



Pós-Graduação em Ciência da Computação

**Um Metamodelo e uma Ferramenta CASE para Projeto
Conceitual de Banco de Dados segundo o Modelo ER**

Por

Cláudia Carolina Nascimento Souza

Dissertação de Mestrado



Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

Recife, Março/2011



Universidade Federal de Pernambuco

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Cláudia Carolina Nascimento Souza

Um Metamodelo e uma Ferramenta CASE para Projeto Conceitual de Banco de Dados segundo o Modelo ER

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: **Prof. Dr. Robson do Nascimento Fidalgo**

Recife, Março/2011

Catálogo na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Souza, Cláudia Carolina Nascimento

Um metamodelo e uma ferramenta CASE para projeto conceitual de Banco de Dados segundo o Modelo ER / Cláudia Carolina Nascimento Souza - Recife: O Autor, 2011.

76 folhas: il., fig., quadros

Orientador: Robson do Nascimento Fidalgo.

Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2011.

Inclui bibliografia.

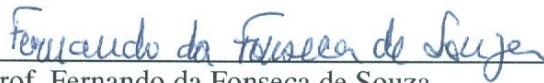
1. Banco de dados. 2. Banco de dados geográficos. 3. Projeto de banco de dados. I. Fidalgo, Robson do Nascimento (orientador). II. Título.

005.74

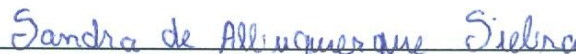
CDD (22. ed.)

MEI2011 – 094

Dissertação de Mestrado apresentada por **Claudia Carolina Nascimento Souza** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título “**Um Metamodelo e uma Ferramenta CASE para Projeto Conceitual de Banco de Dados segundo o Modelo ER**”, orientada pelo **Prof. Robson do Nascimento Fidalgo** e aprovada pela Banca Examinadora formada pelos professores:



Prof. Fernando da Fonseca de Souza
Centro de Informática / UFPE

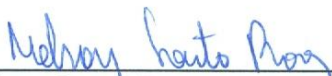


Profa. Sandra de Albuquerque Siebra
Departamento de Ciência da Informação / UFPE



Prof. Robson do Nascimento Fidalgo
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 2 de março de 2011.



Prof. Nelson Souto Rosa

Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

DEDICATÓRIA

Dedico à minha família que tanto amo.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por tudo que Ele me proporciona.

A minha mãe Marie, meu pai Cláudio e minhas tias Lílian e Leila, que são a razão da minha vida, e inspiração para superar todos os obstáculos.

Aos meus irmãos Cleice e Neto que são indispensáveis em todos os momentos da minha vida.

Agradeço ao professor Robson Fidalgo pela orientação, pela confiança, pelo apoio, por sempre estar disponível para todas as dúvidas e pela motivação contínua nas discussões.

Agradeço ao professor Marcos Fagundes pelas orientações e apoio que proporcionou durante minha vida acadêmica.

Agradeço aos meus colegas do Banco da Amazônia da GEPTI/CPADM pelos momentos de descontração e paciência que tiveram comigo nos meus momentos de estresse. Um agradecimento especial ao Harry pela ajuda que forneceu no desenvolvimento da ferramenta e a minha coordenadora Regiane pela liberação as minhas viagens bimestrais a Recife.

Agradeço ao amigo Yuri que me ajudou em todos os momentos no desenvolvimento da ferramenta.

RESUMO

A modelagem conceitual é uma fase importante para um projeto de banco de dados (BD) bem-sucedido, pois é nesta fase que serão extraídos do minimundo (descrição de informações do mundo real) os requisitos necessários para gerar o modelo de dados de acordo com as necessidades do usuário. Nesta fase, a comunidade de Banco de Dados utiliza o MER (Modelo Entidade-Relacionamento) como o padrão de fato para construção de esquemas conceituais. Existem propostas relacionadas de metamodelos e ferramentas CASE (*Computer-Aided Software Engineering*) com base no MER. Porém, se desconhece um trabalho que esteja completamente de acordo com o MER. Isto é, que considere e faça uso correto de todos os seus conceitos (e.g. permitir o uso de tipo união e não fazer uso de chave estrangeira). Visando dar uma contribuição para uma solução do problema descrito no parágrafo anterior, esta dissertação especifica um metamodelo para o MER e, a partir desse metamodelo, implementa um protótipo de ferramenta CASE para o projeto conceitual de BD. Para demonstrar a aplicabilidade desta contribuição foram desenvolvidos cenários de modelagem de esquemas conceituais que visam mostrar a corretude e completude do metamodelo e da ferramenta CASE propostos.

Palavras chave: Modelo Entidade-Relacionamento, Metamodelo, Ferramenta CASE.

ABSTRACT

Conceptual modeling is an important phase for a successful database project (DB), because it is in this stage that the requirements needed to generate the data model according to user needs will be gathered from the universe of discourse (description of real world information). In this phase, the database community uses the ERM (Entity-Relationship Model) as the standard for the construction of conceptual schemes. There are metamodels and CASE (Computer-Aided Software Engineering) tools proposals based on the to ERM. However, a proposal that is completely ERM compliant is unknown. That is, one that considers and that correctly applies of all its concepts (e.g., allow the use of union type and don't use foreign key). In order to contribute to solving the problem described in the preceding paragraph, this work specifies a metamodel for the ERM and, based on this metamodel, implements a prototype CASE tool for conceptual design of DB according to the ERM. To demonstrate the applicability of this contribution, scenarios have been developed for modeling conceptual schemes aimed at showing the correctness of both the proposed metamodel and the CASE tool.

Key word: Entity-Relationship Model, Metamodel, CASE tool.

SUMÁRIO

1	Introdução	13
1.1	Apresentação	13
1.2	Motivação.....	14
1.3	Objetivos do Trabalho	14
1.4	Estrutura da Dissertação.....	14
2	Conceitos Básicos	16
2.1	Conceitos Básicos sobre modelagem conceitual	16
2.1.1	Principais conceitos do MER	16
2.2	Projeto de Modelagem Eclipse	21
2.2.1	Eclipse Modeling Framework (EMF).....	21
2.2.2	Graphical Modeling Framework (GMF)	24
2.3	Considerações Finais.....	25
3	Metamodelos e Ferramentas CASE para MER	26
3.1	Metamodelos Padrões para MER	26
3.1.1	Metamodelo CWM – Common Warehouse Metamodel	27
3.1.2	O Padrão IMM – Information Management Metamodel.....	28
3.1.3	Comparação dos Metamodelos Correlatos	30
3.2	Ferramentas CASE para MER.....	31
3.2.1	brModelo	31
3.2.2	DBMain.....	34
3.2.3	SmartDraw	35
3.2.4	Comparação das Ferramentas Correlatas	40
3.3	Considerações Finais.....	41
4	O Metamodelo ERMM e a Ferramenta ERCASE	42
4.1	O Metamodelo ERMM.....	42

4.1.1	Especificação das Restrições ERMM.....	50
4.1.1.1	Associar Relacionamentos a Relacionamentos	50
4.1.1.2	Associar Entidades a Entidades	50
4.1.1.3	Ciclicidade entre Atributos	51
4.1.1.4	Entidades com nomes iguais	51
4.1.1.5	Atributos com nomes iguais	52
4.1.1.6	Relacionamentos sem associações para Entidades	52
4.1.1.7	Herança sem Link de Especialização	53
4.1.1.8	Herança sem Link de Generalização	53
4.1.1.9	Categoria sem Link de Generalização	54
4.1.1.10	Categoria sem Link de Especialização	54
4.2	A Ferramenta ERCASE	55
4.2.1	Componente Editor Gráfico	56
4.2.2	Componente Validador de Esquemas	58
4.2.3	Componente Gerenciador de Metadados.....	60
4.3	Análise Comparativa entre Trabalhos Relacionados e ERMM/ERCASE	61
4.3.1	Metamodelos MER e ERMM	61
4.3.2	Ferramentas CASE MER e ERCASE	63
5	Estudo de Caso.....	65
5.1	Estudo de Caso Hipotético 1: Elementos Básicos do MER	65
5.1.1	Estudo de Caso Hipotético 1: Verificação do Esquema	67
5.2	Estudo de Caso Hipotético 2: Elementos Avançados do MER	68
5.2.1	Estudo de Caso Hipotético 2: Verificação do Esquema	69
5.3	Considerações Finais.....	70
6	Conclusão.....	71
6.1	Considerações Finais.....	71

6.2	Contribuições	72
6.3	Trabalhos Futuros	72
Apêndice A.....		76
O Metamodelo ERMM.....		76

LISTA DE FIGURAS

Figura 2.1 - Conceitos e elementos gráficos da notação de Elmasri & Navathe.....	20
Figura 2.2 – Interface Gráfica do <i>framework</i> EMF.....	22
Figura 2.3 – Níveis de Geração de Código	23
Figura 2.4 – Exemplo de Editor Gráfico em GMF – Ferramenta ERCASE	24
Figura 2.5 – <i>Dashboard</i> GMF	25
Figura 3.1 – Classes do pacote ER de CWM.....	27
Figura 3.2 – Metaclasses do Metamodelo ER de IMM	29
Figura 3.3 – Avaliação da ferramenta brModelo.....	32
Figura 3.4 – Ocorrência de Erros Sintáticos na ferramenta brModelo	33
Figura 3.5 – Fragmento em XMI do brModelo.....	33
Figura 3.6 - Avaliação da Ferramenta DBMain	35
Figura 4.1. MER X ERMM: Esquema, Entidade e Relacionamento.	44
Figura 4.2. MER X ERMM: Entidade, Relacionamento e Atributo.	45
Figura 4.3. MER X ERMM: Atributo.	46
Figura 4.4. MER X ERMM: Entidade e Herança (Generalização/Especialização).	48
Figura 4.5. MER X ERMM – Entidade e Categoria.	49
Figura 4.6 - Restrição imposta pela metaclasses <i>Link de Relacionamento</i>	50
Figura 4.7 – Componentes da Arquitetura de ERCASE.	56
Figura 4.8 – Ambiente de Modelagem de ERCASE	57
Figura 4.9 – Paleta de ERCASE	57
Figura 4.10 – Aba para a Edição de Propriedades.....	58
Figura 4.11 – Barra de Ferramentas para Seleção, Organização e Alinhamento.....	58
Figura 4.12 – Validação do Esquema.	59
Figura 4.13 – Elemento sinalizado com Erro.....	59
Figura 4.14 – Representação do MER Cliente no ERMM em XMI.	60
Figura 5.1 – Esquema apenas com elementos básicos do MER.....	67
Figura 5.2 – Validação do esquema do estudo de caso 1.....	68

Figura 5.3 – Esquema apenas com elementos avançados do MER	69
Figura 5.4 – Validação do esquema do estudo de caso 2.....	70

LISTA DE QUADROS

Quadro 3.1 – Análise das propostas de Metamodelos para MER	31
Quadro 3.2 – Análise das propostas de ferramentas CASE para MER	40
Quadro 4.1 – Análise Comparativa dos Metamodelos Relacionadas e ERMM	62
Quadro 4.2 – Análise Comparativa das Ferramentas Relacionadas e ERCASE.....	63

PRINCIPAIS ABREVIATURAS

BD	Banco de Dados
CASE	Computer-Aided Software Engineering
CWM	Common Warehouse Metamodel
EMF	Eclipse Modeling Framework
ERCASE	Entidade-Relacionamento CASE
ERMM	Entidade-Relacionamento Metamodel
GMF	Graphical Modeling Framework
IMM	Information Management Metamodel
MER	Modelo Entidade-Relacionamento
OCL	Object Constraint Language
OMG	Object Management Group
SGBD	Sistema Gerenciador de Banco de Dados
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

1 Introdução

Este capítulo descreve as razões para o desenvolvimento deste trabalho, apresentando em linhas gerais a pesquisa desenvolvida.

1.1 Apresentação

Segundo Heuser [16] e Elsmari&Navathe [8], a modelagem conceitual é uma fase importante no planejamento de uma aplicação de BD (Banco de Dados) bem-sucedida. Como resultado desta fase tem-se um esquema conceitual, o qual corresponde em uma descrição dos requisitos de dados dos usuários e inclui descrições de entidades, atributos, relacionamentos e restrições. Como esses conceitos não incluem detalhes de implementação, são mais fáceis de entender e auxiliam a comunicação entre os projetistas e os usuários. Nesse contexto, um esquema conceitual segundo o MER (Modelo Entidade-Relacionamento) pode ser usado para garantir que os requisitos de dados sejam atendidos e não entrem em conflito.

Com base no MER espera-se, no mínimo, que o esquema conceitual modelado esteja sintaticamente correto, pois assim, posteriormente, este esquema conceitual pode ser transformado em um esquema lógico de um SGBD (Sistema de Gerenciamento de Banco de Dados). Desta forma, pode-se perceber que o esquema conceitual caracteriza-se como um artefato importante para o projeto de uma aplicação de banco de dados. Contudo, apesar de existirem trabalhos que visam definir ferramentas CASE (*Computer-Aided Software Engineering*) [3], [6], [29] destinadas ao projeto de esquemas conceituais segundo o MER, se desconhece um trabalho que defina um metamodelo e uma ferramenta CASE que dê suporte e faça uso correto de todos os conceitos do MER (e.g. relacionamento n-ário e tipo união). Com o intuito de contribuir para minimizar estas questões em aberto, este trabalho propõe 1) um metamodelo em conformidade ao MER segundo Elsmari&Navathe [8] e 2) uma ferramenta CASE *open-source* (código-aberto), extensível e multiplataforma para o desenvolvimento de esquemas conceituais de acordo com o metamodelo proposto.

1.2 Motivação

As principais motivações para o desenvolvimento deste trabalho são 1) o projeto conceitual é uma fase importante dentre as fases de projeto de um BD e 2) existem propostas de metamodelos [5], [17] e de ferramentas CASE [3], [6], [29] destinadas ao projeto conceitual de BD. Contudo, a partir de uma análise dessas propostas observou-se que não estão totalmente em conformidade com o MER ou especificam construtores do MER correspondentes ao nível lógico e/ou físico de BD.

1.3 Objetivos do Trabalho

O objetivo geral desta pesquisa é a definição de um metamodelo e de uma ferramenta CASE para apoiar o projeto conceitual de BD no MER, segundo os conceitos de Elsmari&Navathe [8].

A partir do objetivo geral pode-se detalhar os seguintes objetivos específicos:

- Especificar restrições que impossibilitem a ocorrência de erros sintáticos no desenvolvimento de esquemas conceituais;
- Desenvolver cenários que avaliem o metamodelo e a ferramenta CASE propostos.

1.4 Estrutura da Dissertação

De forma a alcançar os objetivos deste trabalho, os demais capítulos estão organizados da seguinte maneira: o Capítulo 2 aborda os principais conceitos relacionados ao MER bem como as tecnologias utilizadas para o desenvolvimento de ferramentas CASE; o Capítulo 3 avalia os principais trabalhos relacionados, destacando suas vantagens e limitações, no contexto desta dissertação; o Capítulo 4 descreve o metamodelo e a ferramenta CASE propostos; o Capítulo 5 apresenta os estudos de caso que permitem avaliar as propostas deste trabalho;

e, finalmente, o Capítulo 6 apresenta as conclusões desta dissertação, suas principais contribuições, bem como as indicações de trabalhos futuros.

2 Conceitos Básicos

Neste capítulo são abordados os principais conceitos sobre modelagem conceitual e sobre os *frameworks* EMF (*Eclipse Modeling Framework*) [31] e GMF (*Graphical Modeling Framework*) [11]. Neste sentido, o presente capítulo está organizado como segue. Na seção 2.1 são destacados os conceitos básicos sobre o MER. Na seção 2.2 são abordados os *frameworks* EMF e GMF. E finalmente, na seção 2.3 são feitas as considerações de encerramento deste capítulo.

2.1 Conceitos Básicos sobre modelagem conceitual

Um BD pode ser descrito em três níveis de abstração [8]: nível conceitual (modelo conceitual), nível lógico (modelo lógico) e nível físico (modelo físico). Dentre estes níveis, o nível conceitual é o foco deste trabalho, pois: 1) é o primeiro nível de modelagem de um BD. Isto é, este nível merece uma atenção especial, pois um erro neste nível é refletido em todos os demais; 2) permite descrever o esquema de um BD independente da tecnologia do SGBD a ser usado. Ou seja, é o nível onde o projetista do BD está com a sua atenção voltada para o entendimento do problema e a especificação da sua solução independentemente de tecnologia; e, 3) existem poucas ferramentas CASE para auxiliar o projetista a especificar esquemas de BD verdadeiramente conceituais.

Dentre os modelos de dados que podem ser usados para especificar esquemas de BD no nível conceitual, o MER [4], [8] é um dos mais aceitos (se não for o mais aceito) pela comunidade de BD. A seguir são apresentados os principais conceitos desse modelo de dados.

2.1.1 Principais conceitos do MER

O MER consiste basicamente de uma coleção de entidades, relacionamentos entre essas entidades e atributos. Esses conceitos básicos, bem como outros avançados são resumidos a seguir. Uma explicação detalhada desses conceitos pode ser encontrada em [8].

- **Entidade Regular:** corresponde à representação de um conceito tangível (e.g., um carro ou um funcionário) ou intangível (e.g., uma conta bancária ou um curso de graduação) da realidade modelada;

- **Atributo:** corresponde a uma propriedade de uma entidade ou relacionamento. Os atributos podem ser classificados nos seguintes tipos: simples ou composto, monovalorado ou multivalorado, derivado, discriminador e por fim, identificador. A seguir, esses tipos são rapidamente explicados e exemplificados. Note que um atributo pode ter mais de uma classificação (e.g. simples e multivalorado) e que a classificação do atributo depende do contexto a ser modelado e não do nome do atributo (e.g. endereço, dependendo do contexto, pode ser um atributo simples ou composto).

- Simples: corresponde a uma propriedade que é considerada indivisível (e.g., cpf e cep);
- Composto: corresponde a uma propriedade que é divisível em subpartes (e.g. telefone dividido em prefixo e sufixo, e endereço dividido em principal e complemento);
- Monovalorado: corresponde a uma propriedade que terá apenas um valor por instância de entidade (e.g., cpf e rg);
- Multivalorado: corresponde a uma propriedade que pode ter mais de um valor por instância da entidade (e.g. telefone e endereço);
- Derivado: corresponde a uma propriedade que terá o seu valor obtido a partir de valores de outros atributos (e.g., quantidade de empregados);
- Identificador (ou chave): corresponde a uma propriedade que tem um valor único e será usada para distinguir cada instância da entidade (e.g., cnpj ou cpf);
- Discriminador (ou chave parcial): corresponde a uma propriedade que funciona como um identificador parcial de cada instância da entidade (e.g., código ou cpf de cliente poderão ser atributos discriminadores da entidade Empregado).

- **Entidade Fraca:** corresponde a uma entidade regular que não tem um identificador que permita distinguir cada instância de suas entidades. Neste caso, uma entidade fraca precisa do identificador de uma outra entidade

(chamada de entidade forte) para formar o seu identificador. Ou seja, a entidade fraca é dependente do identificador da entidade forte;

- **Relacionamento:** corresponde a uma associação entre instâncias de entidades. Um relacionamento podem ser classificado em: unário (i.e. auto-relacionamento), binário (i.e. entre duas entidades) e n-ário (i.e. entre várias entidades);

- **Relacionamento Fraco (ou Identificador):** corresponde a um relacionamento entre uma entidade regula e uma entidade fraca;

- **Papel:** corresponde a função que uma dada entidade desempenha no relacionamento (e.g. uma instância de um empregado pode ter o papel de chefe ou de subordinado em um auto-relacionamento);

- **Entidade Associativa (ou Agregação):** corresponde a um relacionamento que também pode ser entendido como uma entidade;

- **Cardinalidade:** corresponde ao número máximo de ocorrências que uma instância de uma entidade pode ter no relacionamento. A cardinalidade pode ser representada por qualquer número natural. Contudo convencionou-se os valores 1 e N para representar a cardinalidade um ou vários, respectivamente. Elsmari&Navathe representam este conceito através da visão “Look Across”;

- **Participação:** corresponde a obrigatoriedade (participação total) ou não (participação parcial) do relacionamento de todas as instâncias de uma entidade com pelo menos uma outra instância da mesma entidade ou de outra entidade. Elsmari&Navathe representam este conceito através da visão “Look Here”;

- **Herança:** corresponde a um mecanismo de modelagem que permite especializar ou generalizar uma entidade. Isto é, definir uma sub-entidade (especialização) ou uma super-entidade (generalização). Além da herança simples pode –se ter Herança Múltipla (uma entidade herda de duas ou mais outras entidades). A herança simples ou múltipla pode ser:

- Disjunta ou Sobreposta: no primeiro caso (disjunta), cada instância da super-entidade é especializada exclusivamente para uma instância de uma sub-entidade. No segundo caso

(sobreposta), pelo menos uma instância da super-classe é especializada em mais de uma instância de sub-classes;

- Total ou Parcial: no primeiro caso (total), todas as instância da super-classe estão especializada em pelo menos uma sub-entidade. No segundo caso (parcial), pelo menos uma instância da super-entidade não é especializada em uma sub-entidade.

- **Categoria:** corresponde a uma herança seletiva, onde a sub-entidade chama-se categoria e representa um subconjunto da união das suas superclasses.

Muitos dos conceitos recém apresentados estão representados de diferentes formas nas diversas notações gráficas para o MER (e.g. Chen [4], IDEF1X [19], Engenharia da Informação (ou Pé de Galinha) [22], Heuser [16] e Elmasri & Navathe [8]). A primeira notação do MER foi proposta por Chen [3] e basicamente é composta de entidades (retângulos), relacionamentos (losangos), atributos (círculos) e linhas de conexão (linhas). A partir do trabalho de Chen, outros trabalhos foram propostos com o objetivo de incluir novos conceitos e/ou definir novos elementos notacionais. Por exemplo, herança, entidade associativa e categoria. Dentre estes trabalhos, destacam-se o IDEF1X, Engenharia da Informação, Heuser e Elmasri & Navathe. Como o trabalho de Elmasri & Navathe é o único com uma notação que dá suporte a todos os conceitos apresentados anteriormente e, além disso, sua notação é uma das mais usadas pela comunidade de BD, este trabalho será baseado nesta notação. A seguir, é apresentado um resumo da notação de Elmasri & Navathe. Um estudo detalhado desta notação pode ser encontrado em [8].

A notação de Elmasri & Navathe é uma das mais aderentes à primeira notação do MER. Diferente de algumas notações (e.g. IDEF1X e Engenharia de Informação) que são binárias, a notação de Elmasri & Navathe é N-ária. Como vantagem disto, esta notação permite a modelagem de relacionamentos com grau maior que dois e de atributos em relacionamentos, o que não é possível em uma notação binária. Além disso, é importante ressaltar que a notação de Elmasri & Navathe implementa o conceito de participação e

cardinalidade segundo as visões “Look Here” e “Look Across” [30], respectivamente. Isto é, a restrição de participação é definida perguntando-se para a entidade de origem se todas as suas instâncias devem estar associadas a pelo menos uma ocorrência do relacionamento. Por sua vez, a restrição de cardinalidade é definida perguntando-se para a entidade de destino qual é a quantidade máxima de ocorrências que esta entidade pode ter no relacionamento. Na Figura 2.1, baseada em [8], mostra-se os conceitos e elementos gráficos que a notação de Elmasri & Navathe suporta.

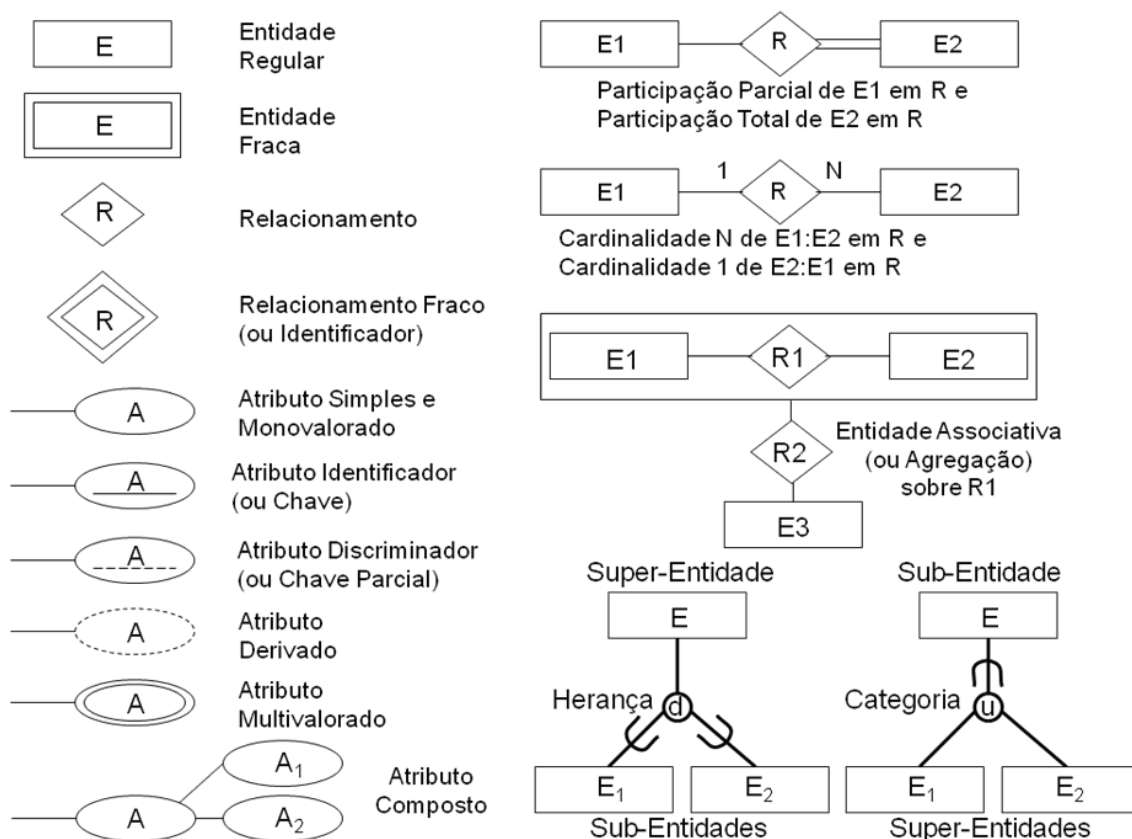


Figura 2.1 - Conceitos e elementos gráficos da notação de Elmasri & Navathe.

Segundo Heuser [16], um dos quesitos de garantia da qualidade de um esquema de BD é garantir que este esteja livre de erros sintáticos e semânticos. Os erros sintáticos ocorrem quando o esquema do BD não respeita as regras de construção do modelo de dados usado. Por exemplo, no MER, a associação direta entre relacionamentos ou entre entidades são erros sintáticos. Já os erros semânticos ocorrem quando o modelo reflete a realidade de forma inconsistente. Por exemplo, considerando as leis do Brasil, têm-se os seguintes erros semânticos: um cliente com mais de um CPF e um homem

casado com mais de uma mulher. Note que a detecção de erros semânticos depende de um contexto e por isso é mais difícil de ser automatizada do que a detecção de erros sintáticos. Por esta razão, no âmbito deste trabalho serão estudados e tratados apenas os erros sintáticos. A seguir são apresentados os *frameworks* do Eclipse destinados ao desenvolvimento de metamodelos e editores gráficos para ferramentas CASE.

2.2 Projeto de Modelagem Eclipse

O Projeto de Modelagem Eclipse (*Eclipse Modeling Project*) [7] visa a promoção das tecnologias de desenvolvimento baseado em modelos dentro da comunidade Eclipse, fornecendo um conjunto unificado de *frameworks* de modelagem, ferramentas e implementações de padrões. Dentre os *frameworks* de modelagem destaca-se o EMF (*Eclipse Modeling Framework*) [31] e o GMF (*Graphical Modeling Framework*) [11] que são destinados, respectivamente, ao desenvolvimento de metamodelos e editores gráficos para a criação de ferramentas CASE. A seguir é apresentada uma visão geral dos recursos destes *frameworks*.

2.2.1 Eclipse Modeling Framework (EMF)

O EMF é um framework para construir ferramentas CASE e outras aplicações baseadas em modelos de dados estruturados. Este framework utiliza o metamodelo *ECore* para definição de metamodelos. O *ECore* apresenta os principais conceitos chaves: 1) *EClass*, é usada para representar uma metaclassse; 2) *EAttribute*, é usado para representar um atributo de uma *EClass*; e, 3) *EReference*, é usada para representar associações entre *EClass*.

O EMF apresenta uma interface que disponibiliza os elementos (e.g., *EClass*, *EAttribute*) que serão utilizados no desenvolvimento de metamodelos. A Figura 2.2 mostra o fragmento do metamodelo ERMM desenvolvido em EMF.

O EMF possui três componentes fundamentais: 1) O *framework EMF Core*. Este *framework* fornece em tempo de execução notificações das alterações do metamodelo e disponibiliza ainda uma API para manipular de forma genérica os objetos EMF; 2) O *EMF.Edit*, que fornece um conjunto de classes genéricas para a construção dos editores gráfico para os metamodelos EMF; e finalmente, 3) *EMF.Codegen* responsável pela geração do código necessário para a construção do editor gráfico.

A Figura 2.2 ilustra o editor EMF com parte do metamodelo ERMM que é proposto no capítulo 4. A área 1 é destinada para a criação do metamodelo que, por sua vez, é subsidiado pelos elementos do metamodelo *Ecore* (i.e., *EClass*, *EAttribute*, *EReference*), disponibilizados na paleta identificada pelo número 2.

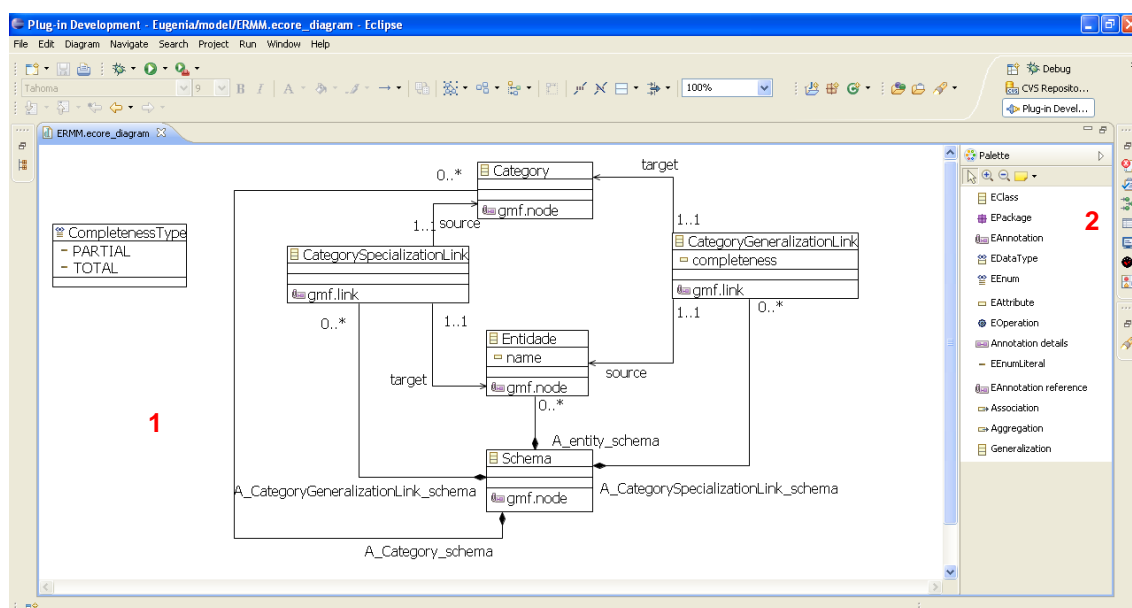
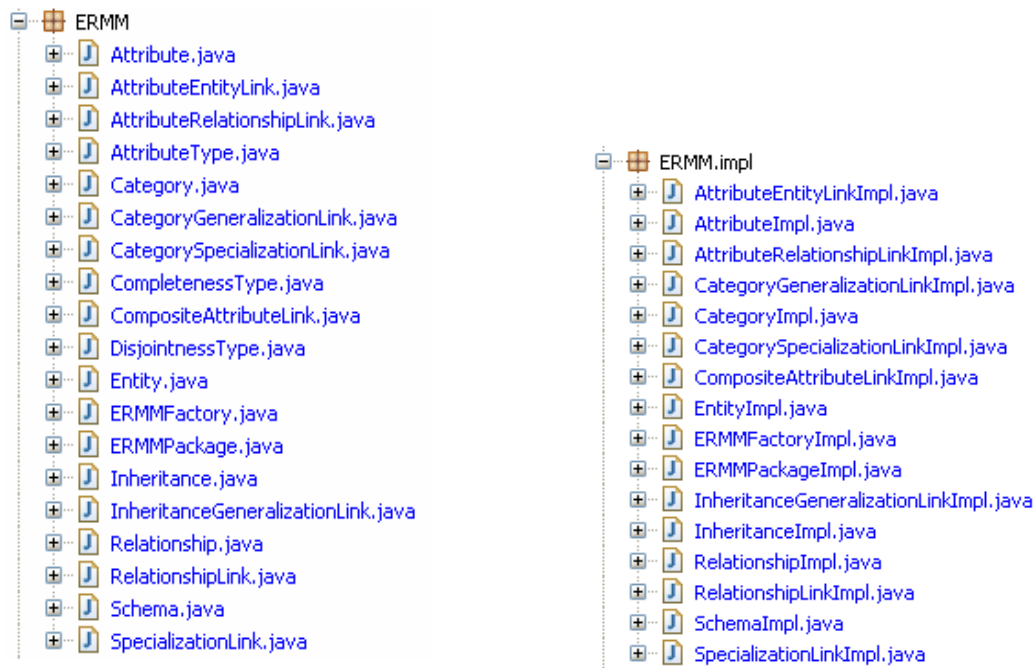


Figura 2.2 – Interface Gráfica do *framework* EMF.

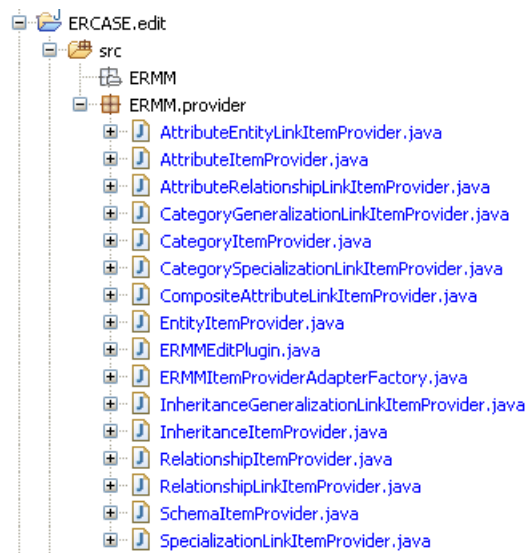
O EMF dá suporte a três níveis de geração de código: 1) Modelo, o qual fornece classes de interface e de implementação para todos os elementos do metamodelo da ferramenta CASE a ser construída; 2) Adaptadores, os quais geram as classes de implementação capazes de adaptar as metaclasses do metamodelo especificado; e 3) Editor, o qual é responsável por produzir a estrutura básica das metaclasses do metamodelo que serão utilizadas na etapa de

geração do editor gráfico no GMF. A Figura 2.3 mostra as classes do ERMM em cada um desses níveis.



(a) ERMM - Nível Modelo

(b) ERMM - Nível Adaptadores



(a) – ERMM – Nível Editor

Figura 2.3 – Níveis de Geração de Código

2.2.2 Graphical Modeling Framework (GMF)

O GMF [11] é um *framework* que permite o desenvolvimento de editores gráficos, a partir de metamodelos definidos em EMF (ver Seção 2.2.1). A Figura 2.4 mostra a interface gráfica da ferramenta ERCASE proposta no capítulo 4, que foi desenvolvida usando o editor GMF.

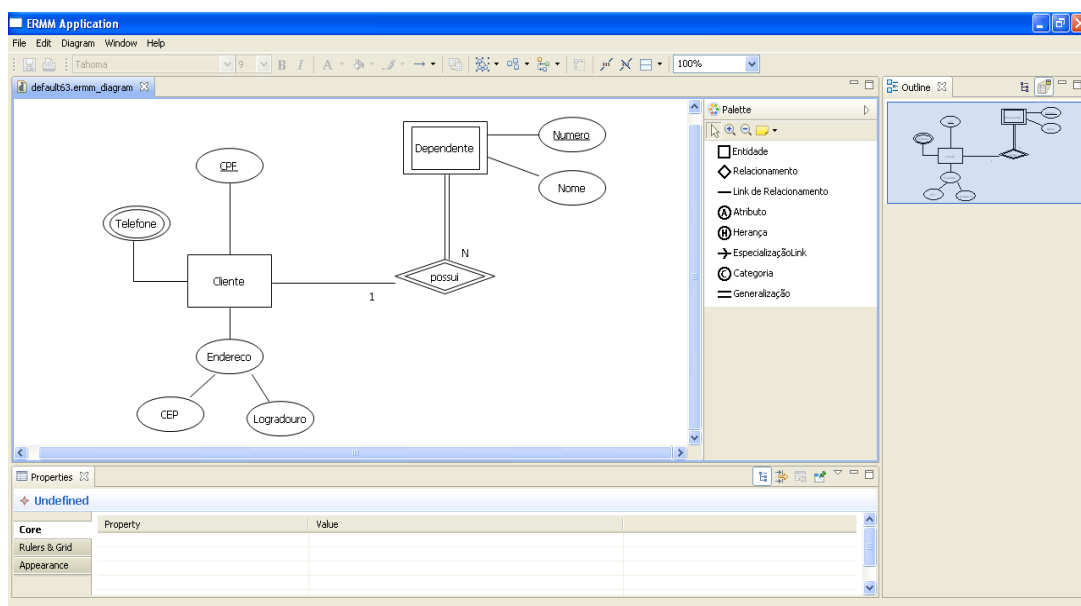


Figura 2.4 – Exemplo de Editor Gráfico em GMF – Ferramenta ERCASE

Para o desenvolvimento de editores gráficos no GMF é necessário seguir um processo. Este processo é executado com o auxílio de um painel, o qual guia o projetista a segui-lo na ordem correta. Sua execução é feita na seguinte ordem: 1) importar o metamodelo de domínio definido em EMF; 2) definir os elementos gráficos que serão exibidos na ferramenta CASE. Para cada metaclassa do metamodelo de domínio é definido um elemento gráfico; 3) definir os elementos que estarão disponíveis na paleta do editor; 4) definir o mapeamento entre as metaclasses do metamodelo de domínio, os elementos gráficos e os elementos da paleta de ferramentas; e finalmente, 5) processar o modelo gerador da ferramenta CASE juntamente com o código-fonte do metamodelo e dos elementos gráficos do editor. A

Figura 2.5 mostra o painel que auxilia no processo de execução para a geração de editores gráficos.

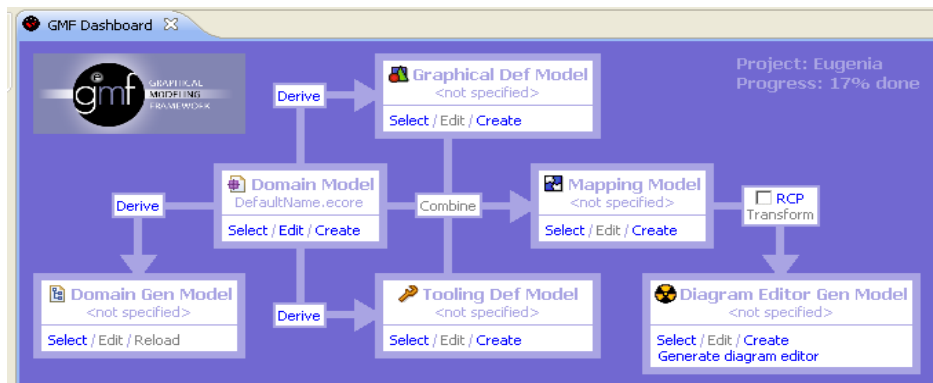


Figura 2.5 – Painel do GMF

2.3 Considerações Finais

Neste capítulo, foram abordados os conceitos do MER segundo Elsmari&Navathe [8]. Este foi adotado como base do desenvolvimento deste trabalho, devido sua notação gráfica ser semanticamente rica, ou seja, oferecer suporte a todos os construtores vistos na seção 2.1.1 e por ser uma notação bem difundida na comunidade acadêmica de BD. Além disso, foram abordados os *frameworks* EMF e GMF que são utilizados, respectivamente, para a criação de metamodelos e editores gráficos. Esses *frameworks* oferecem ao usuário uma interface gráfica amigável para a definição o do metamodelo e do editor gráfico. No próximo capítulo, são discutidos alguns metamodelos e ferramentas CASE que estão relacionados a este trabalho.

3 Metamodelos e Ferramentas CASE para MER

Neste capítulo apresenta-se uma visão geral da área de pesquisa desta dissertação, destacando os principais trabalhos relacionados. Neste sentido, as seções deste capítulo estão organizadas da seguinte forma: A seção 3.1 descreve os metamodelos padrão de referência que estão sendo considerados neste trabalho. A seção 3.2 discute as ferramentas CASE para modelagem conceitual de BD que estão diretamente relacionadas a este trabalho. E finalmente, a seção 3.3 apresenta as considerações de encerramento deste capítulo.

3.1 Metamodelos Padrões para MER

Nesta seção são discutidas duas propostas de metamodelos padrão para a especificação do MER. A primeira proposta versa sobre o metamodelo CWM (*Common Warehouse Metamodel*) [5], o qual foi proposto pela OMG (*Object Management Group*) [28] e define um conjunto de classes para integração de sistemas de DW (*Data Warehouse*). A segunda proposta refere-se ao metamodelo IMM (*Information Management Metamodel*) [17], o qual é uma RFP (*Request For Proposals*) da OMG para definir, gerenciar e integrar um conjunto de metamodelos e *profiles* que alinhem a visão de negócios e de tecnologia à modelagem de dados. Estas propostas são brevemente discutidas a seguir.

3.1.1 Metamodelo CWM – Common Warehouse Metamodel

O CWM é um padrão de metadados cujo objetivo é permitir a integração de aplicações baseadas em DW. Este é dividido em vários pacotes de metamodelos, dentre estes, o pacote Entidade-Relacionamento (*Entity-Relationship*) é o único que está diretamente relacionado a este trabalho. Na Figura 3.1 [5] mostra-se as metaclasses do pacote Entidade-Relacionamento de CWM.

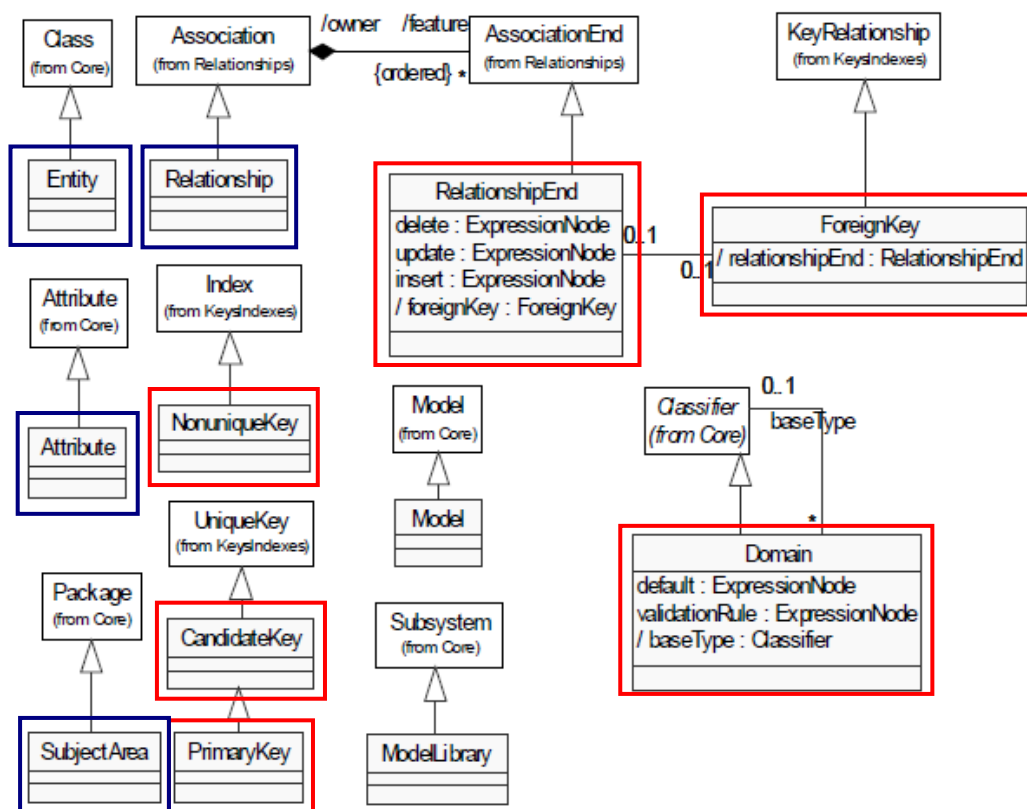


Figura 3.1 – Classes do pacote ER de CWM

Na Figura 3.1, a metaclassse central é Área de Assunto (*SubjectArea*) que é composta por um conjunto de Entidades (*Entity*), Atributos (*Attribute*), Relacionamentos (*Relationship*). A metaclassse Classe (*Class*) é estendida para representar o conceito de Entidade que, pode ser constituída por Atributos, Chaves Candidatas (*CandidateKey*), Chaves Primárias (*PrimaryKey*), Chaves Estrangeiras (*ForeignKey*) e Chaves Não-Únicas (*NonUniqueKey*). A especificação CWM apresenta com mais detalhes o relacionamento entre as metaclasses. A metaclassse Relacionamento-Chave (*KeyRelationship*) do pacote Índice-Chave (*KeyIndexes*) é especializada para corresponder ao

conceito de chave estrangeira. A metaclasses Índice (*Index*) é especializada para representar o conceito de Chaves Não-Únicas. A metaclasses Relacionamento é especializada pela metaclasses Associação (*Association*) para representar conexões entre objetos Entidades. Cada objeto Relacionamento é composto por um Relacionamento-Alvo (*RelationshipEnd*) que, por sua vez, possui as seguintes propriedades: exclusão (*delete*), atualização (*update*) e inserção (*insert*), as quais, respectivamente, registram a manipulação de objetos Entidades. A metaclasses Chave Estrangeira garante a integridade dos dados manipulados referente à relação entre objetos Entidade. A classe *Domain* representa os tipos de dados que devem estar associados a objetos Atributos. O CWM agrupa os Esquemas na metaclasses Modelos (*Model*), as quais são reunidas em uma biblioteca de modelos, representada pela classe Biblioteca Modelo (*ModelLibrary*).

Sobre o metamodelo recém abordado pode-se dizer que este, apesar de ser uma iniciativa de metamodelo para o MER, define um conjunto de conceitos que são mais destinados ao projeto lógico do que conceitual de um BD. A Figura 3.1 destaca, em vermelho e azul, as metaclasses que representam conceitos de projeto lógico e conceitual, respectivamente. Além disso, pode-se observar que este é basicamente a definição de um conjunto de heranças, não representando a maioria dos relacionamentos entre suas metaclasses.

3.1.2 O Padrão IMM – Information Management Metamodel

O IMM [17] é um padrão da OMG que engloba vários pacotes de metamodelos, onde o Metamodelo Entidade-Relacionamento (*Entity-Relationship Metamodel*) está no foco do trabalho. Além disso, este padrão pretende englobar os metamodelos de CWM. Ou seja, IMM está sendo especificado com o objetivo de ser o padrão sucessor de CWM. A Figura 3.2 mostra, em amarelo, as metaclasses do metamodelo Entidade-Relacionamento que são relevantes ao contexto deste trabalho. Neste pacote, o IMM define a metaclasses Área de Assunto (*SubjectArea*) como um catálogo de dados que armazena os principais elementos, sendo também o ponto de partida para navegação do metamodelo. De forma geral, a metaclasses Área de Assunto contém as metaclasses

Entidade (*Entity*), Relacionamento (*Relationship*), Atributo (*Attribute*) e Restrições (*Constraints*). A metaclasses Entidade pode possuir Atributos, os quais podem ser de várias entidades. Os atributos têm um Domínio (*Domain*), o qual pode ser de vários atributos. Um atributo pode estar em várias chaves (*Key*), as quais podem ser formadas por vários atributos. Além disso, uma entidade pode ser supertipo (*Supertype*) de outra entidade. Uma entidade pode participar de mais de um relacionamento, através da metaclasses Papel (*Role*). Um relacionamento tem que ter no mínimo duas, mas pode ter vários papéis, definindo o grau do relacionamento. As restrições que garantem a consistência dos modelos subsidiados em IMM são representadas pela metaclasses Restrição (*Constraint*).

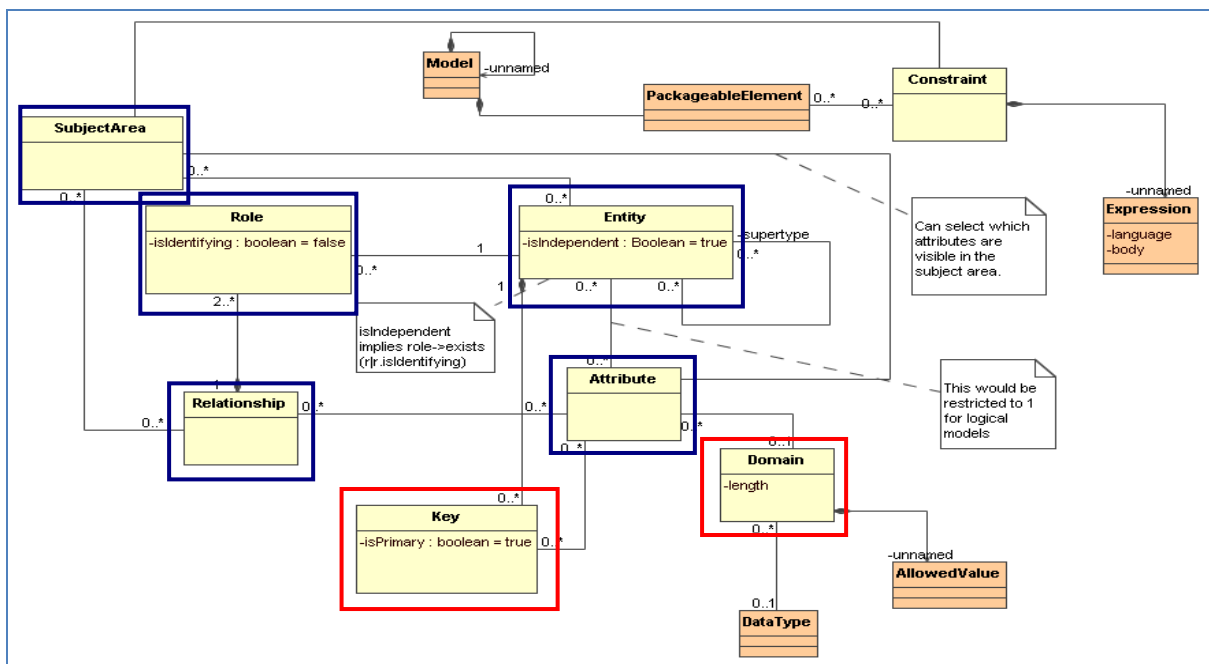


Figura 3.2 – Metaclasses do Metamodelo ER de IMM

Sobre o metamodelo recém apresentado pode-se dizer que este ainda está na fase de proposta (RFP) e, por isso, ele ainda está propenso a mudanças. Semelhante ao metamodelo ER de CWM, este também mistura conceitos da fase de projeto conceitual com conceitos só devem existir na fase de projeto lógico ou físico. A Figura 3.2 destaca, em vermelho e azul, as metaclasses que representam conceitos de projeto lógico e conceitual, respectivamente. Além disso, o metamodelo permite que um atributo esteja em

mais de uma entidade ou relacionamento ao mesmo tempo, pois este (Figura 3.2) tem um relacionamento N:M entre as metaclasses atributo, entidade e relacionamento, sem nenhuma restrição (e.g. em OCL - *Object Constraint Language*) [27] para impedir este erro sintático.

3.1.3 Comparação dos Metamodelos Correlatos

De forma a fazer uma avaliação justa dos metamodelos de CMW e IMM, utilizou-se os conceitos sobre o MER discutidos no Capítulo 2. Com base nesses conceitos, definiu-se os critérios de avaliação mostrados no Quadro 3.1. Neste quadro, os marcadores “+” e “-” denotam, respectivamente, se a proposta oferece suporte ou não oferece suporte.

Itens	CWM	IMM
Entidade Regular	+	+
Entidade Associativa	-	-
Entidade/Relacionamento Fraco	-	+
Atributo Simples	+	+
Atributo Composto	-	-
Atributo Monovalorado	+	+
Atributo Multivalorado	-	-
Atributo Derivado	-	-
Atributo Identificador	+	+
Atributo Discriminador	-	-
Auto-Relacionamento	-	+
Relacionamento Binário	-	+
Relacionamento <i>N-ário</i>	-	+
Participação do Relacionamento	+	+
Cardinalidade do Relacionamento	+	+
Papéis	-	+
Herança Simples	-	+
Herança Múltipla	-	+

Herança Total ou Parcial	—	—
Herança Disjunta ou Sobreposta	—	—
Categoria	—	—
Impede Erro Sintático	—	—

Quadro 3.1 – Análise das propostas de Metamodelos para MER

3.2 Ferramentas CASE para MER

Considerando que existem muitas ferramentas CASE para o projeto de BD e que a maioria dessas ferramentas tem notação mais voltada para o projeto lógico do que conceitual, nesta seção, são discutidas as seguintes ferramentas CASE mais alinhadas a proposta original do MER: *brModelo* (seção 3.2.1), *DBMain* (seção 3.2.2) e *SmartDraw* (seção 3.2.3).

3.2.1 *brModelo*

brModelo [3], versão 2.0, é uma ferramenta CASE *freeware* que roda na plataforma Windows e permite desenhar esquemas conceituais de BD segundo a notação de Chen [4] com as poucas adaptações introduzidas por Heuser [16], onde a principal diferença é a representação de relacionamento identificador/entidade fraca por uma linha mais grossa. Esta ferramenta consiste em um editor de esquemas, implementado em Delphi [3], que permite a definição dos esquemas através da manipulação direta de elementos na tela. Na Figura 3.3 mostra-se um esquema hipotético (sem significado) que visa verificar se a ferramenta *brModelo*, consegue cobrir todos os conceitos apresentados na seção 2.1. Concluída a verificação, constatou-se que a ferramenta *brModelo* não dá suporte à modelagem de: 1) atributo discriminador, 2) herança sobreposta (*overlap*) e 3) categoria, e por isso, estes conceitos não aparecem na Figura 3.3.

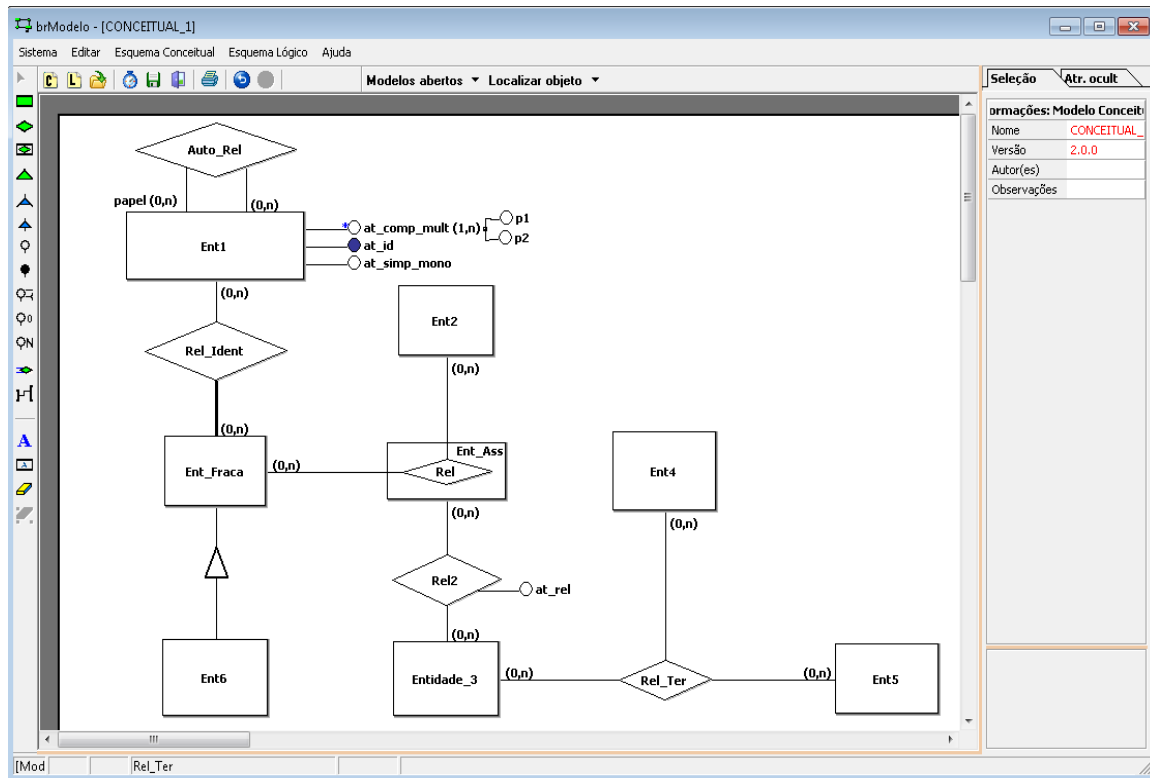


Figura 3.3 – Avaliação da ferramenta brModelo

Com relação à avaliação dos esquemas conceituais modelados na ferramenta, alguns erros sintáticos não são tratados pelo brModelo (ver Figura 3.4) e estes são: 1) Especialização sem subclasses; 2) Relacionamentos sem ou com links de relacionamento incompletos; 3) Entidades com nomes iguais no esquema; 4) Entidades e Relacionamentos sem nomes; e, 5) Atributos com nomes iguais na mesma Entidade.

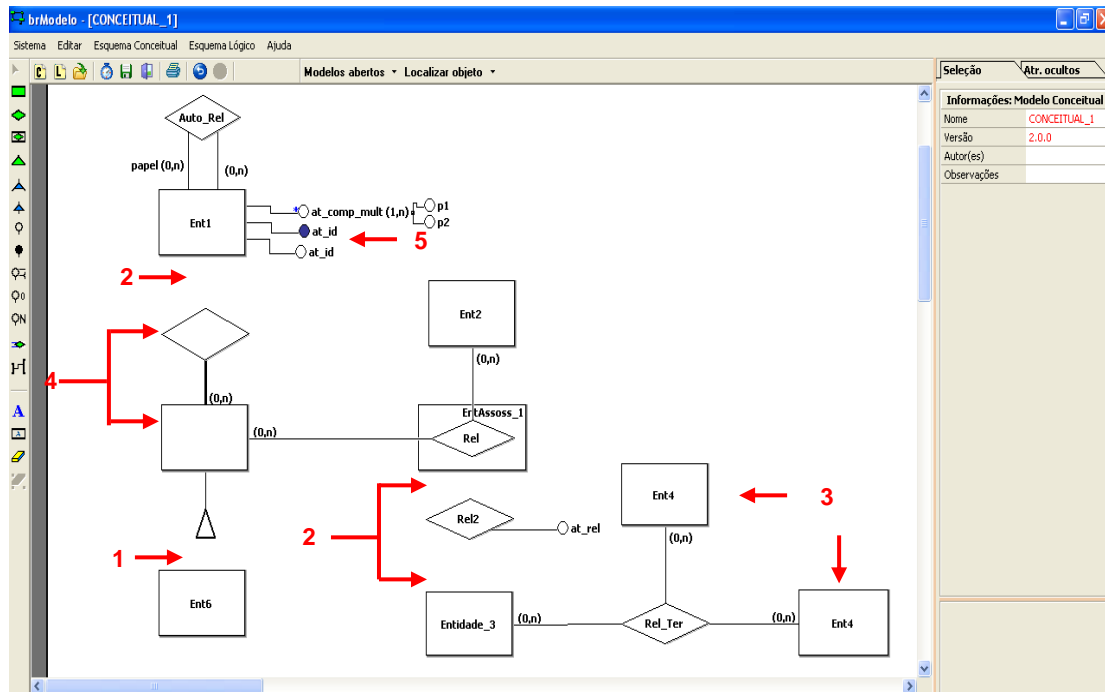


Figura 3.4 – Ocorrência de Erros Sintáticos na ferramenta brModelo

Os esquemas conceituais modelados no brModelo são exportados em formato XML [35], o qual provê um formato XML padrão para troca de metadados. Na Figura 3.5 mostra-se um fragmento do esquema da Figura 3.3 em XML. Ressalta-se que não foi encontrado nenhum metamodelo que descreva o XML de exportação.

```
- <Entidade nome="Ent1" id="1">
  <Left Valor="127" />
  <Top Valor="236" />
  <Width Valor="102" />
  <Height Valor="66" />
  <Fonte default="-1" />
- <Atributos>
  + <Atributo nome="at_comp_mult" id="2">
  + <Atributo nome="at_id" id="11">
  + <Atributo nome="at_simp_mono" id="14">
  </Atributos>
  <AtributosOcultos />
  <Dicionario />
  <Nula Valor="0" />
  <Observacao />
  <Futuro />
  <Anexos />
- <AutoRelacoes AutoRelacionado="-1">
  + <AutoRelacao nome="Auto_Rel" id="19">
  </AutoRelacoes>
  <Especializacoes ehEsp="0" />
</Entidade>
```

Figura 3.5 – Fragmento em XML do brModelo

3.2.2 *DBMain*

DBMain [6], versão 9.1.2, é uma ferramenta CASE comercial que roda nas plataformas Linux e Windows e permite desenhar esquemas conceituais de BD segundo uma notação própria. Esta notação pode ser vista como uma adaptação da notação Chen, onde as principais diferenças são: relacionamentos são modelados como hexágonos e atributos são modelados dentro de suas respectivas entidades ou relacionamentos. Ressalta-se que esta ferramenta faz a verificação da consistência sintática dos elementos de um esquema. Contudo, os esquemas modelados a partir do DBMain são armazenados em um formato proprietário da ferramenta, impossibilitando a troca de informações com outras ferramentas CASE para modelagem conceitual de BD. Além disso, não foi encontrado nenhum metamodelo que descreva o formato proprietário em questão. De forma a verificar se a ferramenta DBMain dá suporte aos conceitos discutidos na seção 2.1, na Figura 3.6 mostra-se, usando a notação do DBMain, o mesmo esquema artificial ilustrado na seção anterior. Concluída a verificação, constatou-se que a ferramenta DBMain não dá suporte à modelagem de: 1) atributo composto, 2) atributo derivado, 3) atributo discriminador, 4) atributo identificador, 5) entidade/relacionamento identificador, 6) entidade associativa; 7) participação, 8) herança disjunta ou sobreposta e 9) categoria, e por isso, estes conceitos não aparecem na Figura 3.6.

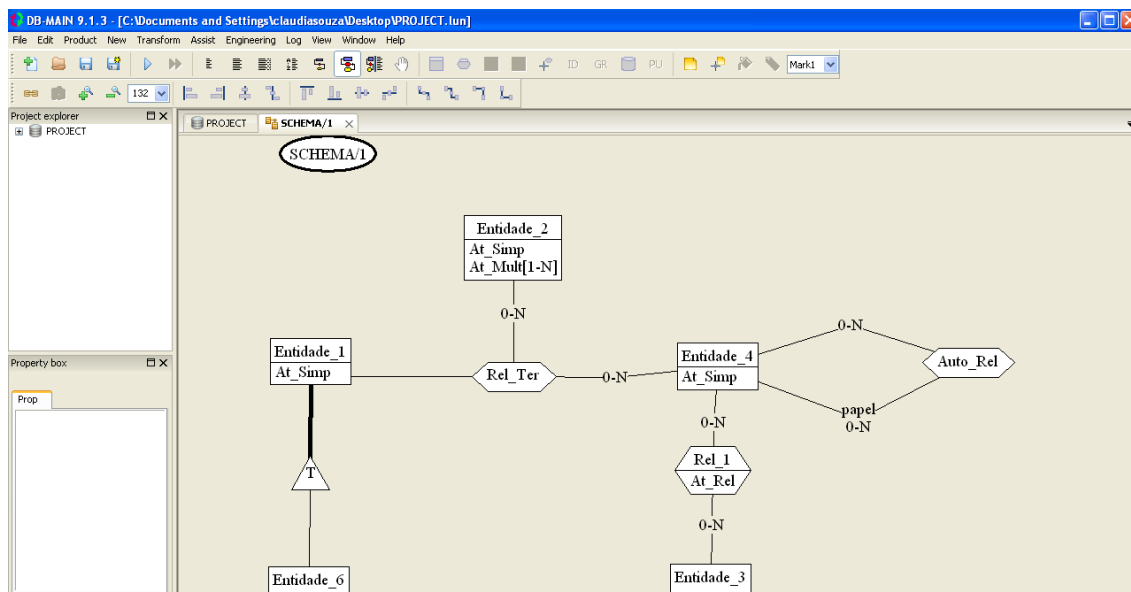


Figura 3.6 - Avaliação da Ferramenta DBMain

3.2.3 SmartDraw

SmartDraw [30], versão 3.0, é uma ferramenta CASE comercial que roda na plataforma Windows e permite desenhar esquemas conceituais de BD utilizando as seguintes notações: Chen [4], Elsmari&Navathe [8], Engenharia de Informação [22] e IDEF1X [19]. Assim como as demais ferramentas, o *SmartDraw* permite a definição de esquemas através da manipulação direta de elementos na tela. Ressalta-se que 1) os esquemas modelados a partir do *SmartDraw* são armazenados em um formato proprietário da ferramenta, impossibilitando a troca de informações com outras ferramentas CASE para modelagem conceitual de BD e 2) não foi encontrado nenhum metamodelo que descreva o formato proprietário em questão. De forma a verificar se a ferramenta *SmartDraw* dá suporte aos conceitos apresentados na seção 2.1, na Figura 3.7 mostra-se, usando a notação de Elsmari&Navathe, o mesmo esquema artificial ilustrado nas seções anteriores. Concluída a verificação, constatou-se que a ferramenta *SmartDraw* não dá suporte à modelagem de: 1) atributo identificador, 2) atributo derivado, 3) atributo discriminador, 4) participação, 5) herança disjunta ou sobreposta, 6) entidade associativa e 7) categoria, e por isso, estes conceitos não aparecem na Figura 3.7.

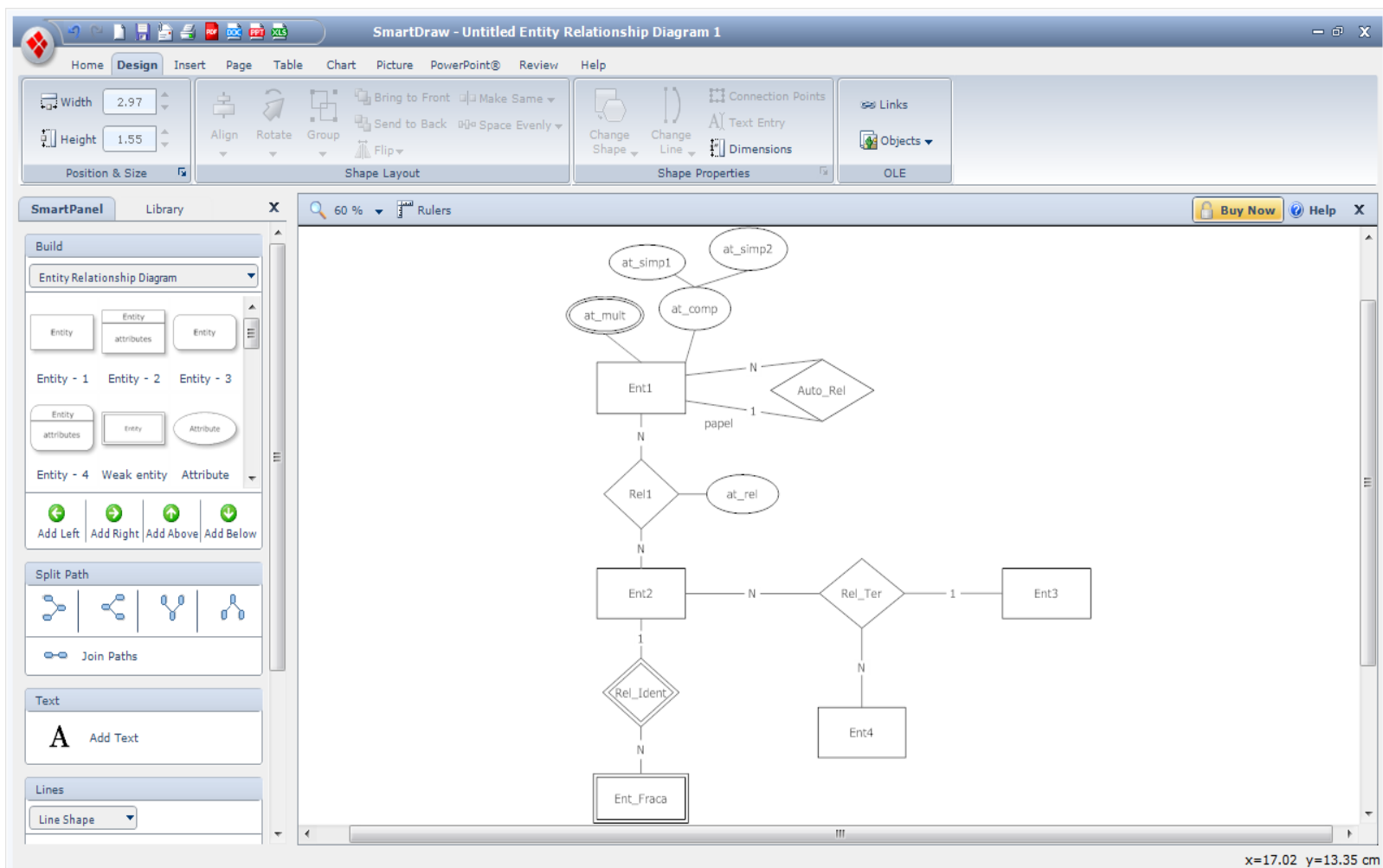


Figura 3.7 – Avaliação da Ferramenta SmartDraw

Com relação à avaliação dos esquemas conceituais modelados na ferramenta, observou-se que a ferramenta em questão não possui função para a avaliação sintática do esquema modelado. A Figura 3.8, mostra os erros sintáticos que a *SmartDraw* deixou de validar: 1) Associação entre Relacionamentos; 2) Associação entre Entidades; 3) Ciclicidade em Atributos; 4) Entidade e Relacionamento sem nomes; 5) Atributo sem Link de Conexão para Entidade; 6) Relacionamento com Links de Conexão inconsistente; 7) Associação de Atributos com Entidades e Relacionamentos diferentes; 8) Entidades com nomes iguais no mesmo esquema; e, 9) Atributos com nomes iguais na mesma Entidade.

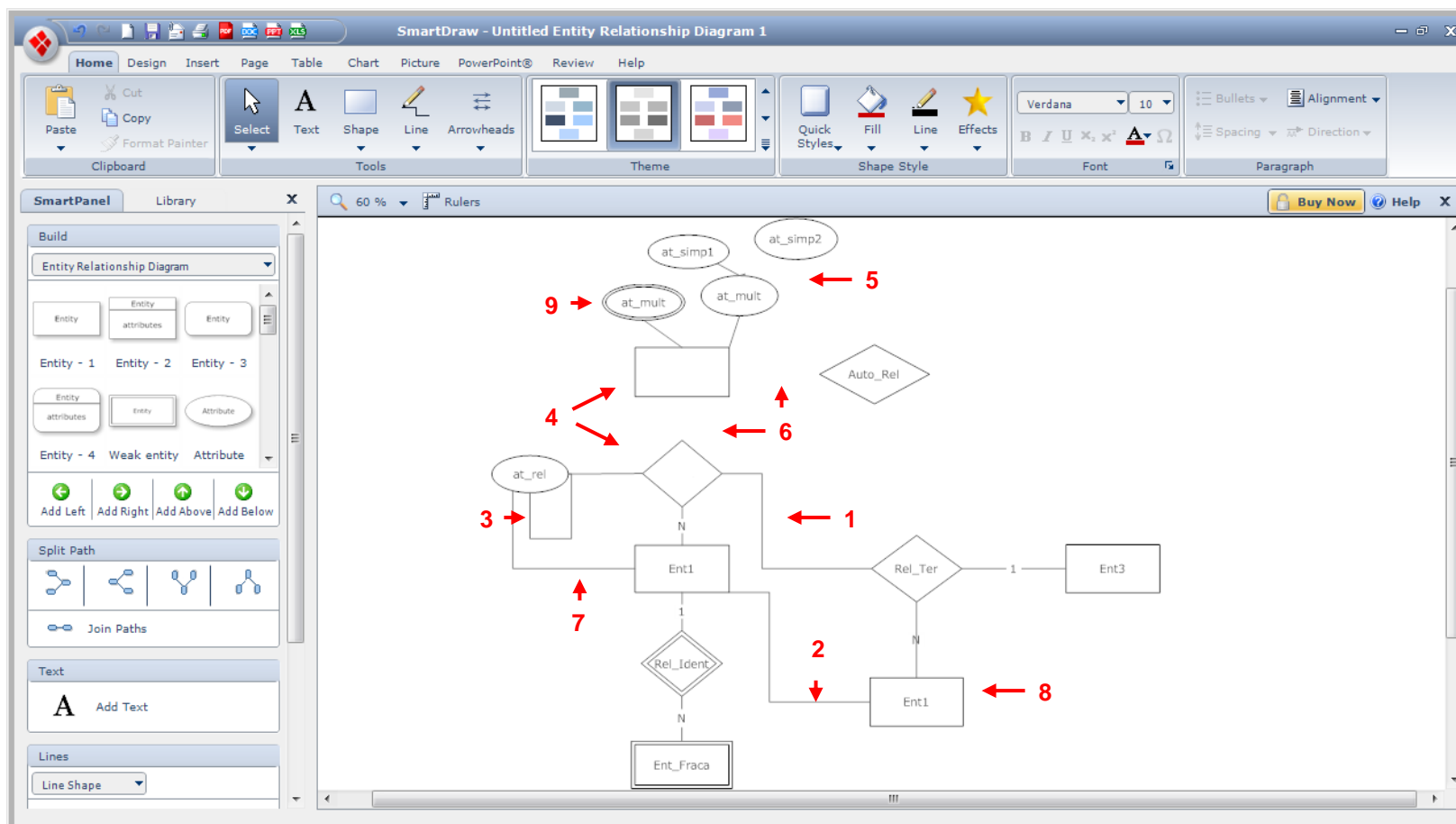


Figura 3.8 – Ocorrências de Erros Sintáticos na ferramenta SmartDraw.

3.2.4 Comparação das Ferramentas Correlatas

Com base nos conceitos discutidos no Capítulo 2, o Quadro 3.2 mostra a avaliação das ferramentas CASE para MER abordadas. Neste quadro, os marcadores “+” e “-” denotam, respectivamente, se a ferramenta oferece suporte ou não oferece suporte.

Itens	brModelo	DBMain	SmartDraw
Entidade Regular	+	+	+
Entidade Associativa	+	-	-
Entidade/Relacionamento Fraco	+	-	+
Atributo Simples	+	+	+
Atributo Composto	+	-	+
Atributo Monovalorado	+	+	+
Atributo Multivalorado	+	+	+
Atributo Derivado	-	-	-
Atributo Identificador	+	-	-
Atributo Discriminador	-	-	-
Auto-Relacionamento	+	+	+
Relacionamento Binário	+	+	+
Relacionamento <i>N-ário</i>	+	+	+
Participação do Relacionamento	+	+	+
Cardinalidade do Relacionamento	+	+	+
Papéis	+	+	+
Herança Simples	+	+	-
Herança Múltipla	-	-	-
Herança Total ou Parcial	+	+	-
Herança Disjunta ou Sobreposta	-	-	-
Categoria	-	-	-
Impede Erro Sintático	-	+	-
Formato Padrão para Troca de Metadados	+	-	-

Quadro 3.2 – Análise das propostas de ferramentas CASE para MER

3.3 *Considerações Finais*

Este capítulo abordou os principais trabalhos relacionados a esta dissertação. A análise dos metamodelos da Seção 3.1 mostra que estes têm pouca ênfase na especificação dos conceitos relacionados ao MER, os quais são essenciais para modelar esquemas conceituais. Além disso, estes metamodelos ainda apresentam características que estão fora do escopo da fase de projeto conceitual (e.g a definição de tipos de dados e chaves estrangeiras) e não apresentam restrições para a avaliação sintática dos seus esquemas conceituais. Já as ferramentas CASE discutidas na Seção 3.2 não implementam em sua totalidade os construtores do MER discutidos no capítulo 2 e algumas delas não levam em consideração padrões (i.e., XMI) para troca de metadados (*DBMain* e *SmartDraw*) ou não impedem a ocorrência de erros sintáticos que comprometem a consistência dos esquemas modelados (*brModelo* e *SmartDraw*). Ressalta-se não foi encontrado nenhum metamodelo para as ferramentas avaliadas, tornando suas implementações obscuras, trabalhosas e altamente dependentes de programação. Em resumo, o estudo desses trabalhos ratifica a importância de um metamodelo e de uma ferramenta CASE para modelagem conceitual de BD segundo o MER. O próximo capítulo apresenta o metamodelo e a ferramenta CASE propostos nesta dissertação.

4 O Metamodelo ERMM e a Ferramenta ERCASE

Ao longo dos capítulos discutidos, a construção de um esquema conceitual segundo o MER mostrou-se um fator importante por descrever de forma concisa e em alto nível os requisitos de dados do usuário. A especificação de metamodelos e ferramentas CASE para o MER ainda é um problema em aberto, especialmente pelo fato de haver poucas ferramentas CASE para modelagem verdadeiramente conceitual de BD e nenhum metamodelo do MER que seja aceito como consenso ou padrão (ver Capítulo 3). Logo, a definição de um metamodelo e o desenvolvimento de uma ferramenta CASE para este propósito são iniciativas importantes. Neste sentido, este capítulo propõe o metamodelo ERMM (*Entity-Relationship MetaModel*) e a ferramenta ERCASE (*Entity-Relationship CASE*) para subsidiar o projeto conceitual de BD baseado nos conceitos e notações preconizados por Navathe [EN05].

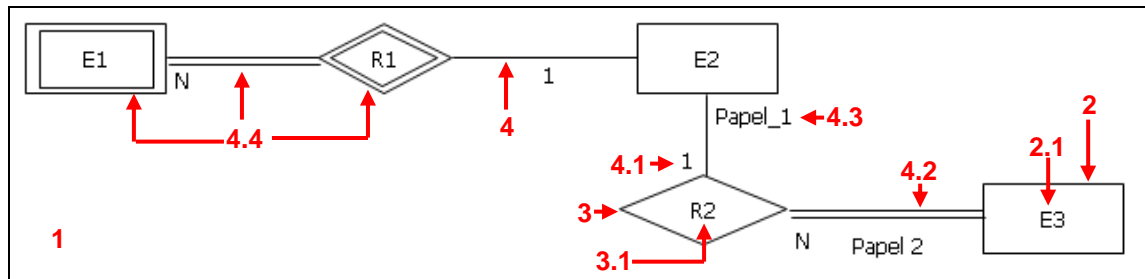
Neste sentido, a Seção 4.1 apresenta o metamodelo ERMM, seu diagrama de classes e restrições. Em seguida, a Seção 4.2 apresenta a ferramenta ERCASE, sua arquitetura e funcionalidades. Na seção 4.3, são discutidas as características do ERMM e ERCASE de acordo com os critérios de avaliação definidos (ver seções 3.1.3 e 3.2.4). Por fim, na Seção 4.4, são apresentadas algumas considerações finais sobre este capítulo.

4.1 O Metamodelo ERMM

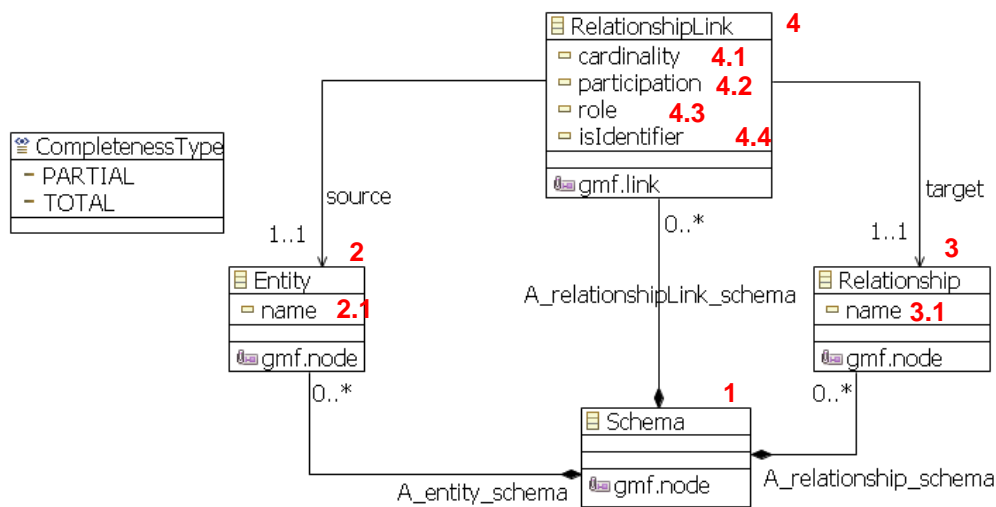
O ERMM é um metamodelo que: 1) é especificado utilizando o metamodelo Ecore (ver Capítulo 2); 2) serve como metamodelo base para ferramentas CASE de projeto conceitual segundo o MER; 3) possibilita a verificação da consistência dos esquemas desenvolvidos; e, finalmente, 4) engloba os construtores básicos do MER e suas extensões (ver Capítulo 2) baseado nos conceitos e notação de Elsmari&Navathe. A seguir é apresentado, em fragmentos, o metamodelo ERMM (no apêndice A encontra-se o ERMM por inteiro). Para facilitar a explicação de cada fragmento apresentado, será

utilizado um esquema conceitual meramente didático/ilustrativo e um conjunto de anotações numéricas que visam mostrar a correspondência entre os objetos do esquema conceitual e as metaclasses do ERMM.

Na Figura 4.1 mostra-se a relação existente entre os objetos Esquema, Entidade e Relacionamento da notação do MER e as metaclasses Esquema (*Schema*), Entidade (*Entity*), Relacionamento (*Relationship*), Link de Relacionamento (*RelationshipLink*) e Tipo de Completude (*CompletenessType*) do ERMM. Na Figura 4.1(b) a metaclasses Esquema é o elemento raiz do metamodelo proposto, a qual representa a composição de todos os elementos do ERMM. Nesta Figura, um Esquema é composto por um conjunto nomeado de zero ou mais Entidades de um Esquema (*A_Entity_Schema*), Relacionamentos de um Esquema (*A_Relationship_Schema*) e Links de Relacionamento de um Esquema (*A_RelationshipLink_Schema*). As metaclasses Entidade e Relacionamento correspondem aos objetos de mesmo nome no MER. A metaclasses Link de Relacionamento tem uma associação Fonte (*Source*) para a metaclasses Entidade e uma associação Alvo (*Target*) para a metaclasses Relacionamento. Assim, todos os objetos Link de Relacionamento sempre irão partir de um objeto Entidade e chegar a um objeto Relacionamento. Além disso, a metaclasses Link de Relacionamento tem as seguintes propriedades: cardinalidade (*cardinality*), papel (*role*), relacionamento identificador/entidade fraca (*isIdentifier*) e participação (*participation*), as quais, respectivamente, registram a quantidade máxima de ocorrências, um possível comportamento especial, a possível dependência de identificador e a participação de um Link de Relacionamento. Para a propriedade participação, tem-se a metaclasses Tipo de Completude que é uma enumeração dos seus valores válidos (i.e. TOTAL (*TOTAL*) ou PARCIAL (*PARTIAL*)).



(a) MER – Esquema, Entidade e Relacionamento.

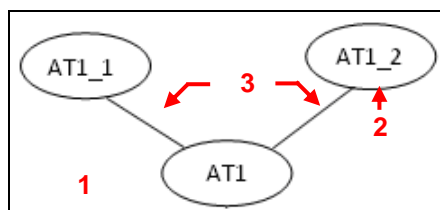


(b) ERMM - Schema, Entity, Relationship, RelationshipLink e CompletenessType

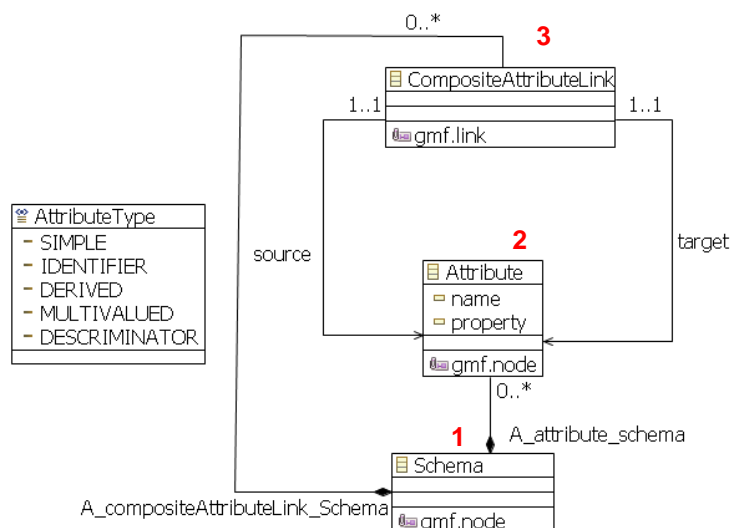
Figura 4.1. MER X ERMM: Esquema, Entidade e Relacionamento.

Na Figura 4.2 mostra-se a relação entre os objetos Entidade, Relacionamento e Atributo da notação do MER e as metaclasses Entidade, Relacionamento, Atributo (*Attribute*), Link de Atributo-Entidade (*AttributeEntityLink*), Link de Atributo-Relacionamento (*AttributeRelationshipLink*) e Tipo de Atributo (*AttributeType*) do ERMM. Conforme mencionado, a metaclasses Esquema é o elemento principal do metamodelo, pois contém todos os elementos de ERMM, esta metaclasses é composta por um conjunto de zero ou mais Atributos de um Esquema (*A_attribute_schema*), Links de Atributo-Entidade (*A_attributeEntityLink_schema*) e Links de Atributo-Relacionamento (*A_attributeRelationshipLink*). A metaclasses Atributo corresponde ao objeto de mesmo nome no MER. Na Figura 4.2(b) as metaclasses Entidade e Relacionamento podem estar associadas à metaclasses Atributo, por meio das

A Figura 4.3 mostra a relação entre objetos Atributo da notação do MER e as metaclasses Atributo e Link de Atributo-Composto (*CompositeAttributeLink*) do ERMM. A metaclasses Esquema é composta por um conjunto de zero ou mais Links de Atributo-Composto de um Esquema (*A_compositeAttributeLink_schema*). Na Figura 4.3(b) a metaclasses Atributo pode estar associada a outros objetos Atributo através da metaclasses Link de Atributo-Composto que, por sua vez, tem uma associação Fonte e Alvo para a metaclasses Atributo. A associação entre estas metaclasses corresponde a objetos atributo do tipo composto no MER.



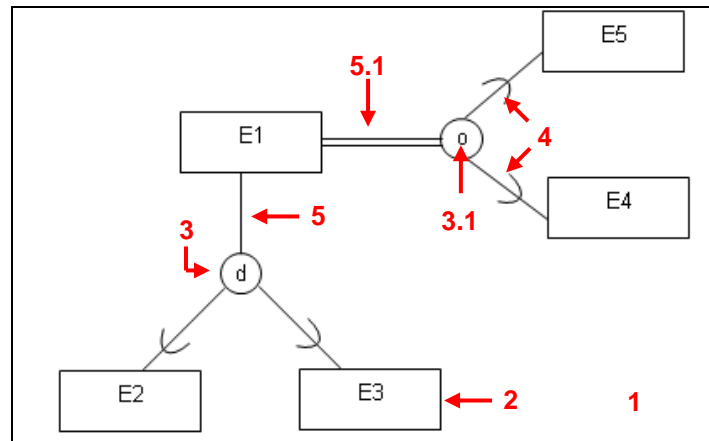
(a) MER – Atributo Composto



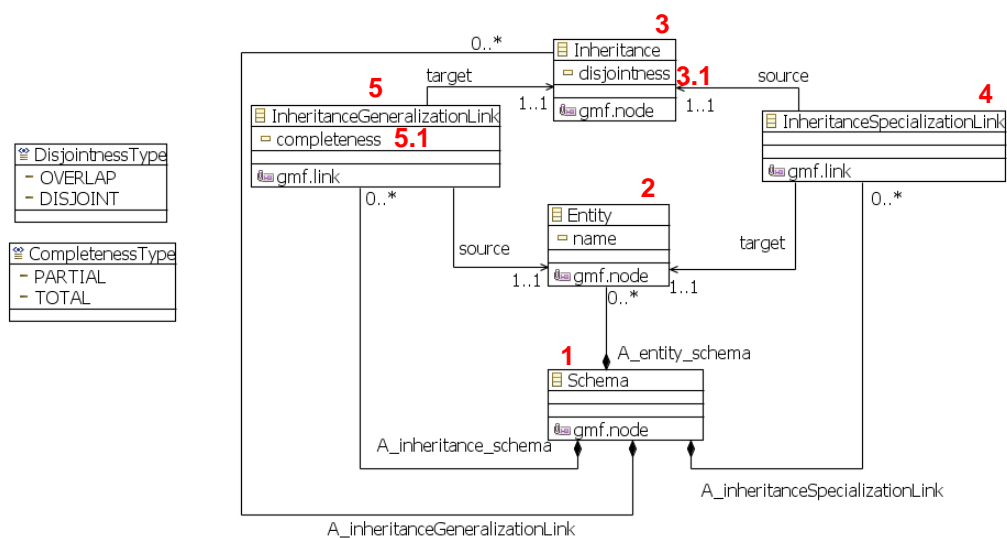
(b) ERMM - Schema, Attribute, CompositeAttributeLink e AttributeType.

Figura 4.3. MER X ERMM: Atributo.

Na Figura 4.4 mostra-se a relação existente entre os objetos Entidade e Herança da notação do MER e as metaclasses Entidade, Herança (*Inheritance*), Link de Herança-Generalização (*InheritanceGeneralizationLink*), Link de Herança-Especialização (*InheritanceSpecializationLink*) Tipo de Completude e Tipo de Disjunção (*DisjointnessType*) do ERMM. Na Figura 4.4(b), um Esquema é composto por um conjunto nomeado de zero ou mais Entidades de um Esquema (A_Entity_Schema), Herança de um Esquema (A_Inheritance_Schema), Links de Herança-Generalização de um Esquema (A_InheritanceGeneralizationLink_Schema) e Links de Herança-Especialização de um Esquema (A_InheritanceSpecializationLink_Schema). A metaclasses Link de Herança-Generalização tem uma associação Fonte para a metaclasses Entidade e uma associação Alvo para a metaclasses Herança. Assim, todos os objetos Link de Herança-Generalização sempre irão partir de um objeto Entidade e chegar a um objeto Herança. Por sua vez, a metaclasses Link de Herança-Especialização tem uma associação Fonte para a metaclasses Herança e uma associação Alvo para a metaclasses Entidade. Assim, todos os objetos Link de Herança-Especialização sempre irão partir de um objeto Herança e chegar a um objeto Entidade. Conforme a Figura 4.4 (b) observa-se que a metaclasses Herança possui a propriedade disjunção (*disjointness*), que define as restrições DISJUNÇÃO (*DISJOINT*) e SOBREPOSIÇÃO (*OVERLAP*), estas restrições correspondem aos valores válidos da metaclasses enumerada Tipo de Disjunção. Além da propriedade disjunção, tem-se a propriedade completude para os objetos Link de Herança-Generalização. Esta propriedade tem seu valores definidos na metaclasses enumerada Tipo de Completude, a qual foi explicada na Figura 4.1.



(a) MER – Entidade e Herança (Generalização/Especialização)

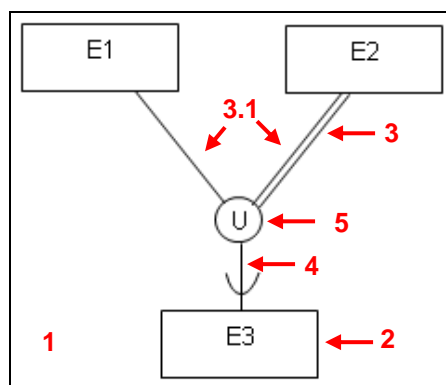


(b) ERMM - Schema, Entidade, Inheritance, InheritanceSpecializationLink, InheritanceGeneralizationLink, CompletenessType e DisjointnessType.

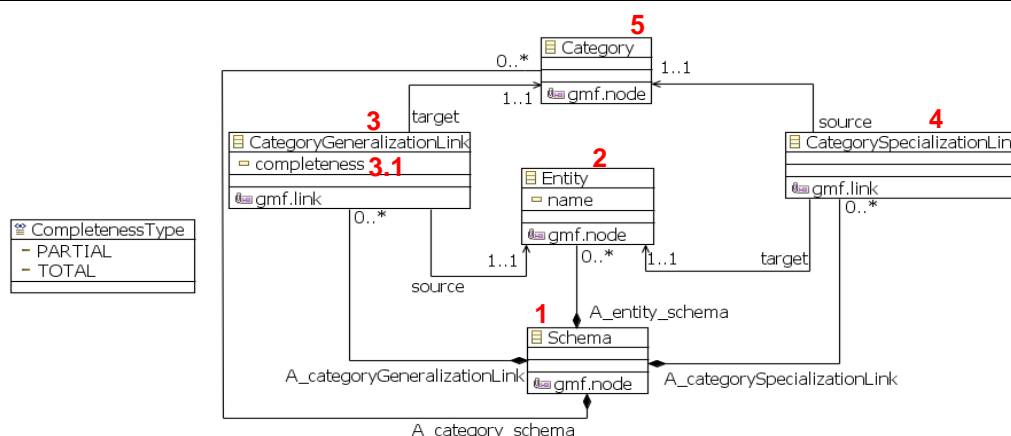
Figura 4.4. MER X ERMM: Entidade e Herança (Generalização/Especialização).

A Figura 4.5 mostra a relação existente entre os objetos Entidade e Categoria da notação do MER e as metaclasses Entidade, Categoria (*Category*), Link de Categoria-Generalização (*CategoryGeneralizationLink*), Link de Categoria-Especialização (*CategorySpecializationLink*) e Tipo de Completude do ERMM. Na Figura 4.5(b), um Esquema é composto por um conjunto nomeado de zero ou mais Entidades de um Esquema (*A_Entity_Schema*), Categoria de um Esquema (*A_Category_Schema*), Links de Categoria-Generalização de um

Esquema (*A_CategoryGeneralizationLink_Schema*) e Links de Categoria-Especialização de um Esquema (*A_CategorySpecializationLink_Schema*). Seguindo o mesmo raciocínio aplicado na generalização e especialização de uma herança, a metaclasses Link de Categoria-Generalização tem uma associação Fonte para a metaclasses Entidade e uma associação Alvo para a metaclasses Categoria. Desta forma, os objetos Link de Categoria-Generalização irão partir de um objeto Entidade e chegar a um objeto Categoria. Por sua vez, a metaclasses Link de Categoria-Especialização tem uma associação Fonte para Categoria e uma associação Alvo para Entidade. Desta forma, os objetos Link de Categoria-Especialização irão partir de um objeto Categoria e chegar a um objeto Entidade. A metaclasses Link de Categoria-Generalização possui a propriedade completude, a qual é definida pela metaclasses enumerada Tipo de Completude.



(a) MER – Entidade e Categoria



(b) ERMM - Schema, Entidade, Category, CategorySpecializationLink, CategoryGeneralizationLink, CompletenessType.

Figura 4.5. MER X ERMM – Entidade e Categoria.

4.1.1 Especificação das Restrições ERMM

Para impedir a ocorrência de erros sintáticos nos esquemas conceituais modelados a partir do ERMM, são definidas algumas restrições em OCL (*Object Language Constraint*) [OCL02] e em linguagem Java com o uso do *framework* GMF [GMF10]. As próximas subseções especificam as restrições definidas para o ERMM.

4.1.1.1 Associar Relacionamentos a Relacionamentos

- **Descrição:** objetos Relacionamento não podem estar associados entre si.
- **Restrição:** O ERMM garante intrinsecamente esta restrição. Isto é, não necessário definir nenhuma restrição em OCL ou JAVA para impor esta restrição no ERMM, pois suas metaclasses já garantem esta restrição. Neste sentido, como pode ser visto na Figura 4.6, este erro sintático é evitado por meio da metaclasses Link de Relacionamento, a qual necessariamente tem apenas uma associação Fonte para a metaclasses Entidade e uma associação Alvo para a metaclasses Relacionamento. Ou seja, o ERMM impede a associação entre objetos Relacionamento.

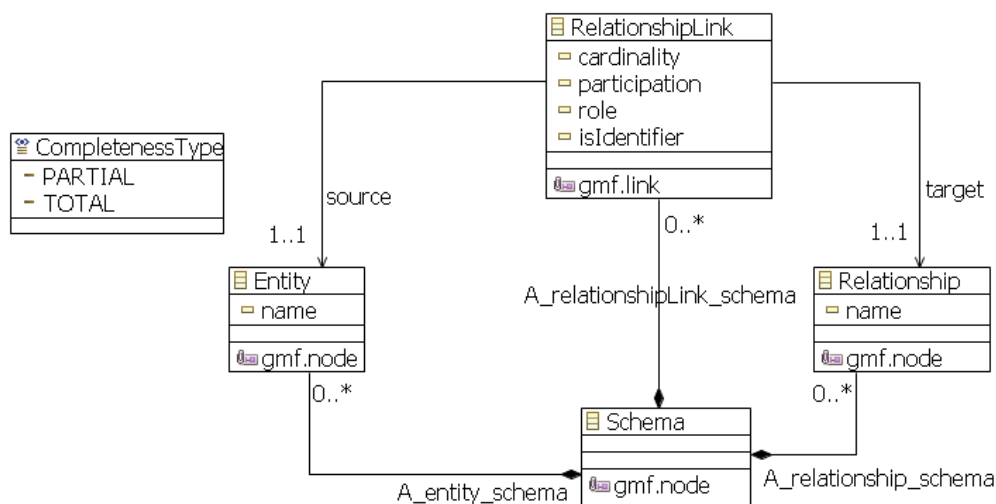


Figura 4.6 - Restrição imposta pela metaclasses *Link de Relacionamento*

4.1.1.2 Associar Entidades a Entidades

- **Descrição:** objetos Entidade não podem estar associados entre si.
- **Restrição:** Pelos mesmos motivos da restrição anterior, esta restrição também é garantida intrinsecamente pelo ERMM por meio da metaclasses

Link de Relacionamento. Ou seja, como pode ser visto na Figura 4.6, este erro sintático é evitado por meio da metaclassse Link de Relacionamento, a qual necessariamente tem apenas uma associação Fonte para a metaclassse Entidade e uma associação Alvo para a metaclassse Relacionamento. Desta forma, o ERMM impede a associação entre objetos Entidade.

4.1.1.3 *Ciclicidade entre Atributos*

- **Descrição:** objetos Atributo não podem estar autorelacionados.
- **Restrição:** Diferente das restrições anteriores, esta é implementada em JAVA e visa impedir que objetos Atributo tenham associações Origem e Alvos iguais. O método *canExistCompositeAttributeLink()* implementa esta restrição. Neste método, será verificado se um objeto Atributo possui referência *source* e *target* diferentes. Caso sim, o método retorna verdadeiro (true) e permite a associação entre Atributos.

```
canExistCompositeAttributeLink ( Schema container, Attribute source, Attribute target) {
    return attributeValidate(container, source, target);
    int count = 0;
    for(CompositeAttributeLink obj :container.getA_compositeAttributeLink_Schema()){
        if(obj.getTarget().equals(target) && ++count == 2){
            return false; }
        }
    return true;
}
```

4.1.1.4 *Entidades com nomes iguais*

- **Descrição:** Objetos Entidade não devem possuir nomes iguais em um esquema.
- **Restrição:** Esta restrição é implementada em JAVA e visa impedir que objetos Entidades tenham o mesmo nome no esquema. O método *setName()* implementa esta restrição. Este método verifica, através de um evento, as ocorrências de alterações no nome de um objeto Entidade. Caso o nome seja diferente dos outros objetos Entidades contidos no esquema a alteração é realizada. Caso contrário, o objeto Entidade permanece com o nome antigo.

```

setName(String newName) {
    if(schema != null){
        for(Entity e : schema.getA_entity_schema()){
            if(newName.equals(e.getName())) {
                return; }
        }
    }
    String oldName = name;
    name = newName;
    if (eNotificationRequired())
        eNotify(new ENotificationImpl(this, Notification.SET, ERMMPackage.ENTITY_NAME, oldName,
        name));
}

```

4.1.1.5 Atributos com nomes iguais

- **Descrição:** Objetos Atributos não podem ter nomes iguais em objetos Entidade ou objetos Relacionamento em um esquema.
- **Restrição:** Esta restrição é implementada em JAVA e visa impedir que objetos Atributos tenham o mesmo nome em objetos Entidade ou objetos Relacionamento. O método `setName()` implementa esta restrição. O comportamento deste método é igual ao descrito na restrição 4.1.1.4.

```

setName(String newName) {
    if(nameExistsInEntity(newName) || nameExistsInRelationship(newName)){
        return;
    }
    String oldName = name;
    name = newName;
    if (eNotificationRequired())
        eNotify(new ENotificationImpl(this, Notification.SET, ERMMPackage.ATTRIBUTE__NAME, oldName,
        name));
}

```

4.1.1.6 Relacionamentos sem associações para Entidades

- **Descrição:** Objetos Relacionamento devem ter no mínimo dois links de relacionamento para objetos Entidade.
- **Restrição:** Diferente das restrições anteriores esta é implementada em OCL e visa impedir que objetos Relacionamentos possuam Links de relacionamentos inconsistentes para objetos Entidades. O trecho OCL a seguir implementa esta restrição. Neste trecho é avaliado se objetos Relacionamento possuem no mínimo duas referências de objetos *Link* de

Relacionamento para objetos Entidades, caso contrário o erro sintático é notificado.

```
if (not Relationship.allInstances()->isEmpty() and RelationshipLink.allInstances()->isEmpty()) then
    false
else
    ((RelationshipLink.allInstances()-> select(r | r.target = self)->size())>1))
endif
```

4.1.1.7 Herança sem Link de Especialização

- **Descrição:** Objetos Herança devem ter links de especialização para indicar subclasses de objetos Entidades.
- **Restrição:** Esta restrição também é implementada em OCL e visa impedir que objetos Herança não possuam links de especialização para objetos Entidade. O trecho OCL a seguir implementa esta restrição. Neste trecho é avaliado se um objeto Herança possui no mínimo uma referência de objeto de Link de Especialização para um objeto Entidade, caso contrário o erro sintático é notificado.

```
if (not Inheritance.allInstances()->isEmpty() and SpecializationLink.allInstances()->isEmpty()) then
    false
else
    ((SpecializationLink.allInstances()-> select(r | r.source = self)->size())=1))
endif
```

4.1.1.8 Herança sem Link de Generalização

- **Descrição:** Objetos Herança devem ter links de generalização para indicar as superclasses de objetos Entidade.
- **Restrição:** Esta restrição é implementada em OCL e visa garantir que todos os objetos Herança possuam links Herança-Generalização para objetos Entidade. O trecho OCL a seguir implementa esta restrição. Neste trecho é avaliado se um objeto Herança possui a referência de um objeto Link de Herança-Categoria para um objeto Entidade, caso contrário o erro sintático é notificado.

```

if (not Inheritance.allInstances()->isEmpty() and InheritanceGeneralizationLink.allInstances()->isEmpty()
and SpecializationLink.allInstances()->isEmpty()) then
    false
else
    ((InheritanceGeneralizationLink.allInstances()->select(r | r.target = self)->size())=1)
endif

```

4.1.1.9 Categoria sem Link de Generalização

- **Descrição:** Objetos Categoria devem ter links de generalização para outros objetos Entidades.
- **Restrição:** Esta restrição é implementada em OCL e visa impedir que objetos Categoria não possuam links de categoria-generalização para objetos Entidade. O trecho OCL a seguir implementa esta restrição. Neste trecho é avaliado se um objeto Categoria possui a referência de um objeto Link de Categoria para um objeto Entidade, caso contrário o erro sintático é notificado.

```

if (not Category.allInstances()->isEmpty() and CategoryGeneralizationLink.allInstances()->isEmpty()) then
    false
else
    ((CategoryGeneralizationLink.allInstances()-> select(r | r.source = self)->size())=1)
endif

```

4.1.1.10 Categoria sem Link de Especialização

- **Descrição:** Objetos Categoria devem ter links de especialização para outros objetos Entidades. Esta restrição é implementada em OCL.
- **Restrição:** Esta restrição é implementada em OCL e visa impedir que objetos Categoria não possuam links de Categoria-Especialização para objetos Entidade. O trecho OCL a seguir implementa esta restrição. Neste trecho é avaliado se um objeto Categoria possua no mínimo uma referência de objeto Link de Categoria-Especialização para um objeto Entidade, caso contrário o erro sintático é notificado.

```

if (not Category.allInstances()->isEmpty() and CategorySpecializationLink.allInstances()->isEmpty()) then
    false
else
    ((CategorySpecializationLink.allInstances()->select(r | r.target = self)->size())=1)
endif

```


4.2 A Ferramenta ERCASE

ERCASE é uma ferramenta CASE para o projeto de esquemas conceituais de BD segundo o MER que: 1) oferece uma interface gráfica intuitiva e com recursos gráficos que permitem abstrair detalhes de implementação do esquema ERMM; 2) é implementada sobre os *frameworks open-source* EMF [SFP08] e GMF [GMF10] da plataforma Eclipse (ver Capítulo 2), garantindo assim sua portabilidade para diversos sistemas operacionais; 3) é baseada no padrão XMI (*XML Metadata Interchange*) [XMI21] para armazenamento, manipulação, recuperação e intercâmbio de metadados; e 4) faz a avaliação sintática do esquema conceitual modelado. Ressalta-se que a ERCASE limita-se a fase de projeto conceitual de BD. Isto é, esta ferramenta não gera código ou esquema para as fases de projeto lógico ou físico de BD. A seguir são descritos os componentes da arquitetura da ERCASE (Figura 4.7):

- **Editor Gráfico:** componente que implementa a interface gráfica de modelagem de ERCASE. Nesta interface, o usuário dispõe de uma área de edição, uma paleta de elementos que representa a notação dos construtores do MER e barras de ferramentas para formatação, validação, exportação e outras funcionalidades.

- **Validador de Esquema:** componente que implementa o sistema de avaliação sintática do esquema MER modelado em ERCASE. Este usa restrições OCL e restrições implementadas via código-fonte no GMF definidas em ERMM para verificar a consistência do esquema. Por exemplo, garantia de unicidade de nomes de entidades e atributos são verificados.

- **Gerenciador de Metadados:** componente que implementa o sistema de repositório de metadados de ERCASE. Este usa XMI para gerenciar os metadados do esquema do MER. O metamodelo ERMM guia este componente para permitir o armazenamento e a manipulação dos metadados dos esquemas MER salvos.

- **Esquemas Conceituais Salvos:** representa os esquemas conceituais de ERCASE que já foram salvos pelo usuário, sendo armazenados no formato XMI.

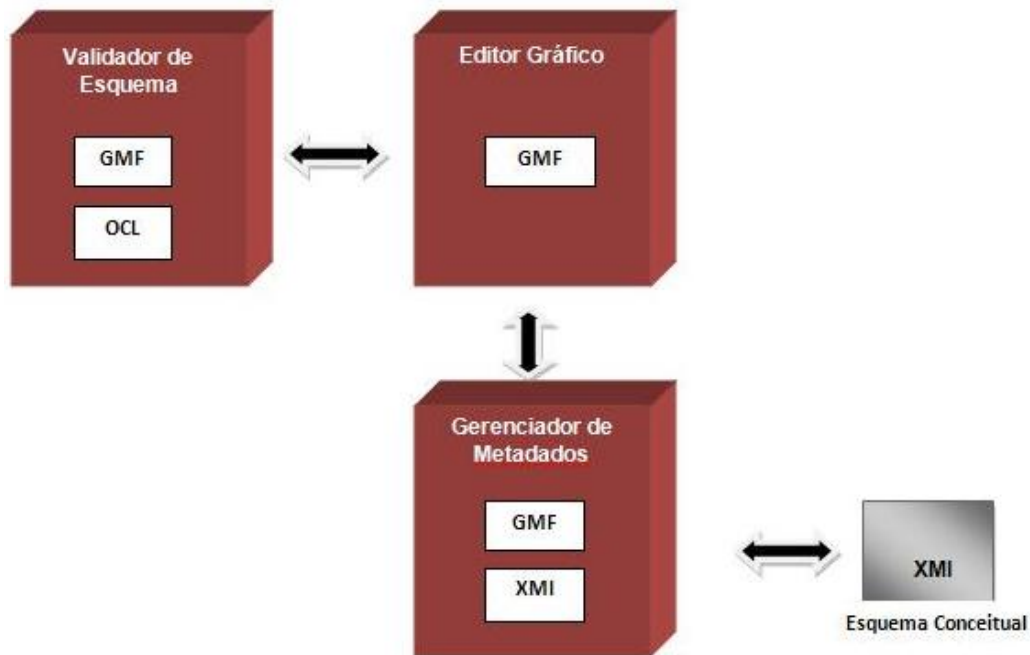


Figura 4.7 – Componentes da Arquitetura de ERCASE.

As seções a seguir apresentam detalhadamente os principais componentes da arquitetura ERCASE.

4.2.1 Componente Editor Gráfico

Neste componente os elementos do MER são arrastados da paleta para a área de edição. ERCASE permite editar (e.g., cores e fontes), desfazer/fazer; copiar/colar/recortar; visualizar sob diferentes níveis de zoom; organizar (e.g., auto-organizar e alinhar) e exportar o esquema como figura (e.g., JPEG, GIF, SVG). A Figura 4.8 apresenta as partes mais importantes do ambiente de modelagem de ERCASE disponibilizado pelo componente Editor Gráfico.

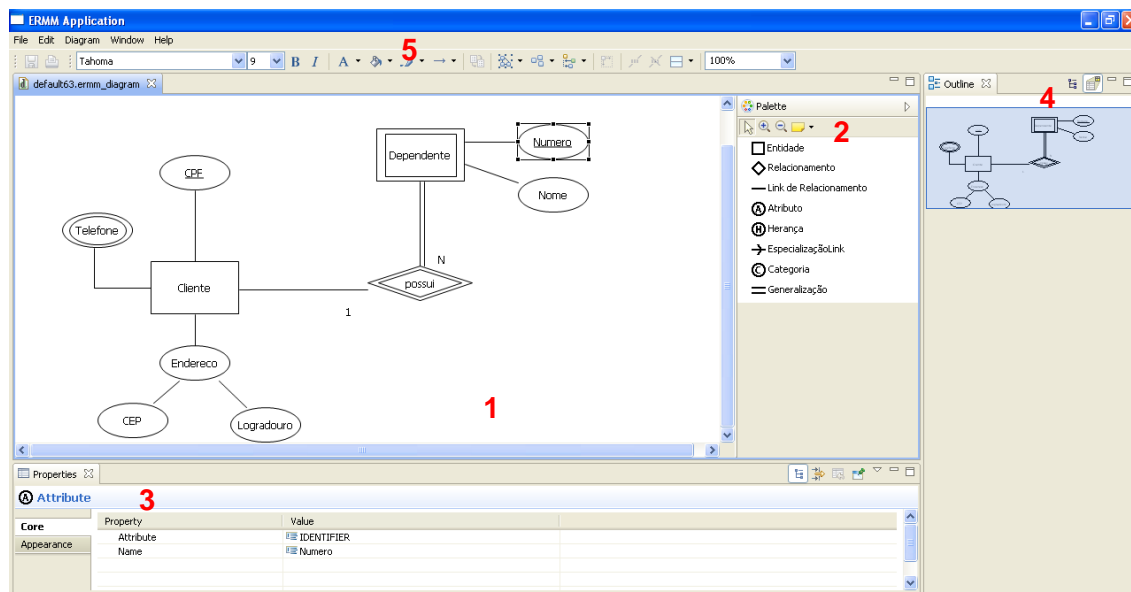


Figura 4.8 - Ambiente de Modelagem de ERCASE

A interface de ERCASE é dividida em 5 áreas principais: 1) Quadro de Edição, 2) Paleta de objetos, 3) aba Propriedades (*Properties*), 4) Área de Visualização (*Outline*) e 5) Barra de Ferramentas. A modelagem de um esquema em ERCASE basicamente consiste em clicar o mouse sobre o elemento do MER desejado na paleta e colocá-lo na área de edição do diagrama. Na Figura 4.9 mostra-se a paleta com as notações gráficas de ERCASE ampliada. Note que a semântica de cada objeto do MER pode ser facilmente percebida pelo seu pictograma e descrição. Além desses elementos, a paleta contempla o cursor, a lupa para zoom no esquema e um elemento para adição de notas textuais.

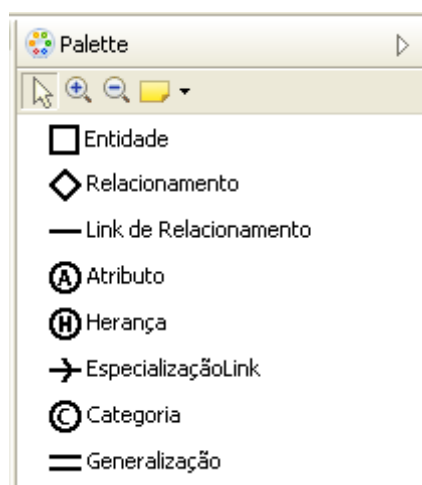


Figura 4.9 – Paleta de ERCASE

A edição das propriedades de cada uma das metaclasses do ERMM (i.e. objetos do MER) é feita selecionando o objeto e alterando o valor da propriedade desejada, conforme é exibido na área 3 da Figura 4.8. Essas propriedades podem ser relacionadas à aparência do elemento (e.g., estilo de fonte, cor da linha, preenchimento) ou de metadados do elemento (e.g., nome, tipo). A Figura 4.10 exemplifica a edição das propriedades nome e tipo do atributo.

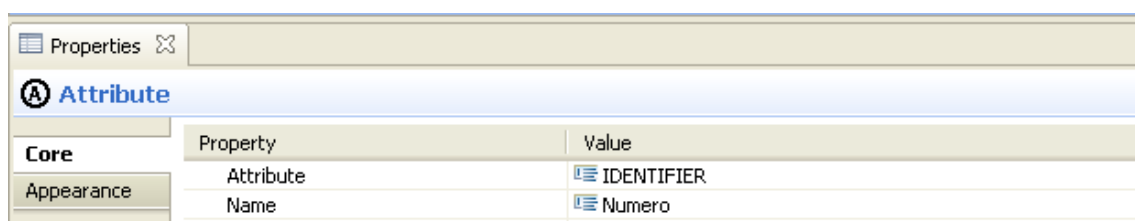


Figura 4.10 – Aba para a Edição de Propriedades

A interface possui ainda uma barra com botões para seleção de todos os elementos, organização automática e alinhamento de elementos (ver área 5 da Figura 4.8, ampliada na Figura 4.11). Além disso, na Área de Visualização tem-se uma visão geral do esquema e pode-se navegar pelo mesmo apenas clicando e arrastando o mouse, facilitando assim a visualização de esquemas grandes, conforme mostrado na Figura 4.8.

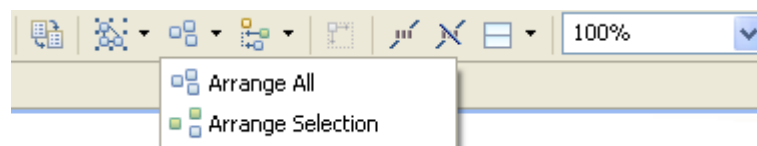


Figura 4.11 – Barra de Ferramentas para Seleção, Organização e Alinhamento.

4.2.2 Componente Validador de Esquemas

Esse componente permite que o projetista, a qualquer instante, valide a consistência do esquema que está sendo modelado. Isto é, verificar se o mesmo segue as regras de restrições sintáticas definidas pelo ERMM. Para construir este componente, algumas expressões OCL e métodos JAVA (ver seção 4.1.1) foram implementados com o uso do *framework* GMF. Desta forma,

o validador interage com o editor gráfico e faz a verificação de consistência da instância corrente do esquema, verificando se os seus elementos respeitam as regras sintáticas definidas no ERMM.

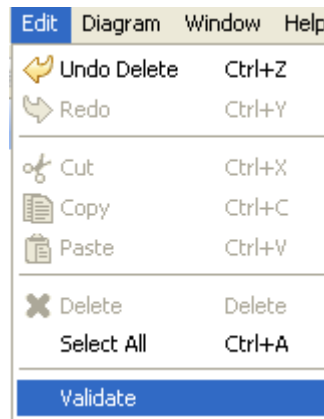


Figura 4.12 – Validação do Esquema.

A validação de um esquema pode ser feita acessando o *menu* Editar (*Edit*) e selecionando a opção Validar (*Validate*), conforme ilustrado na Figura 4.12. O componente de validação realiza a verificação do esquema e todas as inconsistências de modelagem são apresentadas com o símbolo de erro, como ilustrado na Figura 4.13, a qual indica um erro no Relacionamento “possui”, pois este só está associado à Entidade “Dependente”.

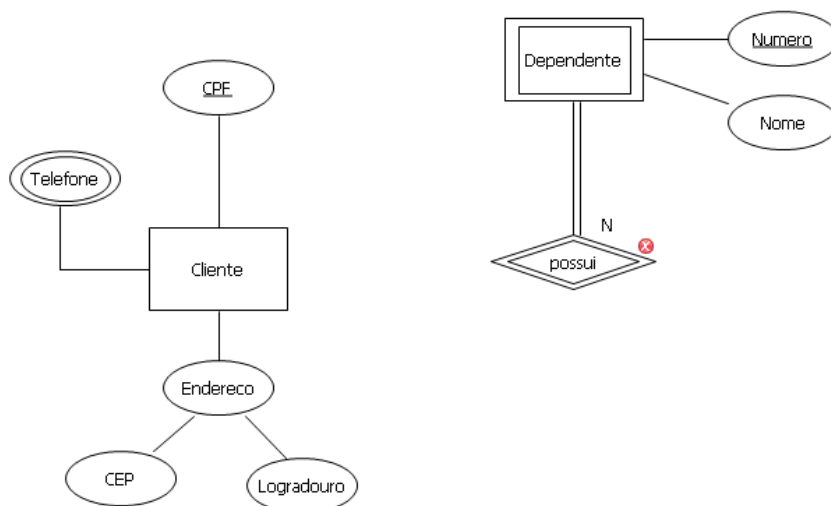


Figura 4.13 – Elemento sinalizado com Erro.

4.2.3 Componente Gerenciador de Metadados

Em ERCASE, cada esquema projetado é composto por dois arquivos: um arquivo contendo as representações gráficas do esquema (extensão “.ermm_diagram”) e um arquivo contendo os metadados de ERMM usados no esquema (extensão “.ermm”). O conteúdo dos arquivos de representações gráficas e metadados de ERMM são armazenados em um documento XML segundo o padrão XMI [XMI21]. A Figura 4.14 exemplifica a estrutura do arquivo XML que contém a especificação do objeto entidade Cliente mostrada na Figura 4.8. Na Figura 4.14, todo esquema (<ERMM:Schema>) é composto por entidades (<A_entity_schema>), relacionamentos (<A_relationship_schema>), atributos (<A_attribute_schema>), links de relacionamento (<A_relationshipLink_schema>) e links de entidade-atributo (<A_attributeEntityLink_schema>). O elemento atributo tem a tag interna “attribute” que armazena o tipo de atributo. Os elementos links sempre irão possuir as tags internas “source” e “target” para armazenar as referências das associações fonte e alvo. Além disso, o elemento link de relacionamento também possui as seguintes tags internas: “participation”, “cardinality”, “role” e “isIdentifier” que armazenam, respectivamente, a dependência do identificador, a quantidade máxima de ocorrências, o nome do papel e se relacionamento identificador/entidade fraca.

```
<?xml version="1.0" encoding="UTF-8"?>
<ERMM:Schema xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:ERMM="http://ERMM.ecore">
  <A_entity_schema name="Cliente"/>
  <A_entity_schema name="Dependente"/>
  <A_relationship_schema name="possui"/>
  <A_relationshipLink_schema source="//A_entity_schema.0" target="//A_relationship_schema.0"/>
  <A_relationshipLink_schema source="//A_entity_schema.1" target="//A_relationship_schema.0" participation="TOTAL" cardinality="N" role="" isIdentifier="true"/>
  <A_attribute_schema name="Numero" attribute="IDENTIFIER"/>
  <A_attribute_schema name="Nome"/>
  <A_attribute_schema name="CPF" attribute="IDENTIFIER"/>
  <A_attribute_schema name="Telefone" attribute="MULTIVALUED"/>
  <A_attribute_schema name="Endereco"/>
  <A_attribute_schema name="CEP"/>
  <A_attribute_schema name="Logradouro"/>
  <A_attributeEntityLink_schema target="//A_attribute_schema.0" source="//A_entity_schema.1"/>
  <A_attributeEntityLink_schema target="//A_attribute_schema.1" source="//A_entity_schema.1"/>
  <A_attributeEntityLink_schema target="//A_attribute_schema.2" source="//A_entity_schema.0"/>
  <A_attributeEntityLink_schema target="//A_attribute_schema.3" source="//A_entity_schema.0"/>
  <A_attributeEntityLink_schema target="//A_attribute_schema.4" source="//A_entity_schema.0"/>
  <A_compositeAttributeLink_schema source="//A_attribute_schema.4" target="//A_attribute_schema.5"/>
  <A_compositeAttributeLink_schema source="//A_attribute_schema.4" target="//A_attribute_schema.6"/>
</ERMM:Schema>
```

Figura 4.14 – Representação do MER Cliente no ERMM em XML.

4.3 Análise Comparativa entre Trabalhos Relacionados e ERMM/ERCASE

O Capítulo 3 apresenta uma análise comparativa dos trabalhos relacionados a este. A comparação entre esses trabalhos relacionados e o trabalho aqui proposto é apresentada da seguinte forma: A seção 4.3.1 compara apenas as propostas de metamodelos e seção 4.3.2 analisa as ferramentas CASE para MER.

4.3.1 Metamodelos MER e ERMM

O Quadro 4.1 compara os trabalhos analisados de metamodelos com o ERMM. Assim como apresentado no capítulo 3 são utilizados os marcadores + e -, que denotam, respectivamente, a proposta oferece suporte, não oferece suporte.

Em relação aos trabalhos das seções 3.1 e 3.2, ERMM caracteriza-se como uma proposta nova, que visa contribuir para a especificação de esquemas conceituais de um MER. Para isto, o metamodelo engloba os construtores do MER, destacados na seção 2.1. Contudo, devido a limitações de tempo o conceito de Entidade Associativa não é implementado no ERMM, ficando para trabalhos futuros para a evolução do metamodelo.

Itens	CWM	IMM	ERMM
Entidade Regular	+	+	+
Entidade Associativa	-	-	-
Entidade/Relacionamento Fraco	-	+	+
Atributo Simples	+	+	+
Atributo Composto	-	-	+
Atributo Monovalorado	+	+	+
Atributo Multivalorado	-	-	+
Atributo Derivado	-	-	+
Atributo Identificador	+	+	+
Atributo Discriminado	-	-	+
Auto-Relacionamento	-	+	+
Relacionamento <i>N-ário</i>	-	+	+
Relacionamento Binário	-	+	+
Participação do Relacionamento	+	+	+
Cardinalidade do Relacionamento	+	+	+
Papéis	-	+	+
Herança Simples	-	+	+
Herança Múltipla	-	+	+
Herança Total ou Parcial	-	-	+
Herança Disjunta ou Sobreposta	-	-	+
Categoria	-	-	+
Impede Erro Sintático	-	-	+

Quadro 4.1 – Análise Comparativa dos Metamodelos Relacionadas e ERMM

4.3.2 Ferramentas CASE MER e ERCASE

O Quadro 4.2 compara os trabalhos existentes de ferramentas CASE com o ERCASE. Os marcadores +, -, e ∅ são utilizados na avaliação das propostas e têm o mesmo significado apresentado na seção anterior.

Itens	brModelo	DBMain	SmartDraw	ERCASE
Entidade Regular	+	+	+	+
Entidade Associativa	+	-	-	-
Entidade/Relacionamento Fraco	+	-	+	+
Atributo Simples	+	+	+	+
Atributo Composto	+	-	+	+
Atributo Monovalorado	+	+	+	+
Atributo Multivalorado	+	+	+	+
Atributo Derivado	-	-	-	+
Atributo Identificador	+	-	-	+
Atributo Discriminado	-	-	-	+
Auto-Relacionamento	+	+	+	+
Relacionamento Binário	+	+	+	+
Relacionamento <i>N-ário</i>	+	+	+	+
Participação do Relacionamento	+	+	+	+
Cardinalidade do Relacionamento	+	+	+	+
Papéis	+	+	+	+
Herança Simples	+	+	-	+
Herança Múltipla	-	-	-	+
Herança Total ou Parcial	+	+	-	+
Herança Disjunta ou Sobreposta	-	-	-	+
Categoria	-	-	-	+
Impede Erro Sintático	-	+	-	+

Quadro 4.2 – Análise Comparativa das Ferramentas Relacionadas e ERCASE

4.4 Considerações Finais

Este capítulo apresentou o ERMM, um metamodelo baseado nos conceitos e notação de Navathe para o MER. A escolha desta notação foi motivada pelo fato desta ser a mais difundida na comunidade acadêmica, possuir rica semântica de construtores para o MER (i.e, Categoria, Tipos de Atributos, Relacionamento *N-ário*) e não implementar conceitos referentes ao projeto lógico (i.e, *primary key*, *foreign key*). Logo, o ERMM, diferentemente dos metamodelos analisados (ver capítulo 3), é dedicado para o projeto conceitual e serve como base para ferramentas CASE projetadas para subsidiar o MER.

Para mostrar a viabilidade de implementação do ERMM, a ferramenta ERCASE foi desenvolvida. Esta ferramenta tem como vantagens: 1) apresentar uma arquitetura modular, o que facilita consideravelmente a futura manutenção do sistema e a geração imediata de novas versões da aplicação; 2) possibilitar que os esquemas modelados sejam armazenados em formato XML /XMI, garantindo o intercâmbio de esquemas com outras ferramentas CASE e pode servir para novos usos como, por exemplo, geração automática do mapeamento do projeto conceitual para o lógico; 3) permitir a avaliação do esquema conceitual modelado e impedir a ocorrência de erros sintáticos em tempo de execução (i.e., Associação entre Relacionamentos, Ciclicidade entre Atributos, Entidades com nomes iguais); e finalmente, 4) garantir portabilidade para diversos sistemas operacionais devido ser implementada nos *frameworks open-source* EMF e GMF do Eclipse. Contudo, ERCASE não implementa o conceito de entidade associativa, conforme preconiza o MER, pois o ERMM não apresenta uma metaclassa que subsidie este construtor na ferramenta. A implementação deste conceito no ERMM e ERCASE é apresentado como sugestão para trabalhos futuros (ver capítulo 6).

Usar uma ferramenta CASE no processo de desenvolvimento de um esquema segundo o MER faz com que o tempo de criação do projeto seja minimizado, o que representa uma redução de custo aliado a um aumento na qualidade. Assim, ERCASE pode auxiliar o projetista a desenvolver um esquema conceitual de BD com mais facilidade. No próximo capítulo, é apresentado dois estudos de caso utilizando a ferramenta ERCASE.

5 Estudo de Caso

Neste capítulo são apresentados dois estudos de caso hipotéticos (sem significado) que visam mostrar a viabilidade do metamodelo ERMM e da sua ferramenta CASE ERCASE. Isto é, mostrar que o metamodelo ERMM e a ferramenta ERCASE dão suporte aos construtores de modelagem da notação de Elsmari&Navathe. Por isso, os esquemas conceituais modelados nos estudos de caso visam testar todos os construtores da notação em questão sem se preocupar com a complexidade de um minimundo real. Neste sentido, o primeiro estudo de caso modela um esquema conceitual de BD dando foco no uso dos elementos básicos do MER. Por sua vez, o segundo estudo de caso modela um esquema conceitual de BD dando foco no uso dos elementos avançados do MER. Neste sentido, as demais seções deste capítulo estão organizadas da seguinte forma: a seção 5.1 apresenta o primeiro estudo de caso; a seção 5.2 apresenta o segundo estudo de caso; e, por fim, a seção 5.3 apresenta as principais considerações sobre este capítulo.

5.1 *Estudo de Caso Hipotético 1: Elementos Básicos do MER*

Nessa seção é apresentado um estudo de caso hipotético que modela um esquema conceitual de BD fazendo uso dos seguintes elementos básicos do MER: entidade, entidade fraca, relacionamento, relacionamento identificador, todos os tipos de atributos (simples, composto, monovalorado, multivalorado, derivado, identificador e discriminador), autorelacionamento, papel, cardinalidade e restrições de participação. Na Figura 5.1, o esquema conceitual artificial modela os seguintes elementos:

1. Entidades e Atributos:

- a. E1, possui os atributos At_Simp e At_Id. Estes atributos são classificados, respectivamente, como Simples e Identificador;

- b. E2, possui os atributos At_Der e At_Mult. Estes atributos são classificados, respectivamente, como Derivado e Multivalorado. Com exceção do atributo At_Mult, todos os demais atributos são monovalorados;
 - c. E3, não possui atributo;
 - d. E4, possui o atributo At_Comp que é subdivido em At_Comp_Simp1 e At_Comp_Simp2. Este atributo é classificado em Composto que, por sua vez, é subdivido em dois atributos Simples, e;
 - e. Ent_Fraca, possui o atributo At_Desc que é classificado como Discriminador.
2. Tipos de relacionamento:
- a. R1, relacionamento de cardinalidade 1:N entre E2 e E1. Ambas as restrições de participação são totais;
 - b. R2, relacionamento de cardinalidade 1:1 entre E1 e E2. A participação de E1 é parcial. A participação de E2 é total;
 - c. R3, relacionamento de cardinalidade 1:N entre E2 e E3. Ambas as restrições de participação são parciais;
 - d. Auto_Rel, relacionamento de cardinalidade 1:N entre E1 (no papel de P1) e E1 (no papel de P2). Ambas participações foram determinadas como parciais;
 - e. R4, relacionamento de cardinalidade N:N. Ambas as participações foram determinadas como totais. Além disso, este relacionamento tem o atributo simples At_Simp_Rel;
 - f. Rel_Ident, relacionamento identificador de cardinalidade 1:N entre E3 e Ent_Fraca. A participação de E3 é parcial, enquanto a participação de Ent_Fraca é total, e;
 - g. Rel_Ter, tipo de relacionamento N:1:N entre E2, E5 e E4. Todas as participações foram determinadas como parciais.

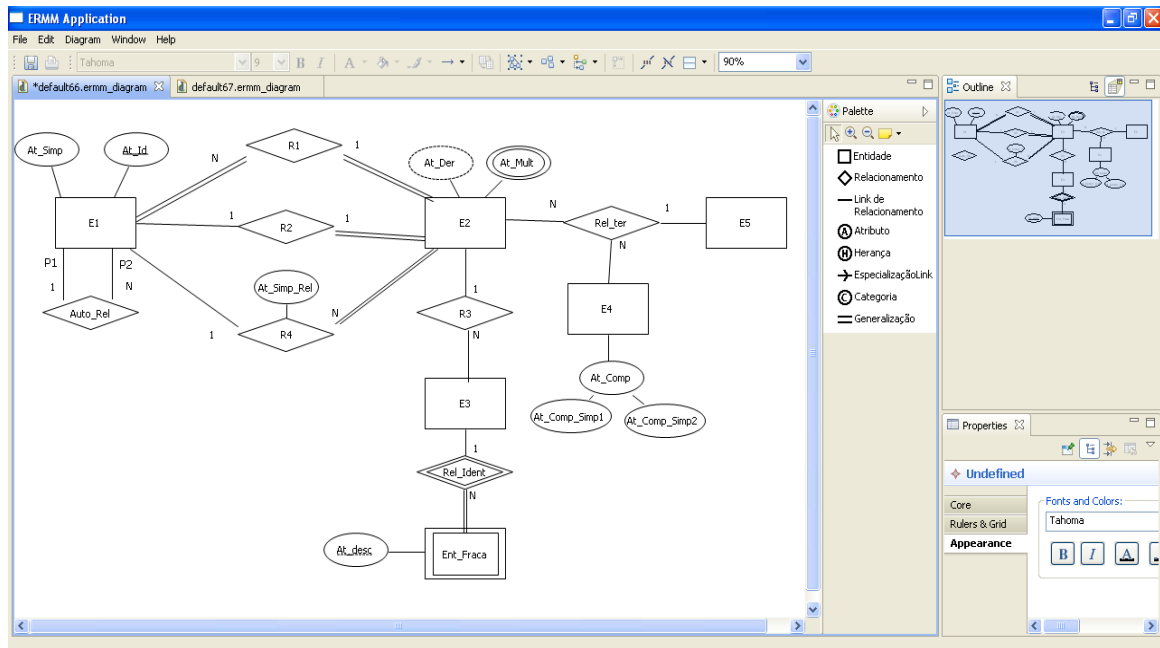




Figura 5.1 – Esquema apenas com elementos básicos do MER

5.1.1 Estudo de Caso Hipotético 1: Verificação do Esquema

Nesta seção, o esquema anterior é alterado para que este tenha erros sintáticos, os quais devem ser identificados pela ferramenta ERCASE. Ressalta-se que a ferramenta ERCASE marca com o símbolo  o elemento que sintaticamente está inconsistente (na seção 4.1.1 encontra-se os erros sintáticos que são evitados pela ferramenta ERCASE). Na Figura 5.1 mostra-se o esquema alterado e com as marcações de erros sintáticos. Note que os erros sintáticos referem-se ao fato que todo Relacionamento tem que ter no mínimo duas associações, o que não aconteceu nos relacionamentos marcados com o símbolo . Ressalta-se que erros sintáticos do tipo: 1) entidades com nomes iguais, 2) ciclicidade em atributos, 3) atributos com o mesmo nome em entidade ou relacionamento, 4) associação entre entidades e 5) associações entre relacionamentos são intrinsecamente impedidos pelo metamodelo ERMM ou via programação da ferramenta ERCASE (ver seção 4.1.1).

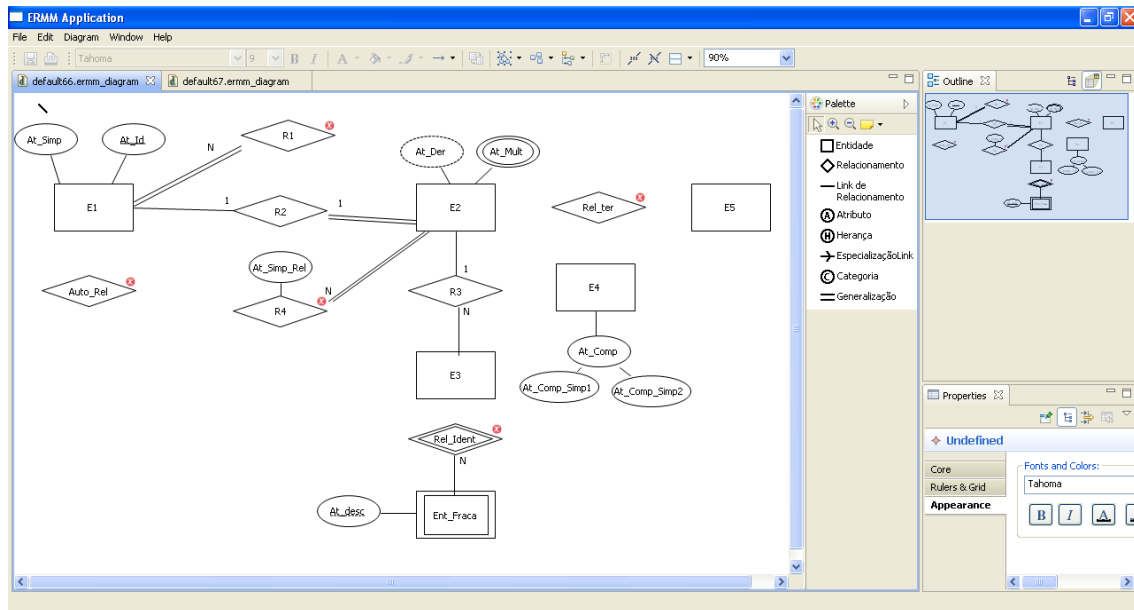


Figura 5.2 – Validação do esquema do estudo de caso 1.

5.2 Estudo de Caso Hipotético 2: Elementos Avançados do MER

Nesta seção é apresentado um estudo de caso hipotético que modela um esquema conceitual de BD fazendo uso dos seguintes elementos avançados do MER: categoria, herança, restrição de disjunção e restrição de totalidade. Na Figura 5.3, o esquema conceitual artificial modela os seguintes elementos:

1. E1 é super-entidade de E2, E3, E4 e E5. A super-entidade possui o atributo Identificador At_Id. A herança das sub-entidades E2 e E3 é disjunta e parcial, enquanto que a herança das sub-entidades E4 e E5 é sobreposta (*overlap*) e total;
2. E3 é super-entidade de E10 e E11. A herança das sub-entidades E10 e E11 é sobreposta e parcial;
3. E4 é super-entidade de E8 e E9. A herança das sub-entidades E8 e E9 é disjunta e total;
4. E8 é categoria das super-entidades E6, E7. Como não pode haver categoria sobreposta [EN05], uma categoria só pode ser total ou parcial. Neste caso, a categoria é parcial, mas poderia ser total.

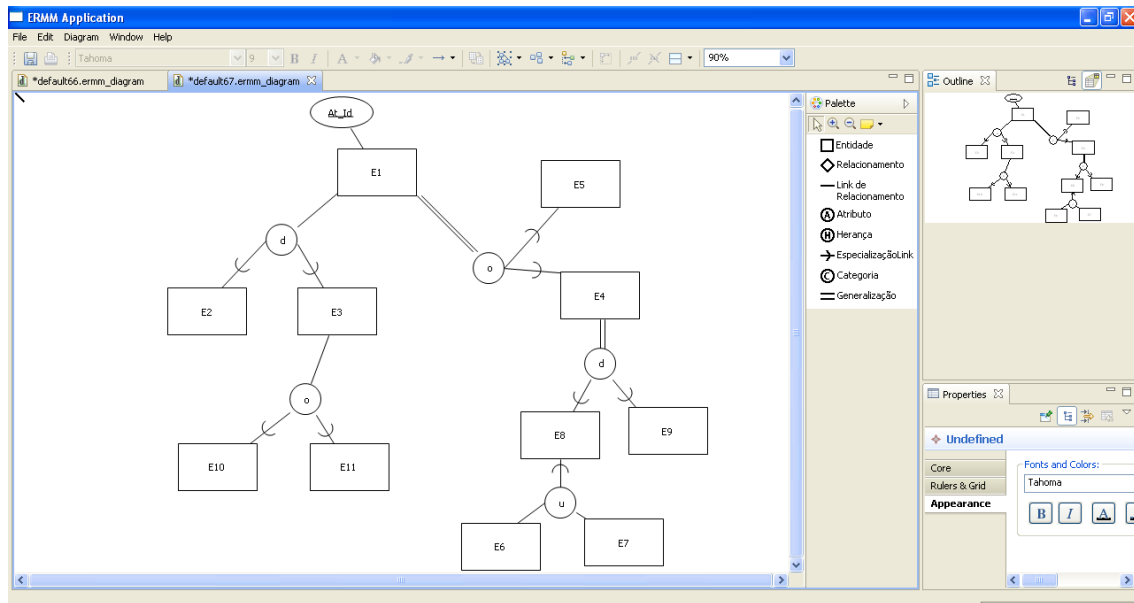



Figura 5.3 – Esquema apenas com elementos avançados do MER

5.2.1 Estudo de Caso Hipotético 2: Verificação do Esquema

Nesta seção, o esquema anterior é alterado para que este tenha erros sintáticos, os quais devem ser marcados com o símbolo  pela ferramenta ERCASE. Na Figura 5.4 mostra-se o esquema alterado e com as marcações de erros sintáticos. Nesta Figura observa-se que os erros sintáticos ocorrem devido: 1) Herança sem Link de Especialização; 2) Herança sem Link de Generalização; 3) Categoria sem Link de Generalização; e, 4) Categoria sem Link de Especialização.

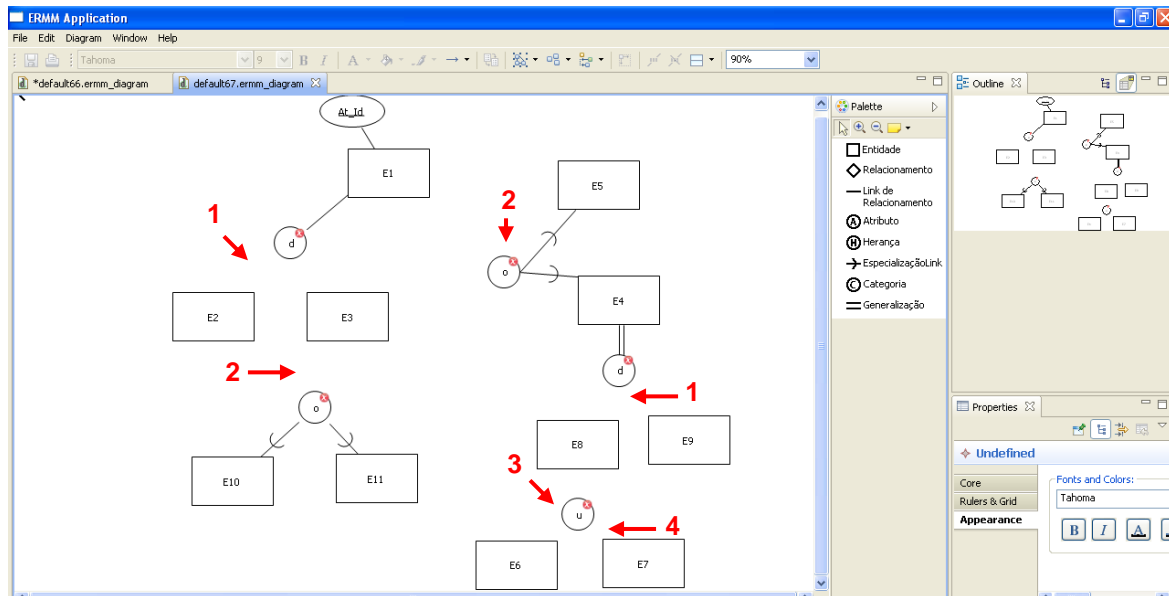


Figura 5.4 – Validação do esquema do estudo de caso 2

5.3 Considerações Finais

Neste capítulo foram apresentados dois estudos de casos hipotéticos que visam mostrar que o metamodelo ERMM e sua implementação na ferramenta ERCASE são factíveis e cumprem com seus respectivos objetivos. Isto é, que estes permitem a modelagem de esquemas conceituais de BD básicos e avançados fazendo uso da notação de Elmasri & Navathe, a qual é uma das mais bem aceitas pela comunidade de BD. Além disso, pode-se constatar que, por meio do metamodelo ERMM e da ferramenta ERCASE, sinaliza-se ou impede-se a ocorrência de erros sintáticos em esquemas conceituais de BD. Por fim, devido a questões de tempo, dentre os recursos da notação de Elmasri&Navathe [8], este trabalho não dá suporte a implementação de entidades associativas (agregações), porém sua implementação é sugerida em trabalhos futuros. No próximo capítulo são apresentadas as conclusões, contribuições e indicações de trabalhos futuros.

6 Conclusão

Este capítulo apresenta as considerações finais sobre o trabalho desenvolvido, suas principais contribuições e sugestões de trabalhos futuros.

6.1 *Considerações Finais*

Conforme discutido na seção 3.1, há várias propostas de modelos de dados destinados ao projeto conceitual de BD. Um dos principais problemas existentes nesses trabalhos é a inclusão de conceitos referente às estruturas lógica e física de um BD. Logo, a fase conceitual que é destinada à estrutura do BD na visão do usuário fica comprometida. Além disso, os trabalhos analisados não abordam completamente os conceitos do MER. Diante do exposto, este trabalho propõe o metamodelo ERMM e a ferramenta ERCASE. ERMM define os elementos do MER segundo Elsmari&Navathe [8] e restringe algumas operações sintáticas não permitidas pelo MER (e.g., Associação entre Relacionamentos, Associação entre Entidades).

Por sua vez, a ferramenta ERCASE disponibiliza um conjunto de funcionalidades (e.g., avaliação, exportação e geração de metadados em XMI) e recursos gráficos (e.g., organização automática e alinhamento de elementos) que visam tornar o processo de modelagem de um MER mais fácil e mais rápido. ERCASE foi implementada em Java, sobre a plataforma Eclipse com uso do *framework* GMF, e tem como núcleo o metamodelo ERMM. A ERCASE tem como principal característica a avaliação sintática de esquemas conceituais de BD.

Durante o desenvolvimento deste trabalho, questões como extensibilidade e independência de plataforma foram premissas consideradas. Por isso, foram utilizadas apenas tecnologias de código-aberto e extensíveis (e.g., EMF, GMF, OCL, Java, XMI). Desta forma, tanto ERMM pode ser estendido e/ou utilizado como base para o desenvolvimento de outras ferramentas CASE quanto o ERCASE pode ser estendida para abordar outros aspectos de modelagem e funcionalidades. Por exemplo, a modelagem de projetos lógicos para BD.

6.2 Contribuições

Como principais contribuições do trabalho desenvolvido, pode-se destacar:

- A análise dos metamodelos (e.g., CWM, IMM) e ferramentas CASE (e.g., brModelo, DBMain, *SmartDraw*) para verificar se estão em conformidade com o MER segundo Elsmari&Navathe [8];
- Criação do metamodelo ERMM para a especificação do MER baseado em Elsmari & Navathe;
- Desenvolvimento da ferramenta ERCASE utilizando o metamodelo ERMM e tecnologias baseadas em modelos de código-aberto e extensíveis (e.g., EMF, GMF), e;
- Avaliação do metamodelo ERMM e da ferramenta ERCASE com a elaboração de estudos de casos hipotéticos para verificar sua aderência ao MER segundo Elsmari&Navathe.

6.3 Trabalhos Futuros

A seguir, são sugeridos alguns trabalhos futuros que podem evoluir ou estender a proposta apresentada nesta dissertação.

- Desenvolver módulos para transformação automática do projeto conceitual para o projeto lógico de BD;
- Desenvolver módulos que permitam a geração automática de código para SGBD. Por exemplo, SQL Server [24] e PostgreSQL [29];
- Estender ERMM e ERCASE de forma a permitir o conceito de Entidade Associativa (agregação), preconizada no MER;
- Avaliar o metamodelo ERMM e a ferramenta ERCASE com a elaboração de estudos de casos reais, e;
- Melhorar a GUI da ferramenta para impedir que linhas fiquem sem tocar em relacionamentos, retirar pop-up durante a criação de atributos e apresentar mensagens que descrevam os erros sintáticos. Além disso, fazer um teste de usabilidade para identificar novas melhorias a serem feitas;

Referências

- [1] BADIA, A. **Entity-Relationship modeling revisited**. *ACM SIGMOD Record*, Vol. 3, n. 1, page 77, 2004
- [2] BRUCE, T. **Designing Quality Databases with IDEF1X Information Models**. Prentice Hall. 1990
- [3] CANDIDO, C. H. **brModelo: Ferramenta de Modelagem Conceitual de Banco De Dados**. Monografia. Universidade de Santa Catarina, SC. 2005.
- [4] CHEN, P.P. **The Entity-Relationship Model-Toward a Unified View of Data**. *ACM Transactions of Database Systems*, Vol.1, n. 1, page 9-36, 1976.
- [5] Common Warehouse Metamodel (CWM) Specification 1.1. Disponível em: <http://www.omg.org/spec/CWM/1.1/PDF>. Acesso em: Janeiro, 2011.
- [6] DB-MAIN - The Modeling Framework. (2011). Disponível em: <http://www.db-main.be/>. Acesso em: Janeiro de 2011.
- [7] Eclipse Modeling Project (2011). Disponível em: <http://www.eclipse.org/modeling>. Acesso em: Janeiro, 2011.
- [8] ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. São Paulo: Pearson Addison Wesley, 2005.
- [9] FONSECA, R. **Um Metamodelo e uma Ferramenta CASE para DATAWAREHOUSE Geográfico**. Dissertação de Mestrado. UFPE. 2007
- [10] FROEHLICH, K.; SONG, Y. **Entity-Relationship Modeling: A Practical How-to Guide**. *IEEE Potentials*. Vol. 13, page 29-34, 1995
- [11] Graphical Modeling Framework (2011). Disponível em: <http://www.eclipse.org/modeling/gmp>. Acesso em: Janeiro, 2011.
- [12] HALPIN, T. **Entity Relationship Modeling from an ORM Perspective: Part 3**. *Journal of Conceptual Modeling*, no. 13. 2000.
- [13] HALPIN, Terry. **Entity-Relationship modeling from an ORM perspective**. *Journal of Conceptual Modeling*, December – 1999.

- [14] HARTMANN, S. **Reasoning about participation constraints and Chen's constraints.** *Information Science Research Centre*. Page 105-113, 1992.
- [15] HAY, DAVID C. **A Comparasion of Data Modeling Techniques.** *Essential Strategies, Inc., October 1999.*
- [16] HEUSER, C. **Projeto de banco de Dados.** Instituto de Informática da UFRGS. 4ª Edição. 1998.
- [17] Information Management Metamodel (IMM) Specification 8.0. Disponível em: <http://www.omg.org/cgi-bin/doc?ab/2005-12-2>. Acesso em: Fevereiro, 2011.
- [18] JONER, B. A., WEBER, R. **Understanding Relationships with Attributes In Entity-Relationship Diagrams.** *Proceedings of the 20th international conference on Information Systems*. Page 214-228, 1999.
- [19] KERN, Vinícius Medina. **Modelagem da Informação com IDEF1X: Linguagem, Método, Princípio do Consenso.** Revista Alcance ano VI no 3, 99-107. Itajaí: Editora da UNIVALI, Novembro – 1999.
- [20] MARTIN, J. **Information Engineering: Planning & Analysis, Book II.** Prentice-Hall, Englewood Cliffs, NJ, 1990.
- [21] MENDES, A. **Arquitetura de Software Desenvolvimento Orientado a Arquitetura.** Campus, 2002.
- [22] MARTIN, J.; FINKELSTEIN, C. **Information Engineering.** Technical Report (2 volumes), Savant Institute, Carnforth, Lancs, UK. 1981.
- [23] MOODY, D.L.; SHANKS, G.G. **What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models.** *Proc. of 13th International Con. On Entity-Relationship Approach*. Vol. 881, pages 94-111, 1994.
- [24] Microsoft SQL Server (2011). Disponível em: <http://www.microsoft.com/sqlserver>. Acesso em: Fevereiro, 2011.
- [26] MELLOR, S.; SCOTT, KENDALL; UHL, A. **MDA Destilada: Princípios de Arquitetura Orientada a Modelos.** Editora Ciência Moderna. 2005.

- [27] Object Constraint Language 2.0 (OCL). Disponível em: <http://www.omg.org/spec/OCL/2.0>. Acesso em: Janeiro, 2011.
- [28] Object Management Group (OMG). Disponível em: <http://www.omg.org/>. Acesso em: Janeiro, 2011.
- [29] PostgreSQL (2011). Disponível em: <http://www.postgresql.org.br/>. Acesso em: Fevereiro, 2011.
- [30] SmartDraw (2011). Disponível em: <http://www.smartdraw.com>. Acesso: Janeiro, 2011.
- [31] SONG, Il-Yeol; EVANS, Mary; PARK, E.K. **A Comparative Analysis of Entity-Relationship of Diagrams.** *Journal of Computer and Software Engineering*, Vol.3, n. 4, pages 427-459,1995.
- [32] STEINBERG, D.; BUDINSKY, F.; PATERNOSTRO, M. **EMF: Eclipse Modeling Framework.** Editora Sereis Editor . 2ª edição. 2008.
- [33] SONG, I.Y; JONES, T.J. **Analysis of binary relationships within ternary relationships in ER Modeling.** *12th International Conference on the Entity-Relationship Approach*.LNCS, Vol. 834, pag. 271-282, 1994.
- [34] TEOREY, T. J.; YANG, J.F. **A Logical Design Methodology for Relational Databases using the extend Entity-Relationship Model.** *ACM Computing Survey*, Vol. 18, n. 2, pages 197-222, 1990.
- [35] XML Metadata Interchange Specification 2.1 (XMI). Disponível em: www.omg.org/spec/XMI/2.1.1. Acesso em: Janeiro, 2011.

Apêndice A

O Metamodelo ERMM

