilizada no meio acadêmico (Cândido, 2004).

O autor da ferramenta, (Cândido, 2004) destaca suas vantagens:

1. Permitir alterações estruturais no modelo diante de novas decisões do analista como, por exemplo, promover atributo a entidade;
2. Atenção especial aos atributos, tratando de forma diferente atributos simples, compostos, multivalorados e chave e podendo ocultá-los; e
3. Dicionário de dados para cada objeto do sistema.

Uma análise mais profunda do programa pode trazer à tona as seguintes desvantagens:

1. Interface de manipulação para funcionalidades extras como promoção de atributos pouco intuitiva;
2. Ausência de geração de Scripts; e
3. Pouco ou nenhum suporte para Bases Objeto-Relacionais.

Na Figura 3.1 é apresentada a interface gráfica do brModelo. É importante notar que na Área **1** são disponibilizadas todas as opções necessárias para geração do modelo conceitual proposto. Caso o projetista queira desenhar o modelo lógico, essas ferramentas são simplesmente substituídas por funcionalidades do outro modelo. Na Área **2**, o desenho do projeto pode ser visualizado em uma tela de tamanho extensível, garantindo a escalabilidade da aplicação. A Área **3** organiza o menu da aplicação com alguns atalhos redundantes. Na Área **4**, os atributos de cada objeto podem ser visualizados.

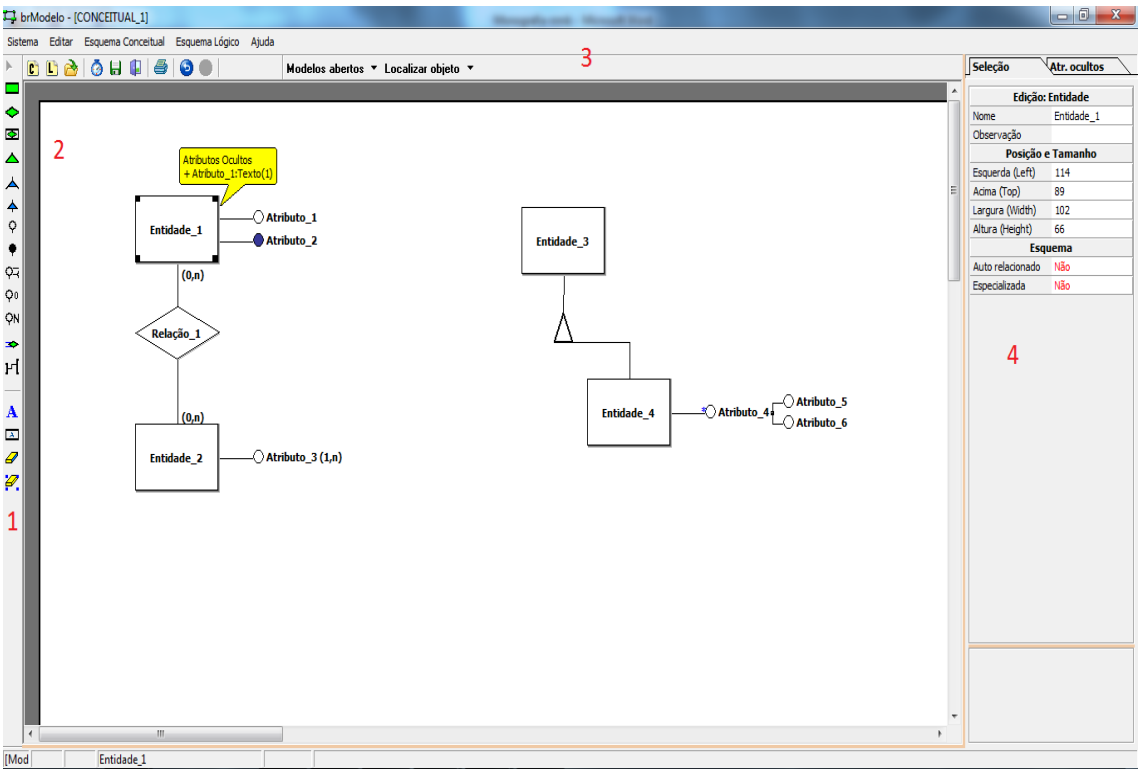


Figura 3. 1 Interface Gráfica da Ferramenta brModelo

### 3.2.2 Microsoft Visio

Microsoft Visio 2007 (Visio) é uma ferramenta CASE que permite ao usuário modelar vários tipos de diagrama. O estilo de desenho é escolhido pelo tipo de diagrama carregado. O Visio ainda conta com algum suporte para bases de objeto e funcionalidades aplicadas como herança de tabela e herança de tipo.

Porém, o Visio conta com uma licença de uso relativamente cara e não é tão focado em bases de dados. Oferece um tipo de diagrama para bases de dados com um excesso de funcionalidades, que podem de certa forma confundir o usuário iniciante (Figura 3.2). Também, vale destacar que apenas dá suporte a diagramas lógicos. Caso o usuário queira criar uma notação equivalente ao modelo de Chen (Chen, 1976) e escolher um diagrama qualquer, deve perder tempo desenhando novos tipos de imagem como atributo multivalorados e perde qualquer regra de negócio possível. Permite fazer um desenho de diagrama livre e bastante propenso a falhas por administradores inexperientes (Figura 3.3). Não conta com ligação nenhuma a bases de dados nem gera scripts.

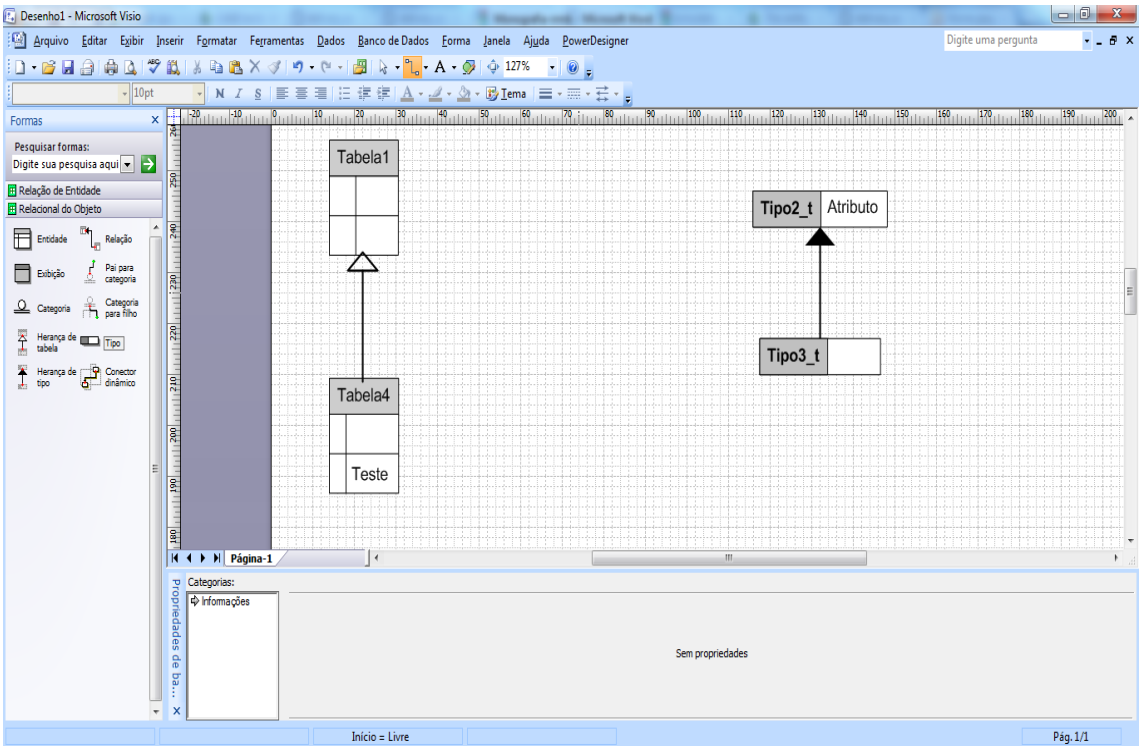


Figura 3. 2 Esquema Lógico para BD no Visio

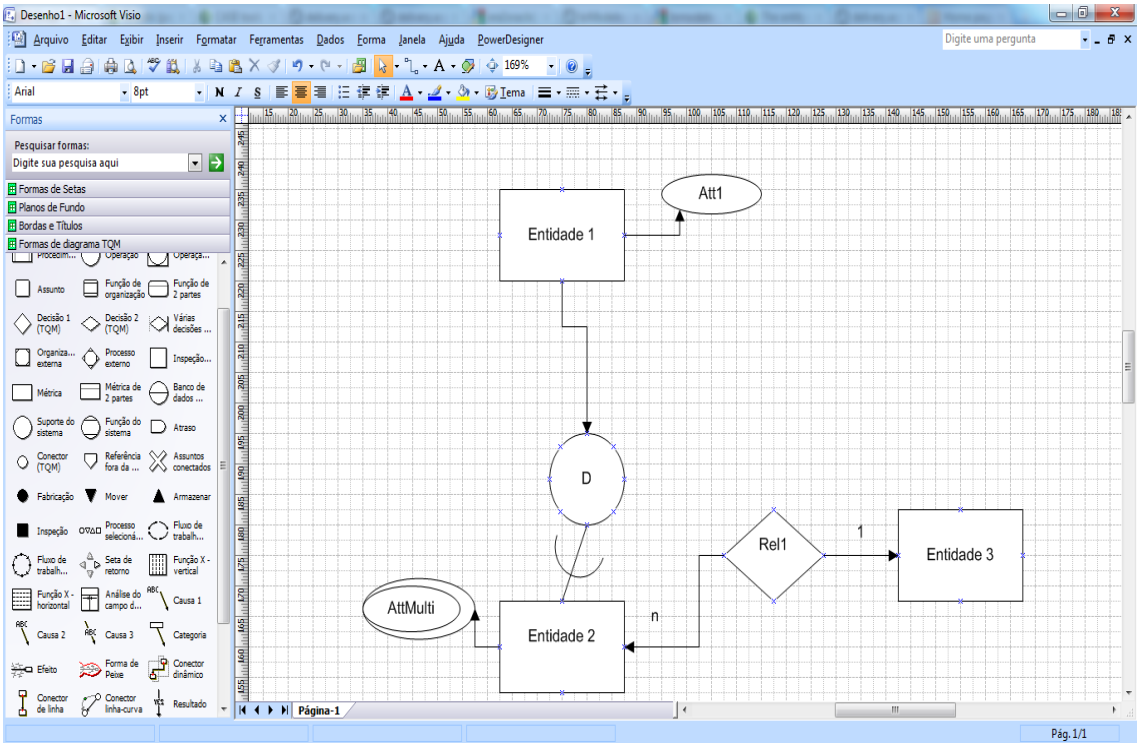


Figura 3. 3 Implementando um modelo conceitual como o de Chen no Visio

### 3.2.3 Sybase Power Designer

O Sybase Power Designer (Power Designer) mescla o melhor das duas ferramentas apresentadas anteriormente. Tem interface mais intuitiva que a do Microsoft Visio (Visio) e regras de negócios bem amarradas. Segundo Power Designer (Power Designer) essa ferramenta é capaz de dar suporte a mais de oitenta sistemas de BD Relacionais de diferentes fabricantes.

Tem uma funcionalidade única de verificação de modelo que não só verifica a corretude do que se foi especificado pelo projetista, como aponta todos os erros que foram gerados. Isso é mais bem exemplificado na Figura 3.4. Onde na Área 1 (um) pode-se observar os erros e na Área 2 (dois) um relatório apurado de todo o processo de validação do modelo.

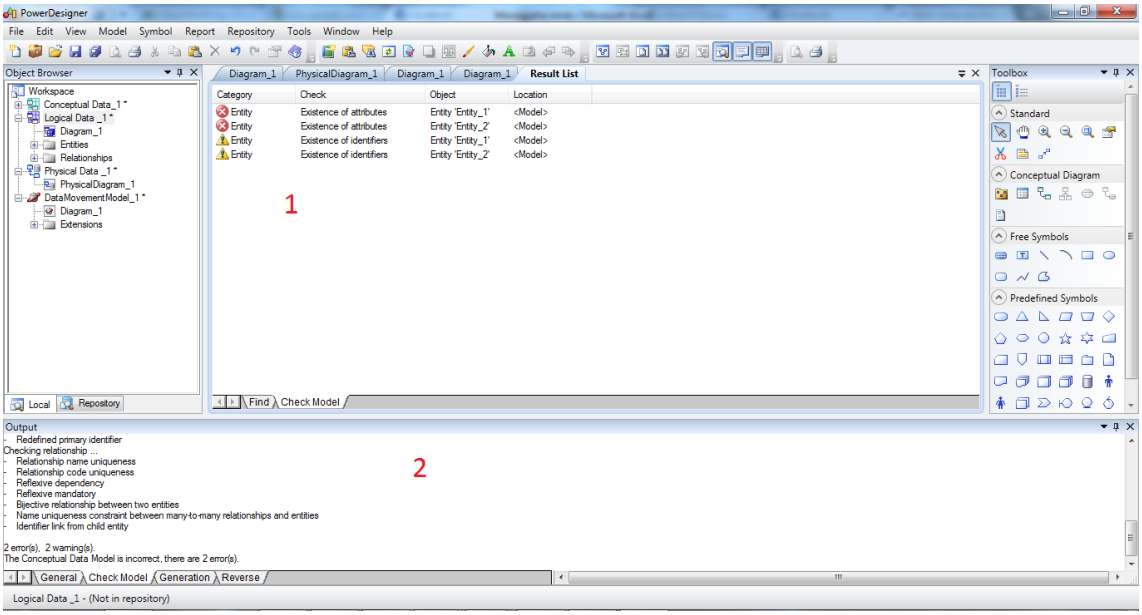


Figura 3. 4 Funcionalidade de Validação de Modelo do Power Designer

Porém a ferramenta, como as outras duas oferece pouco ou nenhum suporte ao modelo entendido. Não conta com automatização de geração de script e tem uma licença mais cara do que o próprio Visio.

### 3.2.4 Comparações entre as Ferramentas Analisadas

Com base no que foi apresentado pelas três ferramentas será feita uma análise conforme os seguintes critérios:

- *Facilidade de Manipulação* - onde um valor 0 representa uma ferramenta de entendimento complexo e 5 uma ferramenta simples e intuitiva;

- *Suporte a aplicações de Bases de Dados* - onde um valor 0 provê nenhum suporte e um valor 5 provê suporte bastante satisfatório;
- *Suporte a modelos estendidos* - onde um valor 0 representa uma ferramenta que não sabe como lidar com modelos estendidos e 5 uma ferramenta que conta com total domínio dos mesmos;
- *Geração de Scripts*- Um valor 0 significa uma ferramenta que não gere scripts enquanto um valor 5 implica em uma ferramenta que contém essa funcionalidade no nível satisfatório;
- *Preço de Licença* - Valor 0 significa licença de custo alto e um valor 5 representa isenção de custos;
- *Funcionalidades Extras* - Onde um valor 0 significa uma ferramenta CASE que trabalha apenas com bases de dados e um valor 5 provê uma variedade maior de serviços para desenvolvimento de outros tipos de diagrama.

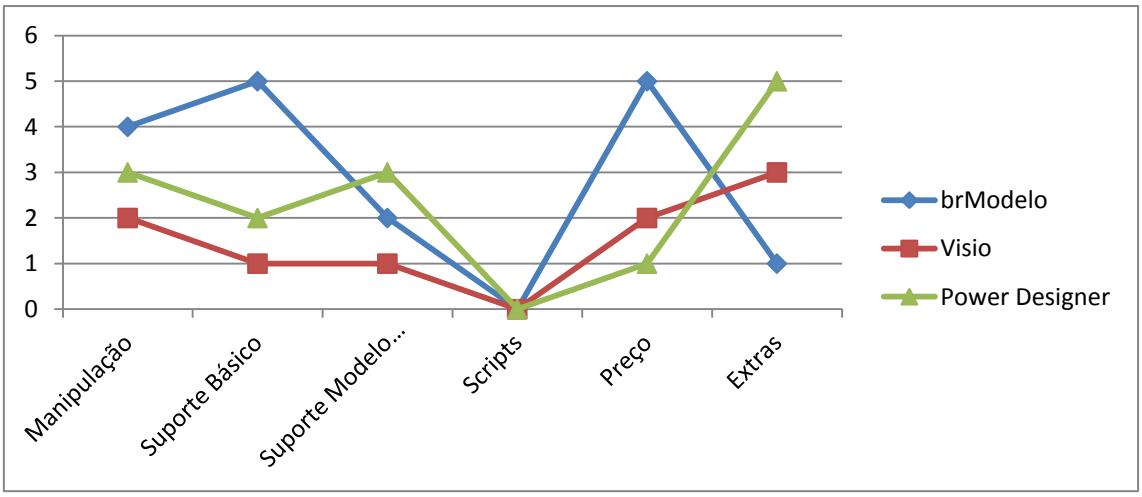


Figura 3. 5 Gráfico de Análise dos Competidores

De acordo com a Figura 3.5, nota-se que as ferramentas analisadas pecam no quesito gerador de Script e deixam bastante a desejar no suporte a bases de dados, sobretudo considerando um modelo ER estendido.

## 3.3 Conclusões

Neste capítulo foram abordadas ferramentas CASE para desenvolvimento de modelso de dados. Na primeira seção foram destacados pontos importantes como seus objetivos. Também foi citada uma pequena parte de sua história, tal qual uma pequena

subdivisão interna, onde o parâmetro principal era o tipo de aplicações que a ferramenta cobre.

Em seguida foram descritas ferramentas que são utilizadas por Administradores de Bases de Dados destacando vantagens, desvantagens e características únicas. Ao final, foi feito um estudo comparativo simultâneo de todas as ferramentas discorridas nesse capítulo. Essa sessão foi responsável por apresentar pontos importantes que são bastante carentes nessas bases.

O próximo capítulo apresentará uma nova ferramenta CASE que objetiva preencher as lacunas analisadas por esta seção. Destacará como cada limitação foi superada, além de abordar alguns processos de engenharia de software aos quais foi submetida.

# Capítulo 4 - Criação da Ferramenta VORM

A ferramenta *Varial Object Relational Model* (VORM) proposta nesse trabalho surge como alternativa em automatizar e facilitar a criação de uma base de dados Objeto-Relacional para Oracle (Oracle). A ferramenta conta com duas partes que trabalham interligadas:

- Ferramenta de diagrama para modelo ER estendido; e
- Gerador de Script que traduz o diagrama em um esquema para uma base Objeto-Relacional, a partir do mapeamento proposto pelo Quadro 2.2

Na Seção 4.1 serão descritos os conceitos implementados pela ferramenta, tanto no âmbito de diagramas, quanto na geração de scripts. A Seção 4.2 descreve a especificação da ferramenta. A Seção 4.3 confirma o que foi proposto pela Seção 4.2 mostrando a interface gráfica da versão *BETA* da ferramenta. Por fim, a Seção 4.4 conclui o que foi exposto por esse capítulo.

## 4.1 Conceitos

Os conceitos implementados pela ferramenta são descritos nesta seção. Em primeira instância serão apresentados os conceitos utilizados pela parte de diagramas ER estendidos. Em seguida serão especificados os da parte de geração de script.

### 4.1.1 Diagrama ER Estendido

Para fins acadêmicos, os conceitos de modelagem de um SGBDOR propostos pela ferramenta (Power Designer) requer que o usuário já tenha familiaridade com esse tipo de base de dados. Por outro lado, grande parte do sucesso da ferramenta brModelo no meio acadêmico se deu pelo modelo intuitivo proposto ao administrador iniciante (Cândido, 2004). Porém a ausência de um suporte ao modelo estendido torna essa base imprópria para criação de bases Objeto-Relacionais.

Baseado no esquema de aula proposto por (Fonseca, Fidalgo, & Carolina) na disciplina introdutória Gerenciamento de Dados e Informações no Centro de Informática da UFPE foi adaptado um modelo ER estendido simplificado para diagramação rápida e eficiente de bancos que requerem esse tipo de suporte. Esse modelo adaptado utiliza-se dos seguintes conceitos abordados pela Figura 4.1.

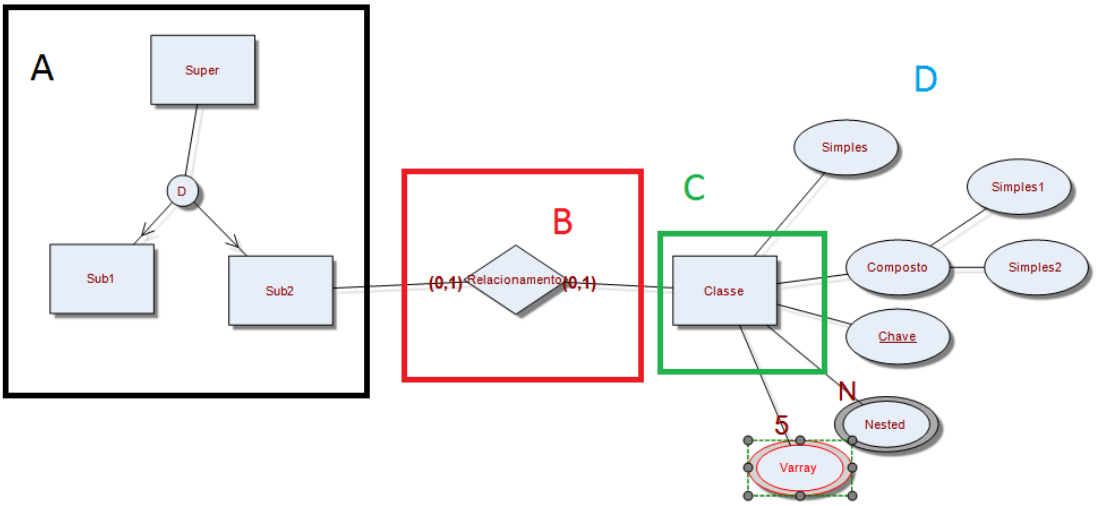


Figura 4. 1 Conceitos do modelo ER Estendido

Pelo modelo apresentado, pode-se analisar as seguintes regiões:

- A região indicada por **(C)** representa o conceito de Classes. Cada entidade representa um tipo que será definido e é ilustrada por um retângulo;
- A região **(A)** representa todo o conceito de herança. Super Classes podem ser estendidas em subclasses. Um círculo com um **D** representa uma especialização de classes disjunta, ligando as subclasses às disjunções uma linha com seta aberta;
- A região **(D)** indica o conceito de atributo. Vários atributos definem uma classe propriamente. Podem ser:
    1. Simples - Quando são simplesmente definidos por um nome que os descreve e um tipo primitivo adotado do SGBD em que foi implementado. São ilustrados por uma simples elipse;
    2. Chave - Quando identificam unicamente a classe em questão. Ilustrados pela elipse de atributos acrescida do nome que o representa sublinhado;
    3. Composto - Quando outros atributos simples o definem; Representado graficamente por atributos simples ligados a outro atributo simples;
    4. Multivalorado *Varray* -Quando um atributo pode ter vários valores possíveis. O administrador deve especificar previamente o tamanho máximo de um *VARRAY* . É ilustrado por uma elipse envolvendo uma menor e o tamanho máximo ilustrado na reta que o liga à entidade; e
    5. Multivalorado *Nested* - Equivalente ao Multivalorado Varray, porém não tem um limite de elementos. Ilustrado por dupla elipse com a borda mais escura e um N na reta que o liga até a entidade;

**Obs**: É importante notar que atributos multivalorados podem também ser atributos compostos; e

Finalmente a região **(B)** representa o conceito de relacionamentos, representado por um losango com a aridade de cada classe representada por uma tupla de formato (n,m) sobre a reta que liga cada entidade ao relacionamento.

### 4.1.2 Geração de Scripts

Grande parte do conceito para geração de scripts para bases Oracle foi descrito nas sintaxes apresentadas pelo Capítulo 2. Deve-se ter em mente a sintaxe de criação de *Types*, *Tables*, *REFS*, listas de Atributos e criação de *VARRAYS* e *NESTED TABLES*. O mapeamento apresentado ao final do Capítulo 2 demonstra com clareza como cada elemento gerado pela diagramação ER Estendida se comportará na base Objeto-Relacional.

## 4.2 Especifiação da Ferramenta

Com o auxílio da ferramenta Astah (Astah), foi possível criar os seguintes casos de usos identificados:

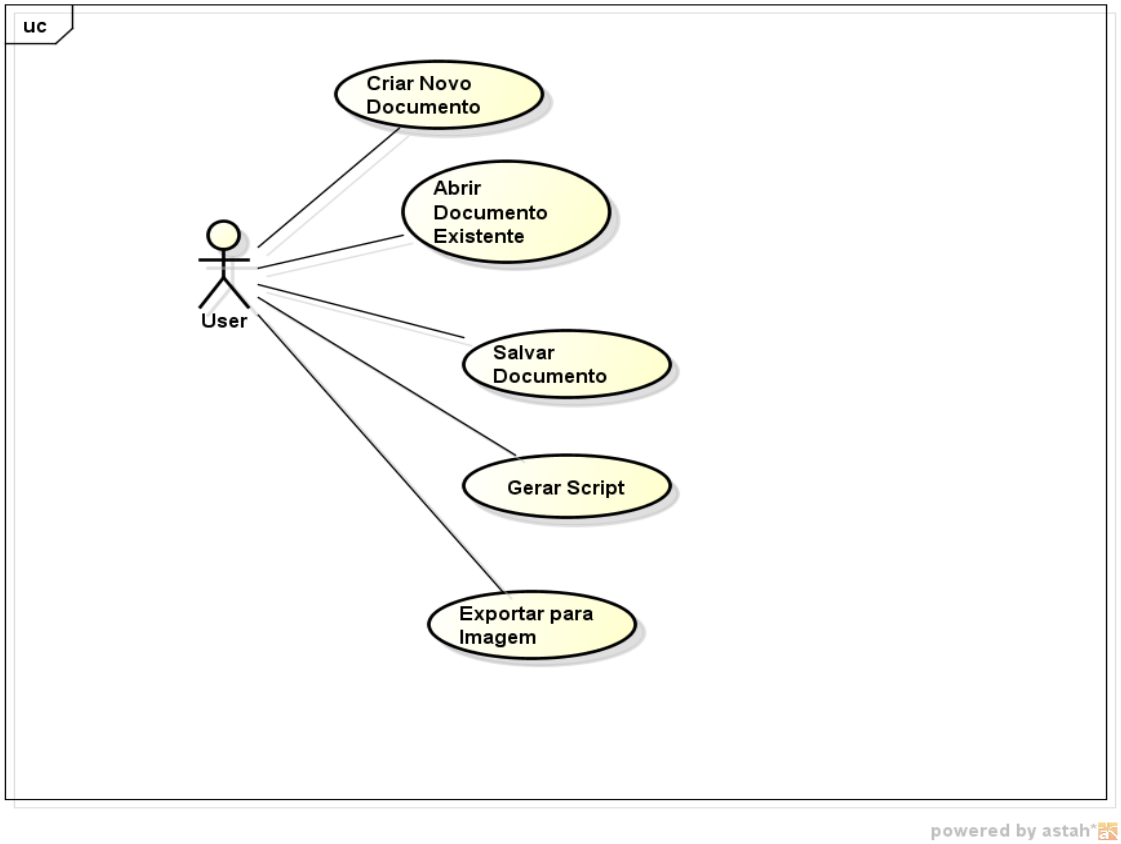


Figura 4. 2 Casos de Uso Manipulações Gerais

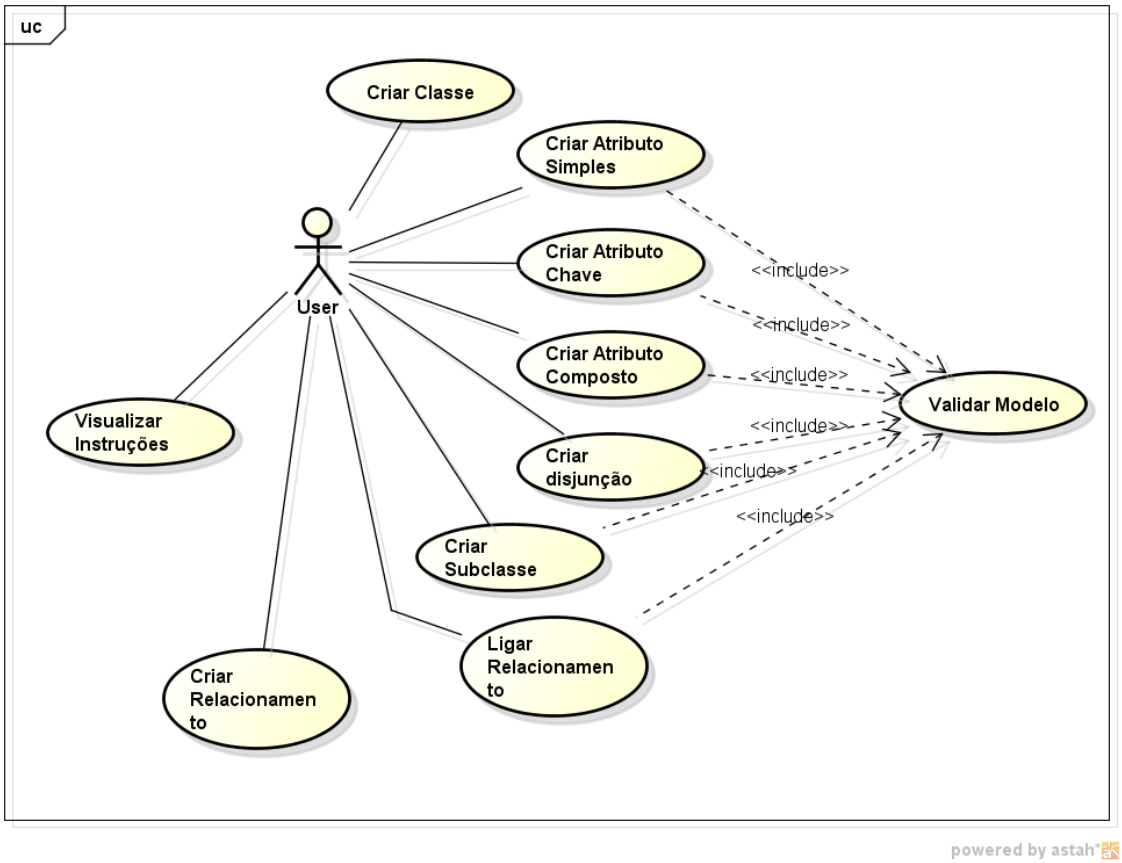


Figura 4. 3 Casos de Uso Criação de Modelos

A Figura 4.2 ilustra os processos básicos da ferramenta. Ela deve ser capaz de criar um novo documento, e garantir persistência dos dados já que a grande maioria das bases de dados sempre são alteradas com o passar do tempo. Logo, é necessário implementar uma funcionalidade para que se possa salvar um projeto lógico e posteriormente abri-lo. Uma vez que o modelo ER Estendido esteja completo, é de interesse do usuário também gerar um script da base e exportar o diagrama para um arquivo do tipo imagem.

O gerador de Script é a principal funcionalidade dessa ferramenta. Um modelo lógico será construído simultaneamente com o modelo visual. Esse modelo deverá ser lido pela aplicação e validado. Algumas decisões devem ser tomadas pelo usuário quando mais de uma opção de mapeamento objeto-relacional for possível.

A exportação para formato imagem ajudará ao usuário na criação de documentos que descrevam a base.

A Figura 4.3 ilustra os casos de uso necessários para a criação do diagrama ER Estendido. Uma vez que o minimundo esteja definido, o usuário pode iniciar a construção do diagrama. Contando com uma interface simples e intuitiva baseada nas

interfaces dos concorrentes. A funcionalidade de Visualizar Instruções é apresentada para familiarizar o usuário com a ferramenta. A cada ação tomada pelo usuário um output será apresentado. Essa mensagem de saída pode ser uma instrução para um passo seguinte ou uma mensagem de alerta para uma ação ilegal executada. A maioria das ações executadas pelo usuário para a construção do modelo deve ser validada por um componente. Esse componente será responsável, portanto, por toda a regra de negócio voltada para a parte de diagramação ER Estendido.

O usuário deve ser capaz de criar classes sem nenhuma restrição. A única propriedade editável nas classes é o próprio nome. Caso duas classes com nomes iguais sejam criadas isso será tratado no tradutor do diagrama em scripts. Por exemplo, duas classes no diagrama chamadas *Teste* serão salvas na base de dados como *Type Teste* e *Type Teste2*. Isso é possível já que o programa identifica cada classe não pelo nome definido pelo próprio usuário, mas por um identificador único interno.

A criação de atributos por sua vez deve passar por uma validação. Cada atributo deve estar associado a uma Classe ou a um atributo Composto. Atributos Chaves não podem tornar-se atributos Compostos, ao contrário de atributos Multivalorados tanto do tipo *Varray*, quanto do tipo *Nested*. Não é possível aninhar atributos compostos em outros atributos compostos. Para isso o usuário deve criar um relacionamento para tornar o modelo mais legível e correto. O usuário deverá ser capaz de editar as propriedades de cada atributo. Atributos Simples podem assumir os tipos primitivos da Oracle (Oracle) apresentados pelo Quadro 4.1.

Quadro 4. 1 Tipos Primitivos Suportados

| |
|---|
| INTEGER |
| VARCHAR2(N) |
| NVARCHAR2(N) |
| CHAR(N) |
| NCHAR(N) |
| DOUBLE |
| BOOLEAN |
| DATE |

A criação de disjunções para herança deve estar associada obrigatoriamente a uma Classe. Somente elementos do tipo *SubClasse* podem estar ligados a essas

disjunções. Caso a *Superclasse* não estenda mais de uma variação, a *Subclasse* pode ser diretamente ligada à *Superclasse* que nesse caso será abstrata. A *Superclasse* abstrata não pode fazer parte de nenhum relacionamento. Não será possível inserir disjunções ou *Subclasses* em outras *Subclasses*.

A única validação feita no processo de criação de relacionamentos é executada antes de gerar o script da base de dados; antes da geração de imagem; ou antes de um comando de salvamento (onde todas disparam a chamada de uma instância da validação do modelo lógico). Todo relacionamento deve estar ligado a exatamente duas entidades. Caso um relacionamento não cumpra essa requisição ele será descartado. É importante destacar que a ferramenta não dá suporte a auto-relacionamento em seu primeiro release. As entidades devem estar ligadas aos relacionamentos utilizando-se de uma estrutura de ligação. A aridade específica de cada classe poderá ser editada nas propriedades do relacionamento.

### 4.2.1 Tecnologias Utilizadas

Desenvolvida na linguagem de programação C# (C#) em ambiente de desenvolvimento Visual Studio 2010 (Visual Studio 2010). Utilizando a estrutura *WindowsForms* disponibilizada pelo framework .net 4.0 (.Net 4.0) ,o qual disponibiliza bibliotecas internas bastantes úteis, tanto na parte de criação de interface gráfica quanto na parte de manipulação de objetos e códigos. Além das vantagens e de componentes próprios de fácil utilização, esse framework dá suporte a milhares de bibliotecas externas. Uma dessas bibliotecas é o Nevron (Nevron) que foi utilizado para a criação do diagrama ER Estendido.

### 4.2.2 Arquitetura da Aplicação

A implementação da ferramenta segue os requisitos propostos e foi pensada para permitir fácil extensão. Sua arquitetura de forma simplificada é demonstrada pelo diagrama de classes representado na Figura 4.4.

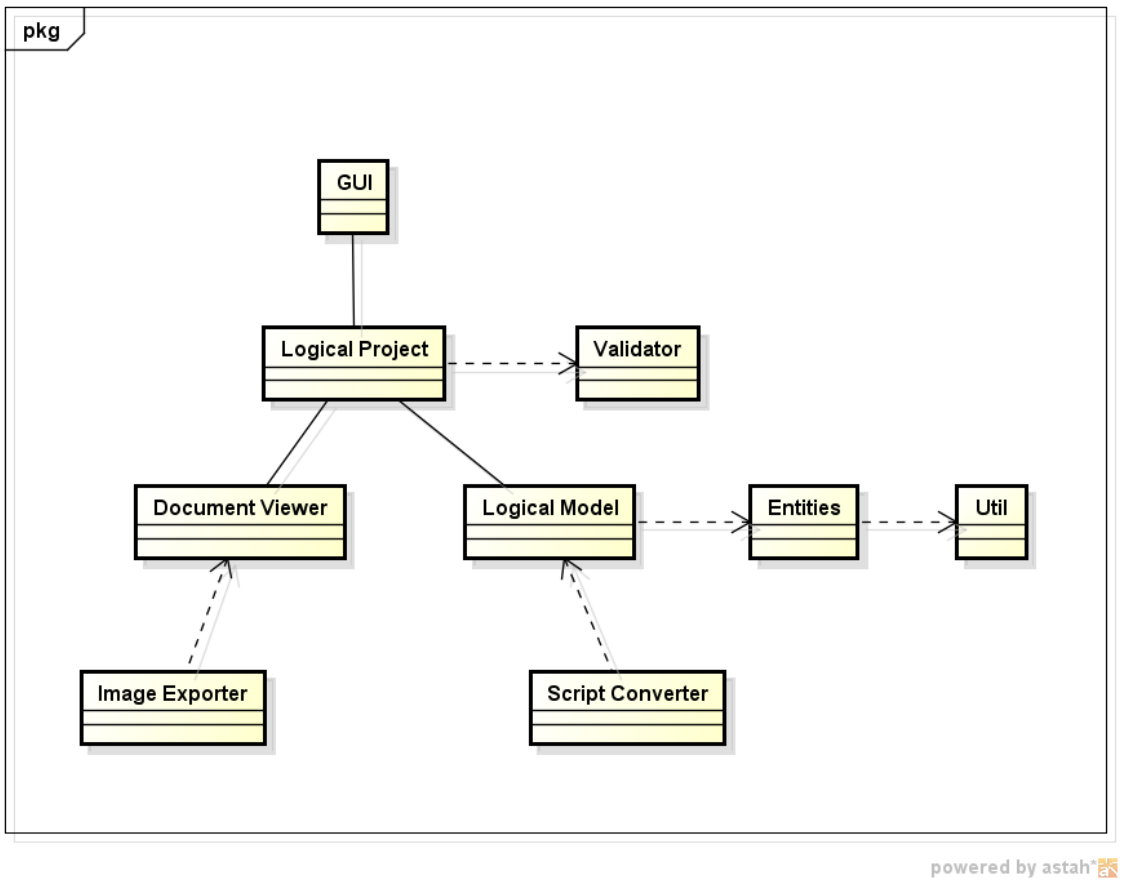


Figura 4. 4 Arquitetura da Ferramenta

O componente *Util* é responsável por promover funções auxiliares extras como a criação de *Enumerators*.

O componente *Entities* armazena todas as classes básicas necessárias o funcionamento do modelo lógico. Também é responsável pelas estruturas de armazenamento das mesmas.

*Logical Model* é a estrutura responsável por prover uma visão lógica do modelo que foi desenhado. Agrupam todas as entidades lógicas do projeto, sejam elas: classes; atributos; relacionamento; e herança, em uma estrutura com regras bem definidas. A partir dessas informações, o *Script Converter* é capaz de transformar essas informações em linguagem de declaração PL/SQL. Baseado em um novo conjunto de regras que definem a ordem em que cada elemento deve ser declarado a fim de evitar falhas na geração do script.

O componente visual é isolado do modelo lógico no módulo *Document Viewer* é utilizado somente para diagramação do modelo ER estendido. A partir de suas informações, o módulo *Image Exporter* é capaz de criar um arquivo do tipo imagem contendo a mesma informação visual do modelo.

Uma instância do módulo *Logical Project* é criada para cada novo projeto de banco de dados. Ela reúne os módulos visuais e lógicos em um único objeto que pode ser serializado e armazenado em um arquivo para garantir a persistência do projeto. O módulo *Validator* é responsável por impor as regras de negócio para o armazenamento dos objetos. Implementado em uma estrutura de regras bem definidas, esse módulo conta com fácil extensão para criação de uma ferramenta mais específica.

Finalmente a *GUI* que engloba todos os componentes gráficos do sistema: desde a tela principal aos pequenos *popups* de decisão para mapeamento.

## 4.3 Interface Gráfica

A Interface Gráfica da versão beta da ferramenta VORM cuja implementação foi descrita nesse capítulo é ilustrada na Figura 4.5.

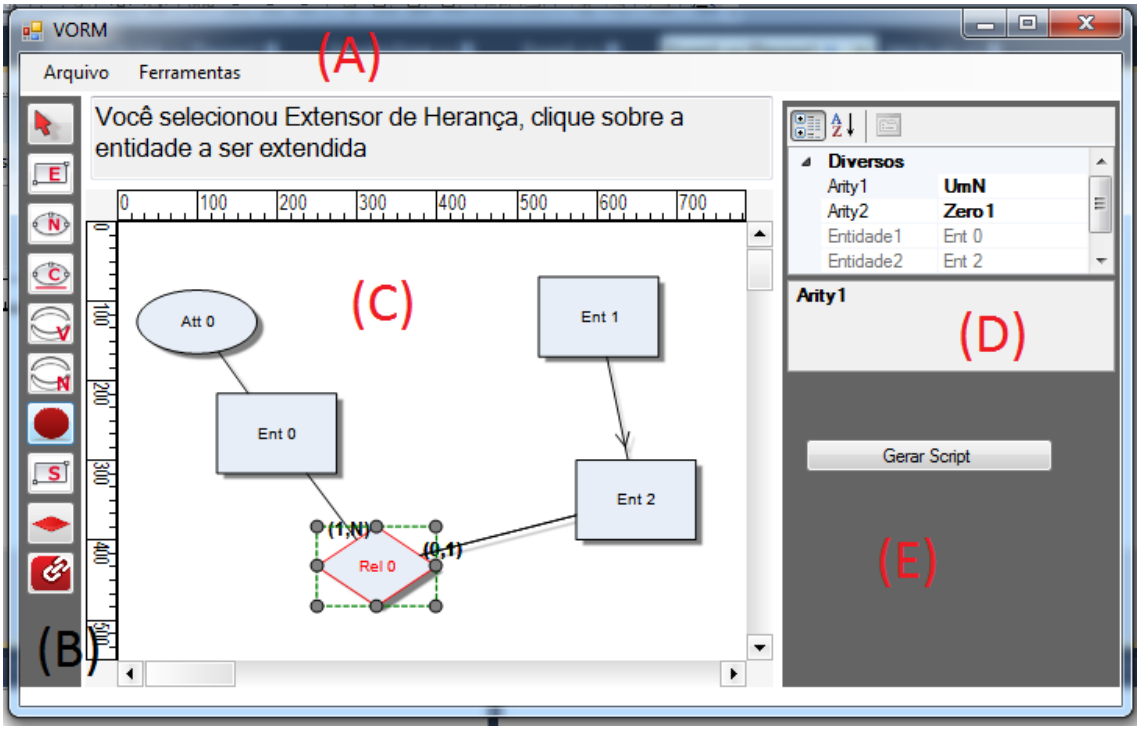


Figura 4. 5 Interface Gráfica VORM

É interessante notar que o design gráfico da VORM tem muitas similaridades com as ferramentas apresentadas no capítulo de concorrentes. Aproveitando o melhor de cada ferramenta chegou-se a essa versão BETA.

A região (A) ilustra o menu onde o usuário pode manipular o arquivo (Operações de criar, salvar e carregar) e requisitar a exportação do mesmo para imagem ou para Script de Banco de dados.

A Região (B) cria a barra de ferramentas para diagramação do modelo ER Estendido. Ao todo são 10 botões dispostos da seguinte maneira:

- Cursor - Corresponde ao elemento neutro. Sua principal função é selecionar um objeto na tela;
- Entidade - Criação de uma Classe;
- Atributo Simples - Ligação de um atributo a uma Classe existente;
- Atributo Chave - Ligação de um atributo chave a uma Classe existente;
- Atributo Multivalorado *Varray* - Liga um Varray a uma classe existente;
- Atributo Multivalorado *Nested* - Liga uma *Nested Table* a uma classe ou atributo composto existente;
- Disjunção - Liga uma disjunção a uma classe previamente definida no modelo;
- SubClasse - Classe filha que pode ser ligada a uma disjunção ou a uma SuperClasse;
- Relacionamento - Criação de um relacionamento na tela; e
- Ligação a Relacionamento - Responsável por criar a ligação lógica e visual de uma Classe a um elemento Relacionamento.

A Região (C) é divida em duas partes. A parte superior equivale a um console que torna a interface mais amigável ao usuário. Dando instruções específicas do que se fazer após cada ação efetuada e alertando sobre erros gerados pelo administrador. A região inferior corresponde ao espaço físico onde é desenhado o diagrama.

A região (D) aponta as características de cada tipo de objeto. Uma vez selecionada uma entidade, atributo de qualquer tipo ou relacionamento, pode ser possível alterar em tempo real algumas de suas características. Trata-se de um número bem limitado para manter o programa em grau de simplicidade alto.

Por fim a região (E) que conta com um único botão para gerar um script. Como explicitado anteriormente neste capítulo, um algoritmo utiliza-se das informações do modelo lógico e monta uma tradução seguindo uma ordem em que os elementos devem ser declarados para que todo o script possa ser aceito pela base de dados. Quando for necessário uma decisão tomada pelo usuário sobre o tipo de mapeamento o *popUp* da

Figura 4.6 é apresentado.

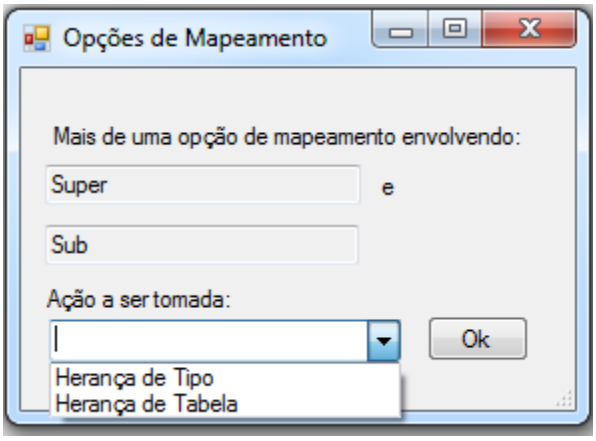


Figura 4. 6 PopUp de decisão de Mapeamento

## 4.4 Conclusões

Esse capítulo abordou o processo de criação da ferramenta VORM. Os conceitos de bases objeto-relacionais empregados pela ferramenta foram detalhados e contextualizados. Em seguida, a concepção da ferramenta foi descrita com a ajuda de diagramas de Casos de Uso e uma especificação informal dos requisitos necessários.

O processo de implementação foi então descrito, destacando as ferramentas utilizadas para criação do programa e uma arquitetura simplificada. A interface gráfica é, em seguida, exibida para confirmar o que foi proposto no início do capítulo.

O propósito maior da ferramenta é aliar o melhor das ferramentas analisadas e adicionar funcionalidades de extrema importância. Logo seu maior diferencial é a possibilidade da criação de scripts.

O próximo capítulo ilustra um caso de uso onde a ferramenta pode ser aplicada. A partir de um minimundo será possível verificar o modelo ER estendido gerado e o script de banco de dados.

# Capítulo 5 - Uma análise de Caso de Uso

Esse capítulo demonstrará um exemplo de funcionamento da ferramenta VORM. Está distribuído de tal forma que a Seção 5.1 descreva um minimundo sobre um subdomínio de um centro de pesquisas em biologia. Será então apresentada uma saída que corresponde à imagem gerada do diagrama ER Estendido e o script simplificado da base. Por fim, na Seção 5.2, as conclusões do capítulo serão apresentadas.

## 5.1 Descrição do problema

O minimundo proposto trata-se de um subdomínio simplificado de uma base complexa. Apesar de limitado, ilustra as funcionalidades propostas pela ferramenta. Segue uma descrição informal do modelo.

Determinada *doença* deve ser armazenada na base com seu nome e até 5 (cinco) dos seus sintomas mais característicos. Cada *doença* pode ser especializada de acordo com sua patologia, mas para análise deste trabalho serão consideradas as especializações *Bacteriana* e *Genética*.

As *bacterianas* devem conter um elemento que identifique a bactéria causadora que compõe um nome científico, seu formato e o grau de agregação. Um *antibiótico*, identificado por um nome e um número ilimitado de compostos químicos identificados pelo nome e dosagem em *mg*. Cada *antibiótico* pode ser associado a mais de uma *doença bacteriana* e uma *doença bacteriana* pode ser tratada por mais de um *antibiótico*.

Por outro lado as *doenças genéticas* precisam de uma informação de origem, contendo o país onde foi manifestada pela primeira vez e a data da primeira manifestação. Algumas *pessoas* afetadas por uma dessas *doenças genéticas* fazem parte de um programa de pesquisa sobre as mesmas. São identificados pelo nome e telefone de contato.

Uma breve análise dessa base simplificada identifica os seguintes aspectos:

- Doença como superclasse. Contendo como atributos Nome (alfanumérico) e um Varray de tamanho 5 contendo um alfanumérico que descreve sintomas;
- Doença Bacteriana como subclasse. Com um atributo composto bactéria;
- Antibiótico como classe. Contendo uma tabela aninhada de compostos químicos e suas quantidades.

- Outra classe identificada é Paciente com outro varray para armazenar telefones e um alfanumérico identificando o nome.
- Há um relacionamento MxN entre Doenças Bacterianas e Antibiótico;
- Há um relacionamento de 1 doença genética para N Pacientes.

Com base nesses dados, utilizando-se da ferramenta VORM seguindo os seguintes passos:

1. Criação de Classes, Atributos, Relações e alterando suas propriedades como demonstrado na Figura 5.1;

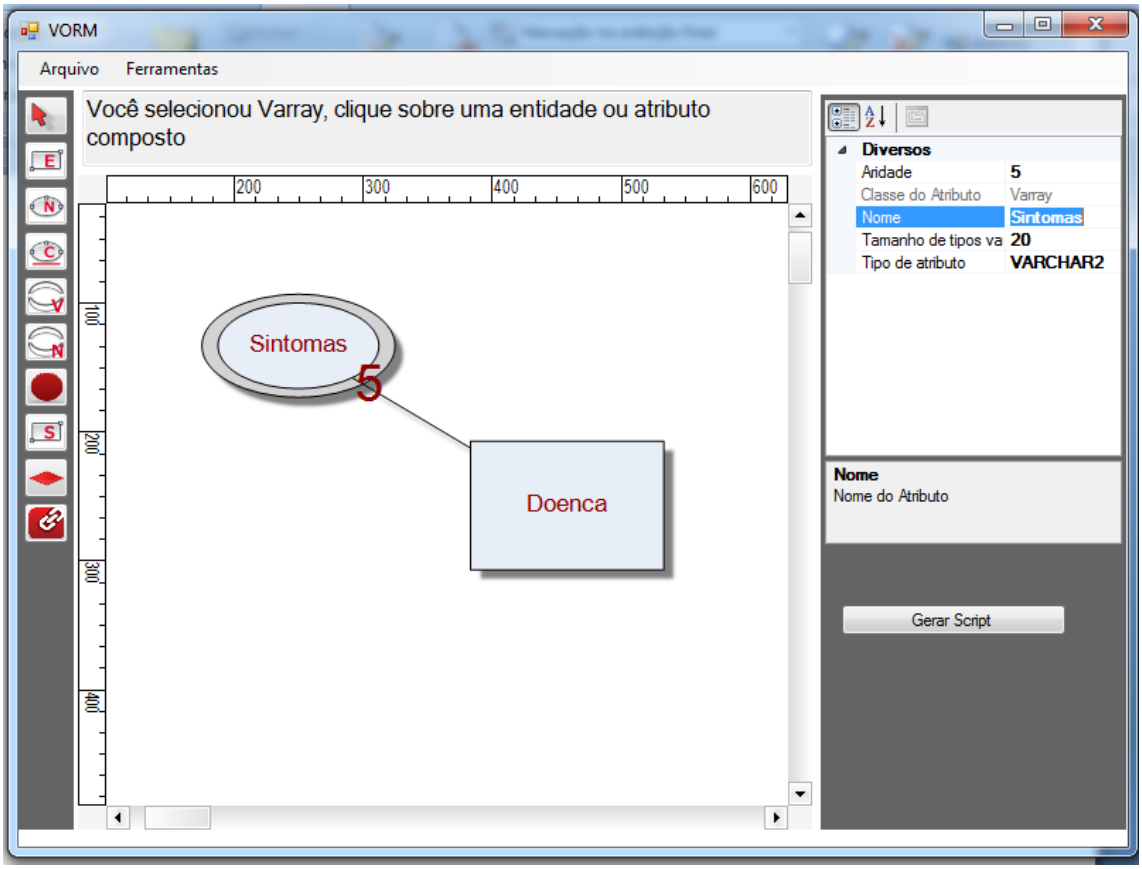


Figura 5. 1 Criação de Entidades e Modificação de Atributos

2. Uma vez que todo o diagrama esteja concluído pode-se exportar a imagem conforme ilustrado pela imagem 5.2 e conferir a imagem exportada na Figura 5.3;

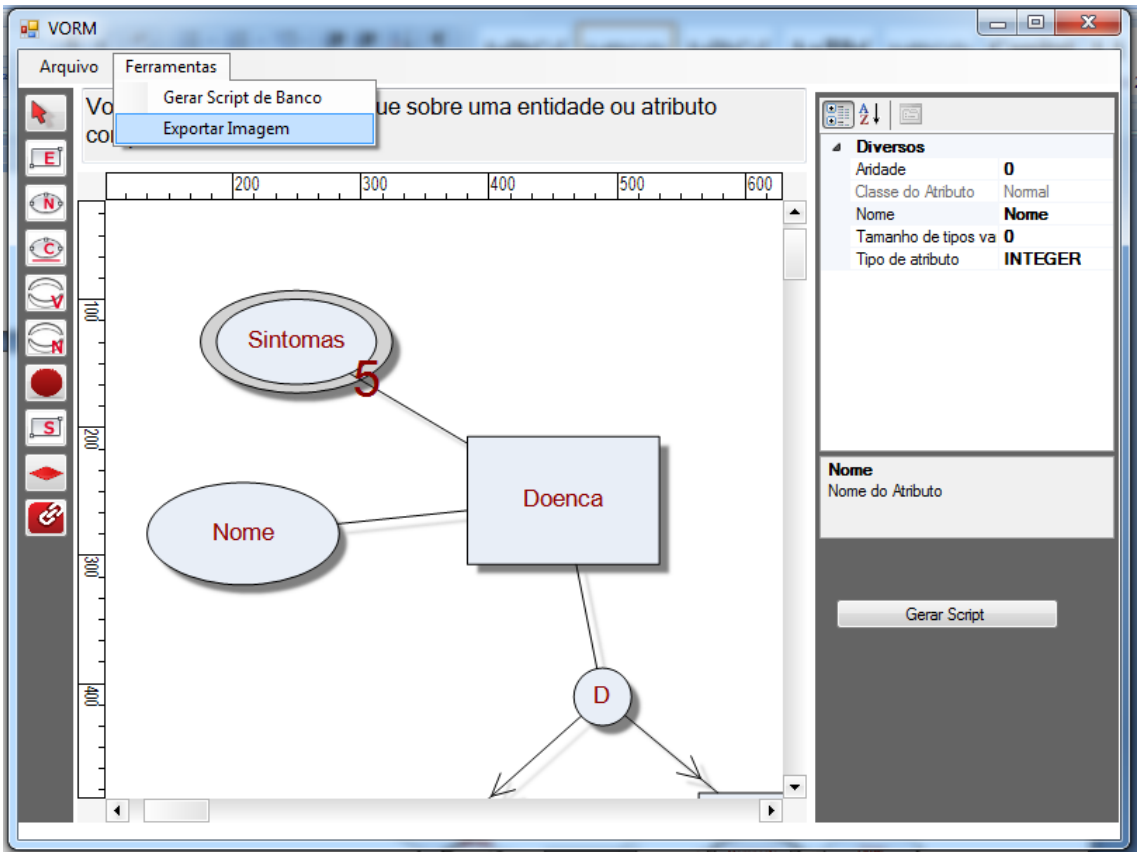


Figura 5. 2 Solicitação de Exportar para Imagem

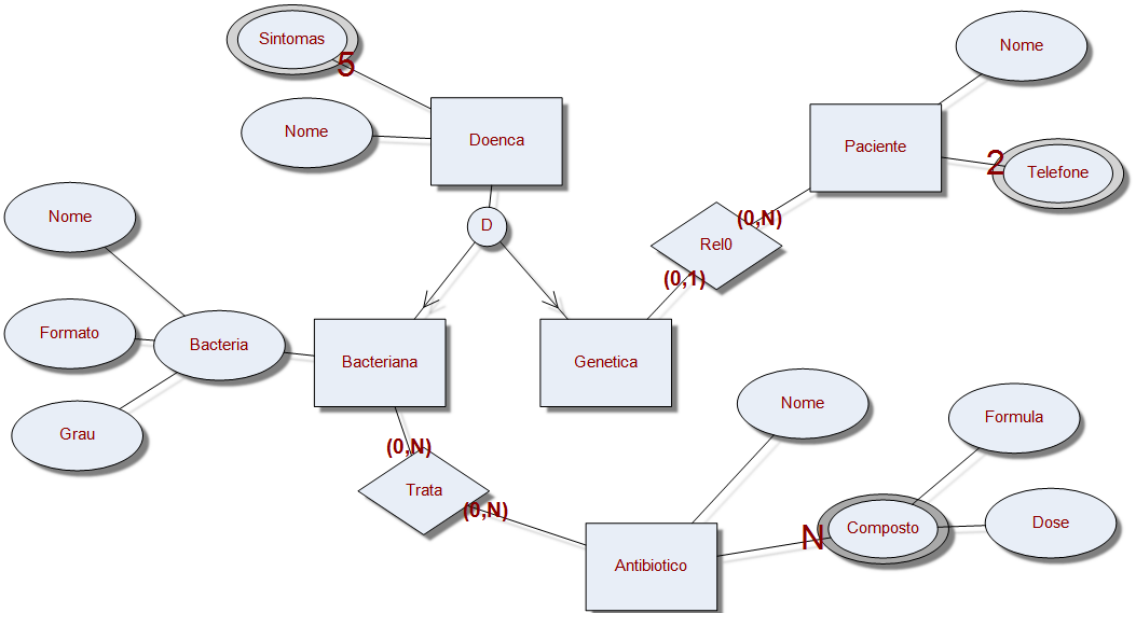


Figura 5. 3 Imagem Gerada pelo Programa

3. Ao final o usuário pode requisitar a geração de script da base de dados, a decisão a ser tomada é apresentada ao usuário na Figura 5.4

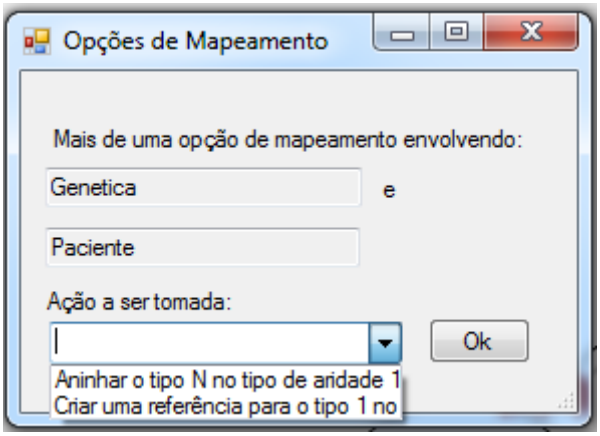


Figura 5. 4 Opções de Mapeamento Genética x Paciente

4. Por fim foi gerado script para os dois tipos de mapeamento. Foram reagrupados para demonstrar as diferenças entre ambos:

Quadro 5. 1 Script Aninhamento

```
CREATE TYPE tp_Sintomas AS OBJECT (
att_Sintomas      VARCHAR2(30));

CREATE TYPE tp_va_Sintomas AS VARRAY(5) of tp_Sintomas ;

CREATE TYPE tp_Telefone AS OBJECT (
att_Telefone      VARCHAR2(10));

CREATE TYPE tp_va_Telefone AS VARRAY(2) of tp_Telefone ;

CREATE TYPE tp_Bacteria AS OBJECT (
Nome                  VARCHAR2(20),
Formato             VARCHAR2(10),
Grau                  INTEGER
);

CREATE TYPE tp_Composto AS OBJECT (
Formula             VARCHAR2(20),
Dose                  INTEGER
);

CREATE TYPE tp_nt_Composto AS TABLE OF tp_Composto;

CREATE TYPE Doenca AS OBJECT (
Sintomas          tp_va_Sintomas,
Nome                  VARCHAR2(20)
) NOT FINAL ;

CREATE TYPE Paciente
AS OBJECT(
Nome                  VARCHAR2(50),
Telefone          tp_va_Telefone
 );
```

```
CREATE TYPE nt_Paciente AS TABLE OF Paciente;

CREATE TABLE tb_Paciente OF Paciente;

CREATE TYPE Antibiotico
AS OBJECT(
Nome                    VARCHAR2(30),
Composto           tp_nt_Composto
 );

CREATE TABLE tb_Antibiotico OF Antibiotico
NESTED TABLE Composto STORE AS tb_lista_Composto;

 CREATE TYPE Genetica
 UNDER Doenca(
 tp_Paciente       nt_Paciente
  );

 CREATE TABLE tb_Genetica OF Genetica;
 NESTED TABLE tp_Paciente STORE AS tb_Lista_Paciente;

CREATE TYPE Bacteriana
UNDER Doenca(
Bacteria           tp_Bacteria
 );

CREATE TABLE tb_Bacteriana OF Bacteriana;

CREATE TABLE tb_BacterianaxAntibiotico (
--ref_Antibiotico REF Antibiotico,
--ref_Bacteriana  REF Bacteriana
);
```

O script de Referência tem quase a mesma estrutura. As diferenças são obtidas alterando-se as regiões cinza pela estrutura apresentada no Quadro 5.2

Quadro 5. 2 Alteracões Script REF

```
CREATE TYPE Genetica
UNDER Doenca
(
Att_Especiali    INTEGER
);

CREATE TABLE tb_Genetica OF Genetica;

CREATE TYPE Paciente
AS OBJECT(
Nome                    VARCHAR2(50),
Telefone           tp_va_Telefone,
ref_Genetica       REF Genetica
 );

CREATE TABLE tb_Paciente OF Paciente;
```

## 5.2 Conclusões

Foi apresentado nesse capítulo um domínio em que se foi possível estudar um caso de uso para a ferramenta VORM. Foi mostrada a manipulação de tabelas aninhadas, *varrays* e atributos compostos na modelagem ER Estendida. Foi possível gerar o script para base de dados Objeto-Relacional demonstrando os resultados possíveis de acordo com a decisão tomada, sobre a forma de mapeamento no relacionamento 1xN, pelo arquiteto da base.

# Capítulo 6 - Conclusões

Este capítulo faz as considerações finais sobre o que foi descrito nos capítulos anteriores. A Seção 6.1 avalia a ferramenta destacando os pontos positivos e as limitações da mesma. Por fim, a Seção 6.2 aponta possíveis trabalhos futuros que podem melhorar o funcionamento de VORM.

## 6.1 Avaliação da Ferramenta

A ferramenta propõe o preenchimento de lacunas deixadas pelas poucas ferramentas especializadas em modelagem para bases de dados objeto-relacional. Provê suporte para diagramas ER Estendidos de forma mais clara. Explicita a criação de VARRAYS e NESTED TABLES sem poluir a informação apresentada. Tem design simples e intuitivo baseado nas vantagens individuais de cada ferramenta concorrente. Tem uma acessibilidade estendida, pois conta com um subsistema de avisos e instruções para manipulação. Tem como seu maior diferencial a criação de scripts PL/SQL para bases Objeto-Relacional. A ferramenta deve ser bem utilizada em ambientes acadêmicos, não limitando, no entanto, o uso em aplicações comerciais. Tem código aberto e sua arquitetura extensível, tornando possível especializar a ferramenta em maior grau para determinadas atividades.

Por ser uma versão beta conta com algumas limitações. Não provê suporte a métodos, funções e procedimentos para cada classe. Conta com uma deficiência no diagrama ER Estendido como a ausência de controles para demonstrar uniões, especialização total e auto-relacionamento. Por não contar com persistência em XML, e ter um formato próprio, a ferramenta VORM tem difícil tradução para outro modelo. Essas limitações, no entanto, não representam reais limites para o desenvolvimento de bases objeto-relacional.

## 6.2 Trabalhos Futuros

Seguem como proposta de trabalho futuro:

- Tratar as limitações propostas pela seção anterior. Dessa forma, a modelagem estará cem por cento conforme o conteúdo de aula apresentado pela disciplina Gerenciamento de Dados e Informação ministrada na Universidade Federal de Pernambuco;
- Prover geração de scripts para mais de um fabricante de Bases de Dados

e não somente Oracle (Oracle, 2011);

- Melhorar a interface gráfica a fim de adequar o programa a uma acessibilidade cada vez maior;
- Integrar diretamente com bases de dados; e
- Prover ambiente colaborativo, migrando para plataforma WEB e provendo mecanismos para manipulação simultânea por pelo menos 2 (dois) projetistas de modelo.

# Referências Bibliográficas


(s.d.). Acesso em 8 de Dezembro de 2011, disponível em Bordland Delphi: http://www.borland.com/br/products/delphi/index.aspx

(s.d.). Acesso em 09 de Dezembro de 2011, disponível em Astah: http://astah.net/

(s.d.). Acesso em 9 de Dezembro de 2011, disponível em Visual Studio 2010: http://www.microsoft.com/visualstudio/pt-br

(s.d.). Acesso em 9 de Dezembro de 2011, disponível em .Net 4.0: http://msdn.microsoft.com/en-us/library/w0x726c2.aspx

(s.d.). Acesso em 9 de Dezembro de 2011, disponível em Nevron: http://www.nevron.com/

(s.d.). Acesso em 9 de Dezembro de 2011, disponível em C#: http://msdn.microsoft.com/pt-br/vcsharp/aa336706

*.Net 4.0*. (s.d.). Acesso em 9 de Dezembro de 2011, disponível em http://msdn.microsoft.com/en-us/library/w0x726c2.aspx

*Astah*. (s.d.). Acesso em 09 de Dezembro de 2011, disponível em http://astah.net/

Baptista, C. d. (2009). *BDOR – Oracle 11g.* Salvador.

Bordingnon, L. G., & Costa, J. (2003). *Viabilidade do Banco de Dados Objeto-Relacional.*

*Bordland Delphi*. (s.d.). Acesso em 8 de Dezembro de 2011, disponível em http://www.borland.com/br/products/delphi/index.aspx

Boscarioli, C., Bezerra, A., Benedicto, M. d., & Delmiro, G. (Agosto de 2006). Uma reflexão sobre Banco de Dados Orientados a Objetos. *IV CONGED* .

*brModelo*. (s.d.). Acesso em 13 de Novembro de 2011, disponível em http://www.sis4.com/brModelo/Default.aspx

*C#*. (s.d.). Acesso em 9 de Dezembro de 2011, disponível em http://msdn.microsoft.com/pt-br/vcsharp/aa336706

Cândido, C. H. (2004). *brModelo: FERRAMENTA DE MODELAGEM CONCEITUAL DE BANCO DE DADOS.* Monografia de Pós Graduação.

Chaudri, A. B., & Zicari, R. (2001). *Succeeding with Object Databases: A Practical Look at Today's Implementations with Java and XML.* Willey Computer Publishing.

Chen, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)* , pp. 9-36.

Devarakonda, R. S. (2001). Object-Relational Database Systems - The Road Ahead. *Crossroads vol 7* , 15-18.

Edelweiss, N., & Galante, R. (s.d.). Bancos de Dados Orientado a Objetos e Relacional-Objeto. *20º SBBD* .

Eisenberg, A., & Melton, J. (2001). *SQL:1999, formerly known as SQL3.*

Elmasri, R., & Navathe, S. B. (2004). *Sistemas de Banco de Dados 4. edição.* São Paulo: Pearson.

*Features Introduced in the Various Server Releases*. (s.d.). Acesso em 29 de Novembro de 2011, disponível em http://orafaq.com/faq/features_introduced_in_the_various_server_releases

Feitosa, D. S., Jesus, H. S., Machado, M. A., & Alencar, S. D. (2006). *Banco de Dados Objetos-Relacionados (BDOR).* Salvador: UFBA.

Fonseca, A. d., Neto, A. d., Souza, L. T., & Dourado, T. L. (2007). *Banco de Dados Objeto-Relacional (BDOR).* Salvador: UFBA.

Fonseca, F., Fidalgo, R., & Carolina, A. (s.d.). *GDI*. Acesso em 15 de Novembro de 2011, disponível em http://www.cin.ufpe.br/~if685/

Fortier, P. J. (1999). *SQL 3: Implementing the Object-Relational Database.* Nova Iorque: McGraw-Hill.

Heuser, C. A. (2008). *Projeto de Banco de Dados.* Artmed.

Iivari, J. (Outubro/96). Why are CASE tools not used? *Communications of the ACM* , 94-103.

Jarzabek, S., & Huang, R. (Agosto/98). The case for user-centered CASE tools. *Communications of the ACM* , 93 - 99.

McClure, S. (Agosto/97). Databases, Object Database vs. Object-Relational. *IDC Bulletin #14821E* .

Nahouraii, E., & Petry, F. (1991). *Object-Oriented Databases.* IEEE.

Neves, N. A., Rocha, G. A., & Segundo, A. d. (2008). *Banco de Dados Objeto-Relacional (BDOR).* Salvador : UFBA.

*Nevron*. (s.d.). Acesso em 9 de Dezembro de 2011, disponível em http://www.nevron.com/

*ODMG*. (s.d.). Acesso em 10 de Outubro de 2011, disponível em http://www.odbms.org/odmg/

Oliveira, A. M., & Mählmann, L. G. (2010). *FERRAMENTA CASE (GERADOR DE CÓDIGO .NET).*

Oracle. (s.d.). *Oracle*. Acesso em 17 de Novembro de 2011, disponível em http://www.oracle.com/

Oracle. (2011). *Oracle*. Acesso em 29 de Junho de 2011, disponível em http://www.oracle.com/

*Power Designer*. (s.d.). Acesso em 24 de Novembro de 2011, disponível em http://www.sybase.com.br/products/modelingdevelopment/powerdesigner

Rao, B. (1994). *Object-Oriented Databases: Technology, Applications, and Products.* Nova Iorque: McGraw-Hill.

*Relational Database*. (s.d.). Acesso em 10 de Outubro de 2011, disponível em http://wiki.tm1forum.com/index.php/Relational_database

Sanches, A. R. (2005). *Breve Histórico das Bases de Dados*. Acesso em 17 de Novembro de 2011, disponível em http://www.ime.usp.br/~andrers/aulas/bd2005-1/aula3.html

Silva, A. G. (2005). *Mapeamento Objeto-Relacional.* Jaguariúna: FAJ.

*SQL Server 2008*. (s.d.). Acesso em 29 de Novembro de 2011, disponível em http://www.microsoft.com/sqlserver/2008/pt/br/default.aspx

Stonebreaker, M. *Object-Relational DBMS - The Next Great Wave.* CA: Informix Software.

*Structs C*. (s.d.). Acesso em 15 de Novembro de 2011, disponível em http://msdn.microsoft.com/en-us/library/ah19swz4(v=vs.71).aspx

*Ted Codd*. (s.d.). Acesso em 29 de Novembro de 2011, disponível em http://wiki.tm1forum.com/index.php/Ted_Codd

Vara, J. M., Vela, B., Cavero, J. M., & Marcos, E. (2007). Model transformation for object-relational database development. *Symposium on Applied Computing* (pp. 1012-1019). Nova Iorque: ACM.

*Varrays e Nested*. (s.d.). Acesso em 27 de Outubro de 2011, disponível em http://docstore.mik.ua/orelly/oracle/prog2/ch19_01.htm

Vessey, I. (Janeiro/95). CASE tools as collaborative support technologies. *Communications of the ACM* , 83-95.

*Visio*. (s.d.). Acesso em 27 de Novembro de 2011, disponível em http://office.microsoft.com/pt-br/visio/

*Visual Studio 2010*. (s.d.). Acesso em 9 de Dezembro de 2011, disponível em http://www.microsoft.com/visualstudio/pt-br

Wang, M. (2001). Implementation of Object-Relational DBMSs. *ACM SIGCSE Bulletin vol 33* , 367 - 370.

Wang, M. *USING UML FOR OBJECT-RELATIONAL DATABASE SYSTEMS.* CA.

Weidler, V. (2004). *Database Comparisons: An Overview.*