Uma Abordagem Orientada a Modelos para Modelagem Conceitual de Banco de Dados

André S. Rosa[1], Italine S. Gonçalves[1], Nilson Mori[1], Carlos Eduardo Pantoja[1]

[1]CEFET/RJ - UnED Nova Friburgo, Rio de Janeiro, Brasil

andre_souza.rosa@hotmail.com, italine.goncalves@hotmail.com, nilsonmori@gmail.com, pantoja@cefet-rj.br

Abstract. *This paper proposes an MDA approach that aims to automate the steps of the modeling process database receiving as input a conceptual model and generates the DDL code in the SQL standard 92, 99 and 2003 that will be used to implement the desired data base in a DBMS. The tool is a set of plugins for the development environment Eclipse and will be developed based on MDA approach being used to build the metamodel that allow the instantiation of different notations of modeling the EMF and the implementation of transformations the MOFM2T language processing, pattern provided by the OMG.*

Resumo. *Este artigo propõe uma abordagem MDA que tem como objetivo automatizar as etapas do processo de modelagem de banco de dados, recebendo como entrada um modelo conceitual e gerar o código DDL no padrão SQL 92, 99 e 2003 que será utilizado para a implementação da base de dados desejada em um SGBD. A ferramenta será um conjunto de plugins para o ambiente de desenvolvimento Eclipse e será desenvolvida com base na abordagem MDA, sendo utilizado para a construção do metamodelo que possibilitará a instanciação das notações distintas de modelagem conceitual o EMF e para a implementação das transformações a linguagem MOFM2T, padrão disponibilizado pela OMG.*

Palavras-chave: Modelagem Conceitual, Arquitetura Orientada por Modelos, Transformação de Modelos para texto.

1. Introdução

Um projeto de banco de dados é composto por três fases distintas: a modelagem conceitual, o projeto lógico e físico (Heuser, 2009). A transição entre eles, até a efetiva implementação de um banco de dados em um Sistema Gerenciador de Banco de Dados (SGBD), é um processo que demanda tempo e esforço por parte do designer de banco de dados.

A modelagem conceitual é independente de aspectos tecnológicos, é a representação dos dados em um alto nível de abstração, através de uma descrição diagramática (Abreu; Machado, 1999). Existem notações para modelagem conceitual de banco de dados como a Entidade-Relacionamento de Chen (1976), IDEF1X (Hay, 1999), Backman (Hay, 1999), Crow'sFoot, Min-Max (Hay, 1999) e UML, que não é

específico para modelagem de banco de dados, mas é amplamente utilizado (Guedes, 2008).

Existem algumas ferramentas para projetar bancos de dados que possibilitam a modelagem conceitual, porém são voltadas para a implementação da base de dados em SGBD específicos como o DBDesigner para MySQL (DbDesigner, 2012), o Visual Database Tools para MS SQL Server (VisualDatabaseTools, 2012) e o DBExplorer (DBExplorer Homepage, 2012), que é versátil, mas não é livre. Em alguns casos, elas utilizam uma notação para modelagem de dados específica, sua própria linguagem de modelagem ou modificada de outras. O atrelamento entre a ferramenta e o SGBD, entre a ferramenta e uma notação específica ou entre ambos, limita o processo de modelagem a uma metodologia pré-estipulada pela ferramenta utilizada. Isso acaba retirando do projetista de banco de dados, a possibilidade de utilizar diferentes notações de modelagem conceitual, pode também impedi-lo de destinar uma modelagem feita em determinada ferramenta a diferentes SGBD e ainda pode gerar dificuldade de adaptação do projetista à ferramenta.

Esse artigo propõe uma ferramenta construída com base na abordagem MDA (Mellor et al., 2005), que visa automatizar o processo de modelagem de banco de dados tendo como entrada de dados o modelo conceitual da base de dados, aplicando sobre ele transformações com base na linguagem MOFM2T, disponibilizada pela OMG, resultando na geração do código DDL compatível com os padrões SQL 92/99/2003. A ferramenta possuirá um metamodelo aderente a notações de modelagem conceitual distintas como a ER, Crow's Foot, IDEF1X e UML, possibilitando que o projetista de banco de dados escolha qual das notações deseja utilizar, facilitando o processo da modelagem. O código gerado sob os padrões SQL permite a implementação em uma variedade de SGBD, concedendo ao projetista de bando de dados flexibilidade no processo de modelagem.

O metamodelo, utilizado pela ferramenta, será construído através do Eclipse Modeling Framework (EMF) e as transformações serão implementadas utilizando o Acceleo, resultando em plugins para o Eclipse. Este artigo está estruturado da seguinte forma: a seção dois descreve alguns conceitos básicos; a seção três mostra a abordagem MDA proposta; a seção quatro apresenta exemplos utilizando três notações de modelagem conceitual distintas utilizando a abordagem proposta; a seção cinco apresenta alguns trabalhos relacionados e a seção seis conclui o artigo.

2. Modelagem Conceitual de Banco de Dados

Esta seção descreve as notações de modelagem de banco de dados que são aderentes ao metamodelo proposto, o Model Driven Architecture (MDA), o Eclipse Modeling Framework e a linguagem de transformação de modelos para texto MOFM2T.

2.1. Notações de Modelagem

A modelagem de dados consiste em determinar a estrutura do banco de dados, usando uma notação textual ou gráfica para o nível de abstração desejado, existindo várias técnicas usadas para construção desse modelo.

O Modelo de Entidade-Relacionamento (ER) de Chen (1976) usa uma técnica diagramática com entidades e relacionamentos, com base em três notações diferentes

para modelar informações do mundo: Modelo de Rede, Teoria dos Conjuntos (Halmos, 1960) e Modelo Relacional (Codd, 1970). A figura 1 exibe um exemplo da notação ER.

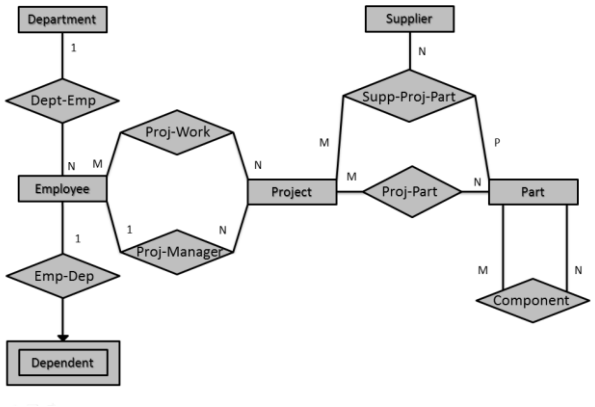

Figura 1 - Exemplo usando a notação Entidade-Relacionamento.

A notação de Crow's Foot é frequentemente usada no processo de modelagem de dados. Nesta notação, as entidades são representadas por quadrados contendo o seu nome, as relações entre elas são representadas por linhas que ligam umas as outras e as cardinalidades são representadas por formas distintas localizadas nas extremidades das linhas (Hay, 1999). A figura 2 exibe um exemplo da notação Crow's Foot.

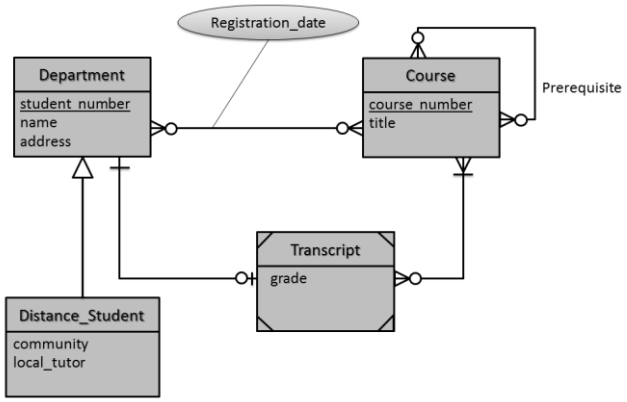

Figura 2 - Exemplo usando a notação de Crow's Foot .

O IDEF1X é uma notação de modelagem conceitual relacional que se baseia no modelo Entidade-Relacionamento e foi tido como padrão de modelagem norte-americano pelo NIST. As entidades nessa notação são representadas por um retângulo que contém uma linha divisória na horizontal, sendo inscritos acima dessa linha os atributos que serão eleitos chave primária da entidade e abaixo dela os demais atributos que poderão ser utilizados para descrever as chaves estrangeiras, se necessário (Bruce, 1992). A figura 3 exibe um exemplo da notação IDEF1X.

A notação Min-Max refere-se à maneira de representar a cardinalidade e é utilizado em combinação com ER para determinar o mínimo e máximo de vezes que a entidade participa da relação. Nesta notação, as cardinalidades são representadas por pares de números inteiros perto da representação da entidade que o elemento pertence. A cardinalidade que está relacionada com a limitação mínima, no lado esquerdo da

notação e a cardinalidade que está relacionada com a limitação máxima no lado direito, tais como: (1, N), (1, 1).

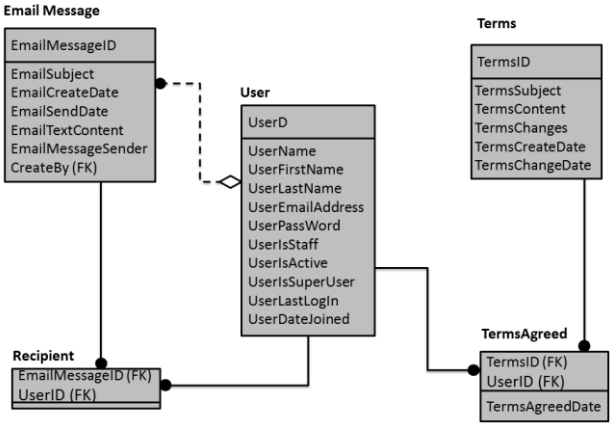

Figura 3 - Exemplo usando a notação IDEF1X.

2.2. Arquitetura Orientada por Modelos - Model Driven Architecture (MDA)

O MDA permite aos desenvolvedores construir modelos do projeto sem o conhecimento de outros modelos de aplicação, depois combiná-las para criar o aplicativo. O uso da técnica MDA permite que a aplicação seja independente de plataforma, aumenta interoperabilidade do projeto e facilita a sua manutenção. O Object Management Group (OMG) é o responsável pela padronização do MDA (Mellor et al., 2005).

2.3. Eclipse Modeling Framework (EMF)

O EMF é um framework integrado ao ambiente de desenvolvimento Eclipse, que gera ferramentas baseadas em modelos estruturados para geração de código. Os metamodelos são criados usando a ferramenta Ecore EMF, que gera um arquivo XMI com base nas construções Ecore. A ferramenta de entrada para o metamodelo recebe um arquivo XML com base na estrutura XMI gerado anteriormente (Steinberg et al., 2008).

2.4. Model To Text (M2T)

O MOF Model To Text (M2T) é uma linguagem padronizada da OMG responsável pela transformação de meta-modelos para artefatos de texto. A ferramenta Acceleo utiliza o M2T e foi projetada para permitir a geração de código.

3. A abordagem MDA

Esta seção demonstra como o metamodelo construído, para ser usado na abordagem MDA para modelagem de banco de dados, está organizado, de forma que seja aderente as diversas notações de modelagem conceitual estando livre de tecnologias específicas (figura 4). Também serão apresentadas as regras de transformação definidas com base na linguagem MOFM2T e a forma que elas agem para que possam transformar o modelo conceitual de entrada de dados em código SQL ANSI 92/99/03 DDL.

A estrutura do metamodelo construído possui o elemento *Model,* que representa o modelo a ser definido na sua totalidade de componentes, estando este relacionado ao elemento *Database* através do relacionamento *isFormedOf,* que determina que a

modelagem se direciona a uma única base de dados gerada com base na notação de modelagem conceitual assumida pelo projetista. Um banco de dados é composto por, pelo menos, um elemento de entidade, para evitar um banco de dados vazio. O elemento *Entity* representa uma entidade que tem algumas informações a serem mantidas no banco de dados, sendo essas informações, representadas através do metamodelo pelo elemento *Field*. Possui também um auto-relacionamento chamado *subGroupOf*, que é responsável pela representação da especialização de um grupo de entidades e verificar elementos de *Entity* para satisfazer o comando de verificação SQL e o relacionamento hasCheck com o elemento *Check*, possibilitando a definição de uma validação para um campo específico. Um elemento *Field* tem relações com *PrimaryKey* e *ForeignKey*, elementos-chave a serem implementadas na camada física e, também, tem relações com os elementos de restrições SQL *TextLimit*, *Integrity*, *DefaultValue* e *NumericLimit*.

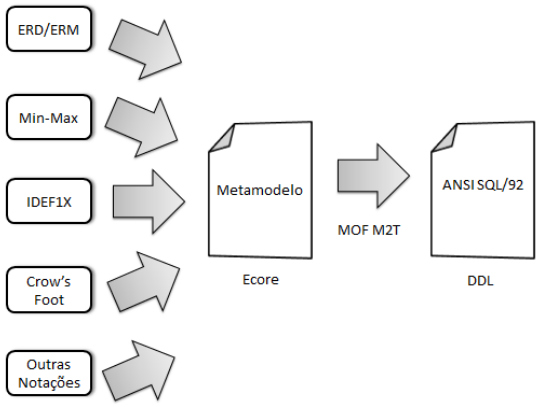

Figura 4 - A abordagem proposta.

A representação de *PrimaryKey* e *ForeignKey* como componentes individuais em relação à *Field*, ao invés de serem uma especialização, foi utilizada por ocorrem casos de modelagem em que uma *PrimaryKey* de uma entidade também pode ser uma *ForeignKey* para outra, não podendo assim, haver uma opção exclusiva, sendo mais coerente modelar para esse caso, a forma em que se encontra o metamodelo.

A interação entre as entidades é representada por suas relações umas com as outras e, deste modo, surge à necessidade da representação das mesmas através do elemento *Relationship* no metamodelo. Esse elemento está relacionado com elementos *Entity* para determinar quem são os elementos envolvidos nesse relacionamento. Uma relação possui cardinalidades limitando a participação de cada elemento, que pode ser de 1:1, 1: N e N: N. Para a cardinalidade N:N, uma tabela associativa é gerada a partir da relação e tem *Field*, *ForeignKey* e elementos *PrimaryKey* como um elemento de Entidade. O metamodelo proposto pode ser visto na figura 5.

O template *ModelToText* vai iniciar o processo de transformação obtendo todos os elementos *Database* em um elemento *Model* e evocar o template *toDatabase*. O template *toDatabase* irá evocar os templates *toEntity* e *crtAsTbl*. O template *toEntity* terá a função de receber como argumento do elemento da classe DataBase um conjunto de elementos da classe *Entity*, iterando por esse conjunto através de um laço de repetição, transformando cada instância de *Entity* contida nesse conjunto no código DDL de criação da tabela que definirá a entidade no banco de dados. O template *crtAsTbl* irá

criar as tabelas para as tabelas associativas nas quais a cardinalidade é N: N e o atributo *isAssociative* é verdadeiro.

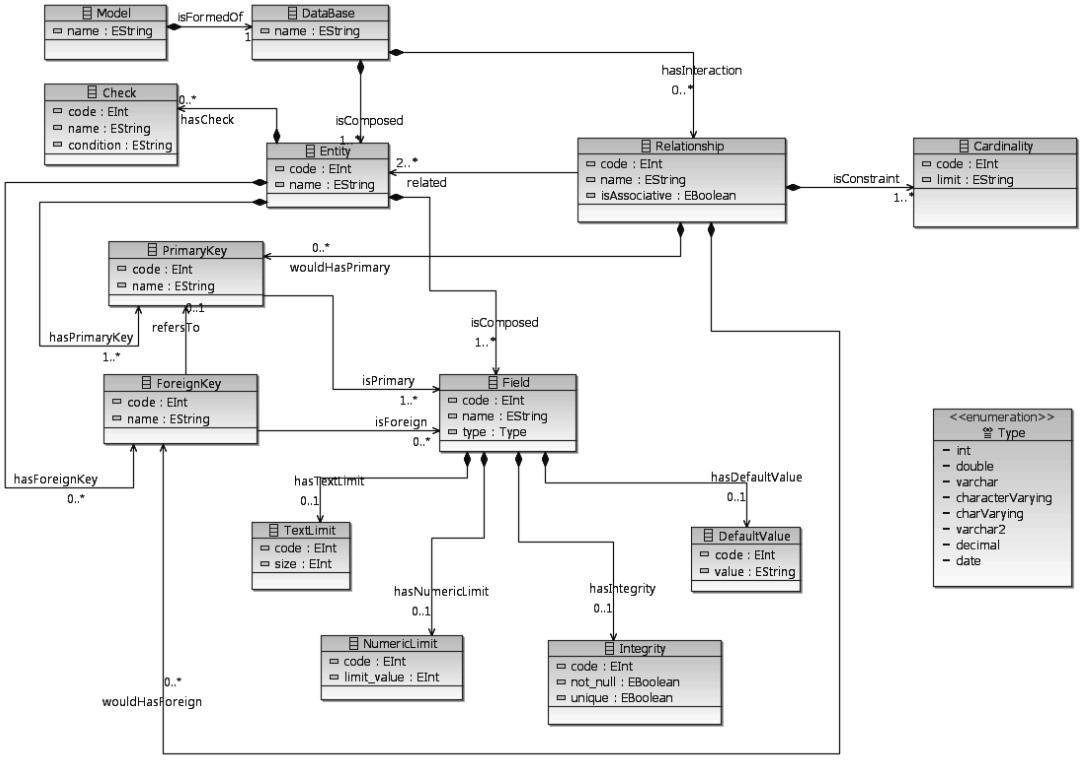

Figura 5 – O Metamodelo.

O template *toEntity* evoca outros templates: *toField*, *prtPrimary*, *prtForeign* e *prtCheck*. O template *toField* recebe um conjunto de elementos de *Field* e irá gerar uma coluna para cada campo da entidade. Posteriormente, os templates de restrição *printNumericLimit*, *printTextLimit*, *printIntegrity* e *printDefaultValue* são evocados.

O template de *prtPri* recebe um conjunto de elementos de *PrimaryKey* que foi modelado para identificar unicamente cada linha de uma tabela e gerar o código SQL. O template *prtForeign* recebe um conjunto de elementos *ForeignKey* e gera o código de chave estrangeira que faz referência a outra tabela. O template de *prtCheck* gera uma validação para uma coluna especifica que foi previamente modelada.

O template *prtRelationCharacteristics* será evocado para gerar as colunas de uma tabela associativa. Este template evoca a *prtFieldsN_N* para gerar as colunas-chave de uma tabela e chama *prtRelFields* que verifica se existe algum atributo da relação a ser gerado. O conjunto de transformações pode ser visto na figura 6.

4. Exemplos

Nessa seção serão apresentados três exemplos de transformações realizados através da ferramenta, utilizando como base para teste os exemplos de modelagem as notações ER, Crow's Foot e IDEF1X apresentadas nas figuras 1, 2 e 3 deste artigo, respectivamente.

A ferramenta foi construída na plataforma de desenvolvimento Eclipse, utilizando o EMF para criar o metamodelo com base na abordagem MDA. O gerador de códigos

Acceleo foi usado para permitir a implementação de regras de transformação e gerar o código SQL no EMF.

```
[template public ModelToText(aModel : Model)]
[comment @main/]
[file (aModel.name+'.txt', false, 'UTF-8')]
    [toDataBase(aModel.isFormedOf)/]
[/file]
[/template]

[template public toEntity(aEnt : Set(Entity))]
[for (iEnt : Entity | aEnt)]
create table '[iEnt.name/]'(
    [toField(iEnt.isComposed)/]
    [prtPri(iEnt.hasPrimaryKey)/]
    [prtForeign(iEnt.hasForeignKey, aEnt)/]
    [if (iEnt.hasCheck<>null)]
    [prtCheck(iEnt.hasCheck)/]
    [/if]);
[/for]
[/template]

[template crtAsTbl(aRel : Set(Relationship))]
[for (iRel : Relationship | aRel)]
[if (iRel.isAssociative = true)]
create table '[iRel.name/]' (
    [prtRelationCharacteristics(iRel.related)/]
    [prtRelFields(iRel)/]
    [prtFieldsN_N(iRel, iRel.related)/]
)
[/if]
[/for]
[/template]


[template public toField(aFields : Set(Field))]
[for ( iField : Field | aFields ) ]
[iField.name/] [iField.type/]
[if (iField.hasNumericLimit <> null)]
[prtNumLimit(iField.hasNumericLimit)/]
[/if]
[if (iField.hasTextLimit <> null)]
[prtTextLimit(iField.hasTextLimit)/]
[/if] [prtIntegrity(iField.hasIntegrity)/]
[if (iField.hasDefaultValue <> null)]
[prtDefVal(iField.hasDefaultValue)/]
 [/if],
[/for]
[/template]

[template public prtTextLimit(aTxt: TextLimit)]
([aTxt.size/])
[/template]


[template public prtNumLimit(aNum : NumericLimit)]
([aNum.limit.value/])
[/template]

[template public prtIntegrity(aInt : Integrity)]
[if(aInt.not_null=true)]not null[/if]
[if ( aInt.unique=true)] unique[/if]
[/template]

[template public prtDefVal(aDef : DefaultValue)]
Default '[aDef.value/]'
[/template]
```

```
[template public prtPri(aPri:Set(PrimaryKey))]
Primary Key(
[for (iP: PrimaryKey | aPri) separator (', ')]
[iP.isPrimary.name/]
[/for])
[/template]

[template public prtForeign(aF : Set(ForeignKey)
,aEnt : Set(Entity))]
[for (iFor : ForeignKey | aF) separator (', ')]
[if (iFor.name <> null)]
    Constraint [iFor.name/] Foreign Key
([iFor.isForeign.name/])
[prtRef(aEnt)/]
[/if]
[/for]
[/template]

[template public prtCheck (aChk : Set(Check))]
[for (iChk : Check | aChk)]
Constraint [iChk.name/]CHECK([iChk.condition/])
[/for]
[/template]

[template public prtRelationCharacteristics
(aEnt : Set(Entity))]
[for (iEnt : Entity | aEnt)]
    [printCandidateKeyFieldsN_N(iEnt)/]
[/for]
[/template]

[template public prtFieldsN_N(aEnt : Entity)]
[for (iPri : PrimaryKey | aEnt.hasPrimaryKey)]
    [toField(iPri.isPrimary)/]
[/for]
[/template]

[template prtRelFields(aRel : Relationship)]
[if (aRel.isMapping<>null)]
    [toField(aRel.isMapping)/]
[/if]
[/template]

[template public prtRef( aEnts : Set(Entity))]
[for (iEnt : Entity | aEnts)]
 [if (iEnt.hasForeignKey<>null)]
[for (iF : ForeignKey | iEnt.hasForeignKey) ]
References [iEnt.name/]
   ([iF.refersTo.isPrimary.name/])
[/for]
 [/if]
[/for]
[/template]
```

Figura 6 - O conjunto de transformações propostas.

É possível instanciar o modelo desejado na ferramenta de duas maneiras: com a entrada de um arquivo XML, o qual possuirá, em suas marcações, a especificação dos elementos que deverão estar presentes na base de dados ou através da instancia de cada elemento, necessário para definição da base de dados, utilizando a árvore hierárquica. A figura 7 representa a modelagem conceitual feita de acordo com a notação ER da figura 1 após ser instanciada na ferramenta exposta visualmente através de uma árvore hierárquica contendo os elementos relativos às características presentes na modelagem.

Da mesma forma como apresentado no modelo anterior, a figura 8 representa a modelagem conceitual feita com base na notação de Crow's Foot exemplificada na figura 2 após ser instanciada na ferramenta através da árvore hierárquica contendo os elementos utilizados na sua modelagem. A figura 9 representa o exemplo da modelagem

feita com base na notação IDEF1X na figura 3 exposta na árvore hierárquica contendo os elementos para sua instanciação na ferramenta seguida do código gerado após ser submetida às transformações.

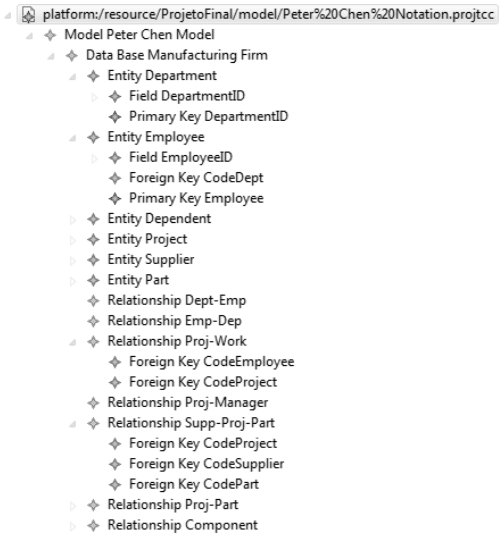

Figura 7 - Árvore hierárquica representando a notação ER.

Após serem submetidos às regras de transformação, os modelos serão transformados no código fonte DDL no SQL ANSI 92/99/03 padrões contendo a descrição da estrutura das tabelas do banco de dados que podem ser implementados em SGBD suportados conforme figura 10.

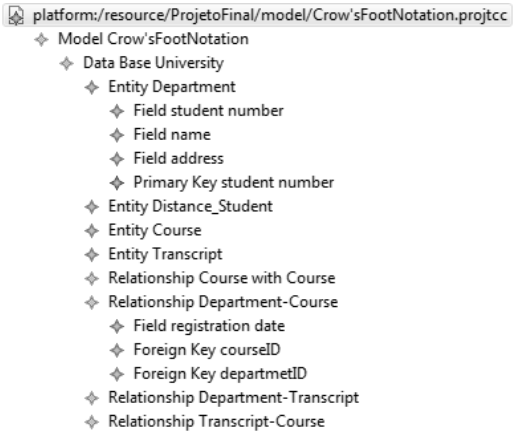

Figura 8 - Árvore hierárquica representando a notação de Crow's Foot.

5. Trabalhos Relacionados

Existem algumas ferramentas relacionadas com a modelagem de banco de dados que são usadas por causa da automação e praticidade oferecida para auxiliar o processo de desenvolvimento, no entanto, algumas dessas ferramentas são pagas e ligadas à SGBD específicos. Há também o fato de que algumas delas têm a sua notação de modelagem própria exigindo mais tempo no desenvolvimento e limitando o desenvolvedor.

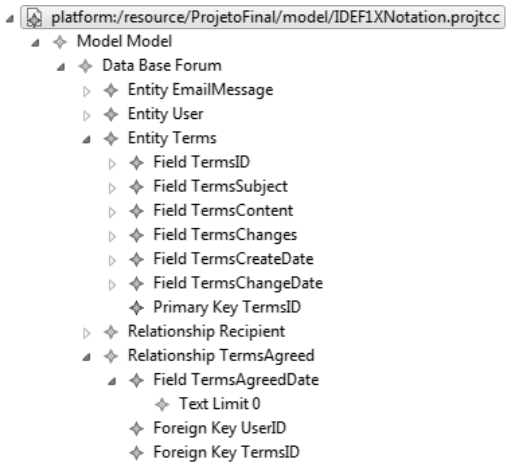

Figura 9 - Árvore hierárquica representando a notação IDEF1X.

```
CREATE DATABASE University,

CREATE TABLE Department(
student number int (7) NOT NULL UNIQUE,
name varchar (20) ,
address varchar (45) ,

CONSTRAINT student number
PRIMARY KEY(student number),
);

CREATE TABLE Distance_Student(
community varchar (50) ,
local_tutor varchar (50) ,
);

CREATE TABLE Course(
course number int (7) NOT NULL UNIQUE,
title varchar (20) ,

CONSTRAINT course number
PRIMARY KEY(course number),
);

CREATE TABLE Transcript(
grade varchar (20) ,
);

CREATE TABLE Course with Course(
course number int (7) NOT NULL UNIQUE,
courseID int  ,
prerequisiteID int  ,

CONSTRAINT courseID
 FOREIGN KEY (course number) REFERENCES
 Course (course number),
 CONSTRAINT prerequisiteID
 FOREIGN KEY (course number)
 REFERENCES Course (course number)
);
```

```
CREATE TABLE Department-Course(
student number int (7) NOT NULL UNIQUE,
course number int (7) NOT NULL UNIQUE,
registration date varchar  ,

courseID int  ,
departmetID int  ,

CONSTRAINT courseID
FOREIGN KEY (course number) REFERENCES
Course (course number),

CONSTRAINT departmetID
FOREIGN KEY (student number) REFERENCES
Department (student number)
);

CREATE DATABASE Forum,

CREATE TABLE User(
UserID int (7) NOT NULL UNIQUE,
UserFirstName int (30),
UserLastName varchar (30),
UserEmailAddress varchar (30),
UserPassWord varchar (30),
UserIsAtaff int (30),
UserIsActive int (30),
UserIsSuperUser int (30),
UserLastLogIn varchar (30),
UserDateJoined varchar (30),

CONSTRAINT UserID PRIMARY KEY(UserID)
);

CREATE TABLE Terms(
TermsID int (7)  NOT NULL UNIQUE,
TermsSubject varchar (30),
TermsContent varchar (30),
TermsChanges varchar (30),
TermsCreateDate varchar (30),
TermsChangeDate varchar (30),

CONSTRAINT TermsID PRIMARY KEY(TermsID)
);
```

```
CREATE TABLE EmailMessage(
EmailMessageID int (7) NOT NULL UNIQUE,
EmailSubject varchar (50),
EmailCreateDate varchar (50),
EmailSendDate varchar (50),
EmailTextContent varchar (50),
EmailMessageSender varchar (50),
CreateBy varchar (50),

CONSTRAINT EmailMessageID
PRIMARY KEY(EmailMessageID).

CONSTRAINT CreateBy
FOREIGN KEY (CreateBy)REFERENCES User (UserID)
);


CREATE TABLE Recipient(
EmailMessageID int (7) NOT NULL UNIQUE,
UserID int (7) NOT NULL UNIQUE,

CONSTRAINT EmailMessageID
 FOREIGN KEY (EmailMessageID)
 REFERENCES EmailMessage (EmailMessageID),

CONSTRAINT UserID
 FOREIGN KEY (UserID)
 REFERENCES User (UserID)
);

CREATE TABLE TermsAgreed(
TermsID int (7),
UserID int (7) NOT NULL UNIQUE,
TermsAgreedDate varchar (10),

CONSTRAINT UserID
FOREIGN KEY (UserID) REFERENCES User (UserID),

CONSTRAINT TermsID
FOREIGN KEY (TermsID) REFERENCES Terms (TermsID)
);
```

Figura 10 - Código SQL gerado.

O DBDesigner, apesar de não ser mais um projeto ativo, é uma ferramenta livre que possui modelagem gráfica, porém é limitada pelo fato de ser voltada exclusivamente para o SGBD MySQL e por possuir notação de modelagem conceitual particular. O Oracle Designer (OracleDesigner, 2012) oferece opções de modelagem como o diagrama de ER, diagrama de funções; e análise e design de objetos, mas é destinado para o Oracle DBMS, que é pago. O CA Erwin (SLIKSoftware, 2012) é compatível com a maioria dos principais SGBD no mercado como MySQL, ODBC, Oracle, Postgree, SQL Server porém tem como notação de modelagem o IDEF1X, que também é uma ferramenta paga.

A abordagem MDA para desenvolvimento de banco de dados proposta neste artigo tem como objetivo atingir o nível mais alto de reutilização, empregando um versátil metamodelo, que pode se adaptar às notações de modelagem diferentes. A saída de dados é gerada por regras de transformação, descritas em MOF M2T, no padrão ANSI SQL que transportam grande potencial para vários SGBD diferentes.

6. Conclusão

O foco deste artigo foi fornecer aos usuários a possibilidade de uso de notações diferentes de modelagem conceitual, utilizando a abordagem MDA. Além disso, oferece uma ferramenta que está mais integrada aos diferentes SGBD, o que contribui para a versatilidade no desenvolvimento e aumenta a reutilização da ferramenta.

Em trabalhos futuros, será desenvolvido um ambiente gráfico para as notações de modelagem aderentes a ferramenta, usando o Graphical Modeling Framework (GMF) e também será incorporada uma ferramenta para perceber os requisitos do sistema e automaticamente instanciar o metamodelo proposto neste artigo. Depois disso, o modelo gráfico poderá ser autogerado assim como o código DDL no padrão ANSI SQL.

Referências

Abreu, M.; Machado, F. N. R. Projeto de Banco de Dados: Uma Visão Prática. Erica, 1999.

Bruce, T. A. Designing quality databases with IDEF1X information models. Dorset House Pub., 1992.

Chen, P. P.-S. The entity-relationship model—toward a unified view of data. ACM Trans. Database Syst., v. 1, n. 1, p. 9–36, 1976.

Codd, E. F. A relational model of data for large shared data banks. Commun. ACM, v. 13, n. 6, p. 377–387, 1970.

DbDesigner Homepage. .Disponível em: <http://dbdesigner.sourceforge.net/.>. Acesso em: 3/9/2012.

Guedes, G. T. A. UML - Uma Abordagem Prática. Novatec, 2008.

Halmos, P. R. Naive Set Theory. Springer-Verlag, 1960.

Hay, D. C. A comparison of Data Modeling Techniques. Essential Strategies. ,1999. Disponível em: <http://essentialstrategies.com/documents/comparison.pdf>. .

Heuser, C. A. Projeto de Banco de Dados. 6º ed. Bookman, 2009.

Mellor, S. J.; Scott, K.; Uhl, A.; Weise, D. MDA Destilada: Princípios de Arquitetura Orientada por Modelos. Ciência Moderna Ltda, 2005.

Oracle Designer. .Disponível em: <http://www.oracle.com/technetwork/developer-tools/designer/overview/index-082236.html>. Acesso em: 19/9/2012.

SLIK Software. .Disponível em: <http://www.sliksoftware.co.nz/products/dbexplorer/index.htm>. Acesso em: 3/9/2012.

Steinberg, D.; Budinsky, F.; Merks, E.; Paternostro, M. Emf: Eclipse Modeling Framework. Pearson Education, 2008.

Visual Database Tools. .Disponível em: <http://msdn.microsoft.com/pt-br/library/y5a4ezk9.aspx>. Acesso em: 3/9/2012.

DBDesigner. Disponível em: <http://dbdesigner.sourceforge.net/>. Acesso em: 19/9/2012.