Uma Metodologia Para Apoio ao Projeto de Banco de Dados Geográficos Utilizando a MDA

João Victor Guinelli[1], André de Souza Rosa[1], Carlos Eduardo Pantoja[1], Ricardo Choren[2]

[1]CEFET/RJ - Campus Nova Friburgo
Av. Gov. Roberto da Silveira, 1900 – Prado – 22.635-000 –
Nova Friburgo – RJ – Brasil

[2]Instituto Militar de Engenharia – IME/RJ – Pça Gen Tibúrcio 80 –
Rio de Janeiro – RJ – Brasil

jvguinelli@gmail.com, andre_souza.rosa@hotmail.com, pantoja@cefet-rj.br

choren@ime.eb.br

Abstract. *This paper presents a MDA methodology for database project with an extension to geographic database projects, that is capable to generate automatic codification for SFS/SQL and ANSI SQL pattern. The methodology uses a generic metamodel for the conceptual and geographic modeling concepts to generate data definition language. This paper also presents a database modeling tool, which is a set of Eclipse plug-ins and a simple example modeled using OMT-G.*

Resumo. *Este artigo apresenta uma metodologia MDA para o projeto de banco de dados com uma extensão para projeto de banco de dados geográficos, que é capaz de gerar codificação automática para o SFS/SQL e o padrão ANSI SQL. A metodologia usa um meta-modelo genérico para os conceitos dos modelos relacionais e geográficos para geração da linguagem de definição de dados. Este trabalho apresenta, também, uma ferramenta para modelagem de banco de dados, que consiste de um conjunto de plug-ins para a plataforma de desenvolvimento Eclipse e um simples exemplo modelado com o OMT-G.*

1. Introduçao

A modelagem consiste em criar modelos para explicar cada característica e comportamento de um sistema e representam uma simplificação da realidade. Os modelos criados durante a modelagem ajudam o projetista a visualizar melhor o sistema, permitem a especificação da estrutura ou do comportamento desse sistema, além de proporcionarem um guia para o seu desenvolvimento e documentarem as decisões tomadas [Booch et al. 2000].

A Arquitetura Orientada por Modelos (MDA) é uma abordagem para desenvolvimento de softwares que é proposta e padronizada pela *Object Management Group* (OMG). Essa abordagem se baseia na criação de modelos em diferentes níveis de abstração que combinados criam a implementação do sistema. A utilização da MDA contribui para criação de softwares independentes da plataforma, com maior interoperabilidade e de

fácil manutenção, já que os modelos criados podem ser alterados, ter novas funcionalidades adicionadas e serem recombinados [Mellor et al. 2005].

As fases do projeto de um banco de dados podem ser dívidas em: i) projeto conceitual; ii) projeto lógico; e iii) projeto físico. No projeto conceitual, uma abstração da lógica do negócio, que precise ser informatizada, é criada independente de utilização de tecnologias de implementação, onde a conceitualização das principais estruturas de armazenamento é gerida utilizando-se de diversos diagramas e modelos gráficos. O projeto lógico é a fase consequente ao projeto conceitual, onde a partir de transformações entre modelos, as estruturas gráficas são transformadas em estruturas textuais lógicas com o objetivo de eliminar erros e redundâncias de dados. O projeto físico de banco de dados é obtido através de uma transformação do modelo anterior e leva em consideração as tecnologias para implementação do banco de dados, que são escolhidas de acordo com a necessidade e recursos do cliente [Elmasri and Navathe 2005].

Há algum tempo, a pesquisa na área de Banco de Dados passou a preocupar-se com o suporte a aplicações não convencionais, que são aplicações que trabalham com tipos de dados não tradicionais, como, por exemplo, dados espaciais, temporais e espaço-temporais [Laender et al. 2005]. Atualmente existem diversos modelos para modelagem de banco de dados geográficos, como o UML-GeoFrame [Lisboa and Iochpe 1999] e o OMT-G, entre outras, além de diversas ferramentas para modelagem conceitual de banco de dados geográficos que utilizam a MDA como a ArgoCASEGEO + TerraLib [Gazola et al. 2006] e OMT-G Design [Schaly and Frozza 2010]. Apesar dessas ferramentas utilizarem a MDA elas estão atreladas a modelos e linguagens de modelagens específicas e não utilizam um meta-modelo genérico, não podendo ser extensível a outros modelos geográficos, além de incompatibilidade com o modelo relacional.

Atualmente existe para modelagem conceitual de banco de dados tradicional a metodologia que utiliza a MDA proposta em [Rosa and Pantoja 2013]. Essa metodologia permite a escolha entre diversas linguagens de modelagem para banco de dados e, a partir de uma meta-modelo genérico e um conjunto de regras de transformação, geram o código SQL/DDL baseado na ANSI SQL 92/99/03. Porém, a ferramenta desenvolvida para a metodologia [Rosa et al. 2013] não possui uma extensão para banco de dados geográficos. Contudo, por utilizar a MDA, a metodologia pode ser extensível para novas especificações e modelos.

Portanto o objetivo deste artigo consiste na proposta da extensão de um meta-modelo genérico para modelagem de banco de dados relacionais, a fim de adaptá-lo para que seja possível realizar a modelagem de banco de dados geográficos; e em uma extensão das regras de transformação para que seja possível a geração de código SFS/SQL para banco de dados geográficos. Uma ferramenta comum a banco de dados relacionais e geográficos também será desenvolvida para testar a metodologia proposta. As contribuições previstas são: uma metodologia estendida para modelagem de banco de dados geográficos; uma ferramenta MDA, onde a implementação do meta-modelo será feita utilizando o Ecore, que é uma ferramenta de meta-modelagem que faz parte do *Eclipse Modeling Framework* (EMF) [Steinberg et al. 2008]; e um conjunto de regras, que serão especificadas utilizando-se o *Model-To-Text* (M2T) [OMG 2008] e a implementação será feita utilizando-se o Acceleo [Obeo 2012], que adota o M2T.

Este artigo está estruturado da seguinte forma: na seção 2 apresenta-se a metodologia MDA, com a revisão do meta-modelo e a extensão das regras de transformação propostas; na seção 3 um exemplo simples utilizando a ferramenta em um projeto de banco de dados geográfico será apresentado; na seção 4 apresenta-se alguns trabalhos relacionados; e por fim na seção 5 a conclusão do trabalho.

2. A Metodologia Proposta

Nesta seção é apresentada a metodologia adaptada utilizada para a geração da codificação automática de linguagem de definição de dados para banco geográficos através de uma ferramenta MDA. Uma revisão das funcionalidades do meta-modelo genérico da metodologia é feita e uma análise de viabilidade para a extensão da metodologia para ser aplicável aos modelos OMT-G e UML-GeoFrame é proposta. Um conjunto de regras de transformação usando a especificação M2T também será apresentada.

A *Model-Driven Architecture* (MDA) ou Arquitetura Orientada a Modelos é uma abordagem da Engenharia de Software que permite a utilização de modelos de domínio que partem de um nível maior de abstração do negócio até a implementação efetiva do software através da utilização de linguagens e especificações para realizar as transformações entre seus diversos modelos [Mellor et al. 2005]. Portanto, a utilização da MDA permite tratar todas as fases do projeto de um banco de dados de forma independente, através da definição de modelos que podem sofrer transformações até a efetiva implementação do banco de dados nas tecnologias escolhidas.

A metodologia para modelagem de banco de dados relacionais [Rosa et al. 2013] utiliza a MDA, como base de uma ferramenta CASE, para permitir a criação da estrutura física do banco de dados através da geração automática de codificação em *Structured Query Language* (SQL), que é uma linguagem de criação e manipulação de banco de dados. A ferramenta permite escolher entre diversas notações e linguagens de modelagens conceitual para banco de dados relacionais existentes, e a partir de uma conjunto de regras de transformação, utilizando a especificação Model-To-Text (M2T), é possível gerar a codificação baseada na norma ANSI SQL 92/99/03. Na figura 1 é possível ver o funcionamento da metodologia.

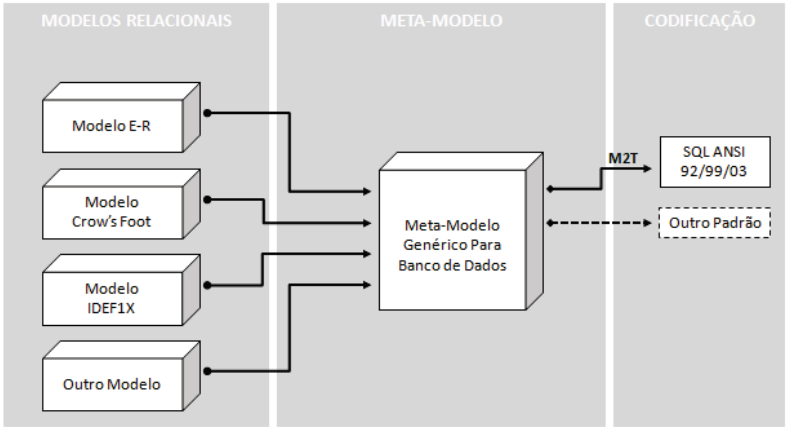

Figure 1. A arquitetura da metodologia.

O diferencial da metodologia desenvolvida é que ela permite a utilização das mais

utilizadas notações e linguagens de modelagem conceitual para banco de dados relacionais e ainda permite a expansão para se adaptar a novas tecnologias, e.g. Banco de Dados Geográficos, sem a obrigatoriedade de se reconstruir uma nova ferramenta. Isso é possível porque o meta-modelo reúne as principais características dessas notações e linguagens de modelagem.

Além disso, a ferramenta construída baseada na metodologia não é atrelada a nenhuma tecnologia, pois usa como base para a criação da estrutura física do banco de dados um padrão normativo internacionalmente conhecido, o *Simple Features Specification* para SQL do *Open Geospatial Consortiun* (OGC), desta forma possibilitando uma maior aderência às tecnologias existentes. E ainda, as regras de transformação continuam sendo as mesmas para qualquer que seja o modelo de entrada, visto que um único meta-modelo pode ser responsável pela instância de diversas linguagens de modelagens e notações.

Para se adaptar esta metodologia a geração de código SQL/DDL para banco de dados geográficos é necessário verificar se as construções geográficas do SFS para SQL são aderentes ao meta-modelo proposto inicialmente. O SFS é uma especificação, proposta pelo OGC, que define de que forma uma componente espacial vetorial de dados geográficos devem ser armazenadas em um banco de dados e propõe um esquema padrão para o armazenamento, leitura, consulta e atualização desses dados geográficos através da SQL. O PostgreSQL e o Oracle Database são exemplo de dois SGBDs convencionais amplamente utilizados atualmente que utilizam extensões geográficas baseadas na especificação SFS para SQL.Nesse caso, será necessário apenas a geração de um novo cartucho com regras de transformação para geração do código geográfico. Caso contrário o meta-modelo deverá ser estendido para conter as construções geográficas não observadas no meta-modelo inicial.

A metodologia também poderá ser estendida para ser adaptável a modelos geográficos específicos, e.g. OMT-G e UML-GeoFrame, necessitando que seja implementada um conjunto de transformações usando a linguagem de transformação *Query-View-Transformation* (QVT) [OMG 2011] entre o modelo escolhido e o meta-modelo genérico. O QVT é uma linguagem de mapeamento de conceitos entre modelos, que é padronizada pela OMG. Dessa forma, se existir alguma ferramenta que utilize um modelo específico e este puder ser transformado para o meta-modelo genérico, a geração da codificação automática poderá ser realizada sem a necessidade de se criar transformações específicas a cada novo modelo a ser adicionado. A metodologia estendida pode ser vista na figura 2.

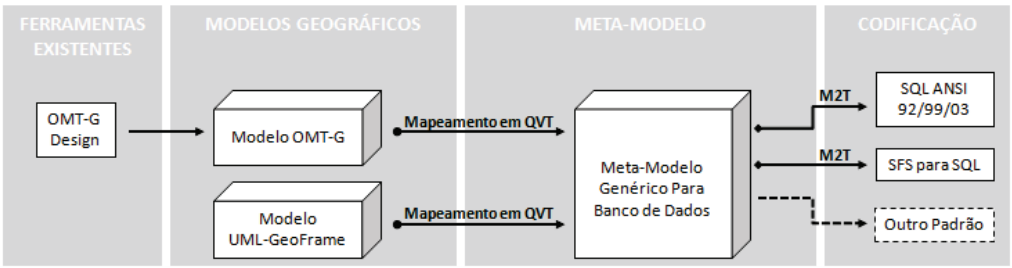

Figure 2. A extensão geográfica da metodologia.

A metodologia gera uma maior flexibilidade ao projetista já que este poderá escolher entre vários modelos; permite a adição de novos modelos e ferramentas através da

integração de mapeamentos usando o QVT; e, ao se usar o MDA, permite a rastreabilidade de conceitos entre o código gerado e o modelo escolhido.

2.1. O Meta-Modelo

O meta-modelo (Figura 3) na metodologia leva em conta aspectos do projeto conceitual e lógico, sendo composto por classes e relacionamentos utilizados para representação dos elementos necessários para modelagem de bancos de dados relacionais. A classe *Model* está no topo da hierarquia das relações do modelo, representando a modelagem e contem todos os elementos que podem ser utilizados. A partir do *Model*, existe o relacionamento com a classe *Database*, que representa a base de dados que será modelada com suas entidades e relacionamentos.

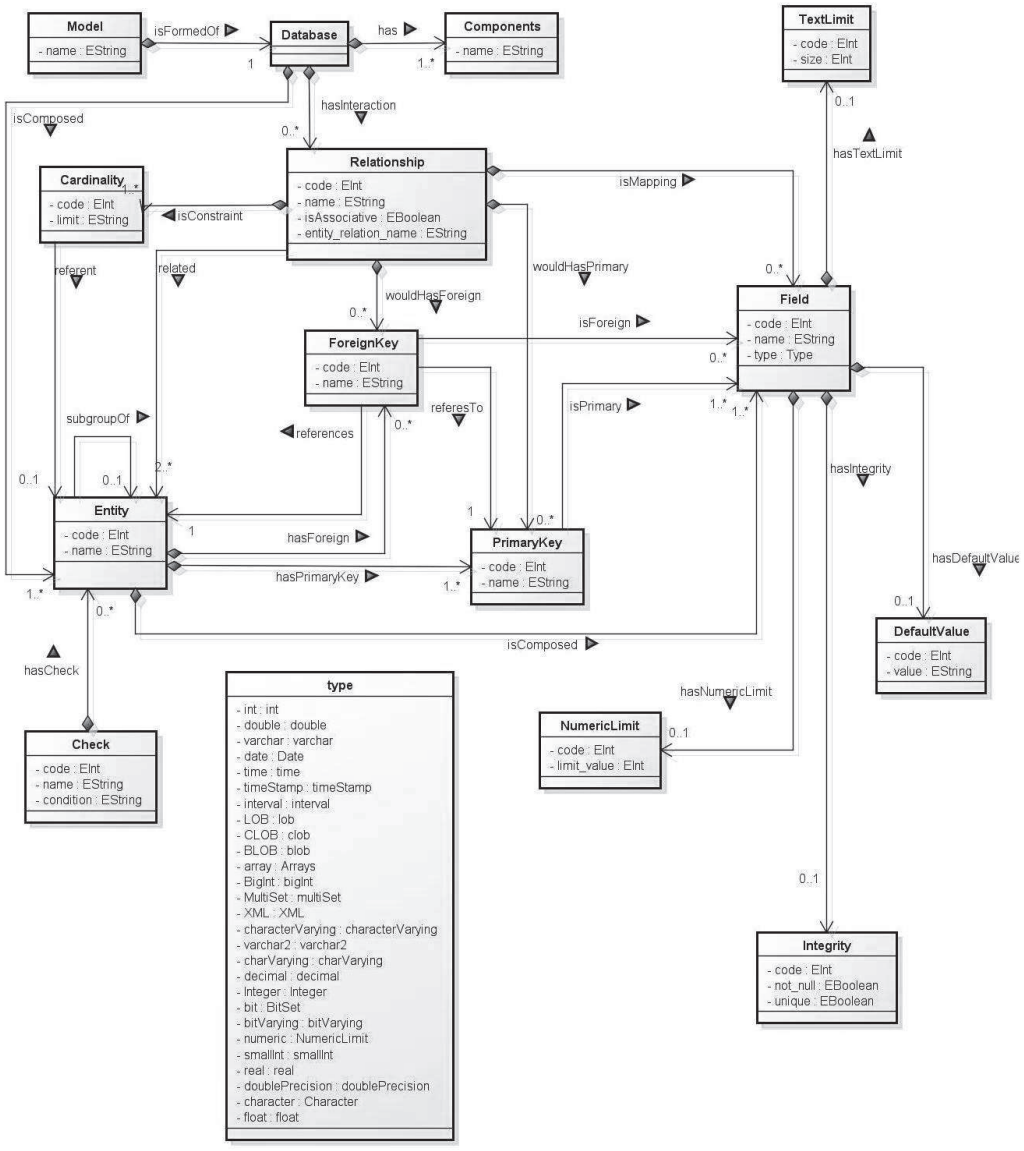

Figure 3. O meta-modelo.

As entidades e relacionamentos são representados no meta-modelo respectivamente pelas classes *Entity* e *Relationship*. A descrição dos atributos de uma entidade,

assim como das relações, quando necessário, é feita através da classe *Field*. Para representar o número de entidades participantes em determinado relacionamento é utilizada a classe *Cardinality*. Os casos de auto-relacionamento das entidades são abordados pela ligação *subgroupOf*.

Algumas classes representam aspectos essenciais pertinentes ao projeto lógico. Para possibilitar a representação da unicidade de cada registro na base de dados, é utilizada a classe *PrimaryKey*. A integridade referencial entre as entidades é representada pela classe *ForeignKey*. Para representação das integridades referentes aos atributos são utilizadas as classes *TextLimit* e *NumericLimit*, para definição de tamanho limite para textos e números, respectivamente. A classe *Integrity* é utilizada para descrever se o valor de um atributo será único e se será permito de valores nulos para o mesmo. A *DefaultValue* é a classe utilizada para representar a propriedade dos atributos de possuir um valor padrão.

Para representar condições referentes a entidades específicas atribuídas pelo projetista, caso necessário, é utilizada a classe *Check*. Está presente no meta-modelo um *enumeration* chamado *Type*, que contem os tipos possíveis a serem atribuídos ao atributos. Esses tipos foram selecionados por estar em conformidade com os padrões ANSI/SQL abordados pela metodologia utilizada.

O SFS da OGC define, para a criação do SQL/DDL de um esquema de banco de dados geográficos, um conjunto de tipos geográficos específicos. A diferença percebida para o SQL/DDL entre o modelo geográfico e o relacional se dá apenas no momento da definição dos tipos de um campo de uma determinada tabela. Dessa forma foram adicionados os tipos *Geometry*, *Point*, *LineString*, *Polygon*, *GeometryCollection*, *MultiPoint*, *MultiLineString* e *MultiPolygon* à classe *Type* do meta-modelo.

2.2. As Regras de Transformação

O processo de transformação automatizada da modelagem feita de um determinado modelo para o código que será utilizado na sua implementação ocorre através de um conjunto das regras definidas utilizando a especificação M2T. As regras são estruturadas em forma de *templates*, sendo cada *template* responsável por uma etapa da transformação.

A geração de código para o modelo relacional ANSI SQL 92/99/03 e para o modelo geográfico SFS/SQL é composto de um conjunto de *templates* iniciado pelo *ModelToText*, que irá criar o arquivo onde será armazenado o código gerado durante o processo, irá receber o modelo instanciado e evocar o próximo *template* envolvido no processo de transformação, que é o *ToDataBase*. Esse *template* irá imprimir o código referente a criação da base de dados e evocar os demais *templates* que irão imprimir o código para geração dos componentes da base. À partir do *ToDataBase*, serão evocados os *templates* *PrintPrecedenceA*, *PrintPrecedenceB* e o *CreateAssociativeTable*.

O *PrintPrecedenceA* tem a função de iniciar a geração do código DDL referente a entidades que não possuam chaves estrangeiras. Esse *template* irá verificar se as mesmas possuem chaves estrangeiras para outras entidades e então será evocado o *template* *PrintEntityWithoutForeign*, que continuará com o processo de geração do código para esse caso. O *PrintEntityWithoutForeign* irá gerar o código das entidades evocando diversos *templates* responsáveis, cada um, por uma característica da entidade. Será evocado o *ToField* para gerar o código referente aos atributos da entidade, o *PrintPrimary* para gerar o código referente às chaves primárias da mesma e irá ser verificado se o objeto da

entidade em questão possui ligação com algum objeto da classe *Check*, para definição de alguma regra de integridade particular. Caso essa verificação seja positiva, irá ser evocado o *template PrintCheck*.

Para imprimir o código dos atributos das entidades, o *template ToField* utiliza os valores armazenados nas instâncias da classe *Field* e verifica se existe relação definida na modelagem com os objetos das classes *NumericLimit*, *TextLimit*, *Integrity* e *DefaultValue*. Para cada um desses será evocado o *template* responsável pela geração do código DDL, sendo respectivamente os *templates PrintNumericLimit*, *PrintTextLimit*, *PrintIntegrity* e *PrintDefaultValue*.

O *template PrintPrecedenceB* tem função similar ao *template PrintPrecedenceA*, sendo sua função verificar as entidades que possuam chaves estrangeiras para as demais e dar continuidade ao processo de geração de código. Será evocado por *PrintPrecedenceB* o *template PrintEntityWithForeign*, que evoca os mesmos *templates* que *PrintEntityWithoutForeign*, com o acréscimo do *template PrintForeignKey*, que irá imprimir o código das chaves estrangeiras que a entidade possua.

O *template CreateAssociativeTable* tem a função de verificar se, dentre os relacionamentos existentes na modelagem, algum deles dará origem a uma tabela associativa na base de dados. Essa verificação é feita utilizando o atributo booleano presente na classe *Relationship* chamado *isAssociative*. No momento da modelagem, caso o projetista identifique a necessidade da geração da tabela associativa a partir do relacionamento, ele deverá setar esse atributo como *True*. O *template* dará inicio a geração do código dessas tabelas, evocando os *templates PrintRelationCharacteristics*, *PrintRelationFields*, *PrintPrimary* e *PrintForeignKeyFieldsN_N*.

O *template PrintRelationCharacteristics* é responsável por verificar as entidades envolvidas na relação e evocar para cada uma delas o *template PrintCandidateKey*, que tem a finalidade de verificar as chaves primárias da entidade em questão e imprimir esses atributos e seus respectivos valores, através da evocação do *template ToField*. O *template PrintRelationFields* tem a finalidade de imprimir os atributos da relação evocando o *template ToField*. Dessa forma, os atributos da tabela associativa também serão impressos no código DDL.

O *template PrintForeignKeyFieldsN_N* foi definido para gerar o código referente as chaves estrangeiras, que apontam para as entidades envolvidas na tabela associativa. Para cada chave estrangeira presente na tabela associativa, esse *template* irá imprimir o código com o auxílio do *template PrintReferences*, que tem a função de completar com os dados necessários das chaves primárias o código gerado das chaves estrangeiras. Na figura 4 pode ser visto um exemplo de *template* para geração dos tipos geográficos.

3. Prova de Conceito

Nesta seção será apresentado o simples exemplo utilizando a metodologia extendida através de uma ferramenta MDA, que é um conjunto de *plug-ins* para a plataforma de desenvolvimento Eclipse, onde o meta-modelo foi implementado utilizando o Ecore do *Eclipse Modeling Framework* (EMF) [Steinberg et al. 2008] e as regras de transformação foram implementadas utilizando o Acceleo [Obeo 2012], que é uma ferramenta para geração de artefatos de texto que segue a especificação M2T da OMG.

```
[comment encoding = UTF-8 /]
[module PrintGeograficEntity('http://br.cefet.rj.mda.gedbm')]

[template public PrintGeograficEntity(aEntity : Entity)]
[if (aEntity.type.toString() = 'NetworkClass' or aEntity.type.toString() = 'AdjacentPolygons' or aEntity.type.toString() = 'Tesselation'  or
aEntity.type.toString() = 'Sampling' or aEntity.type.toString() = 'Isoline')]
    [aEntity.name/] Multipoint,
[/if]
[if (aEntity.type.toString() = 'Polygon' or aEntity.type.toString() = 'Tesselation' )]
    [aEntity.name/] Polygon,
[/if]
[if (aEntity.type.toString() = 'Point' or aEntity.type.toString() = 'Node')]
    [aEntity.name/] Point,
[/if]
[if (aEntity.type.toString() = 'Line' or aEntity.type.toString() = 'UnidirectionalLine')]
    [aEntity.name/] LineString,
[/if]
[if (aEntity.type.toString() = 'BidirectionalLine')]
    [aEntity.name/] MultiLinestring,
[/if]
[/template]
```

Figure 4. Exemplo de templates em M2T.

O exemplo consiste de um trecho de uma modelagem, em OMT-G, de um banco de dados geográfico que é responsável por controlar os logradouros de municípios [Laender et al. 2005]. A modelagem pode ser vista na figura 5.

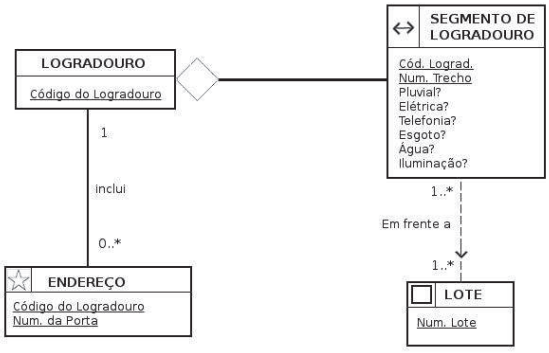

Figure 5. Exemplo de modelagem conceitual geográfica.

O modelo deve ser instanciado no Ecore, com base nas construções observadas no meta-modelo da metodologia. Cada entidade e seus atributos devem ser instanciados conforme o exemplo da figura 6.

Em seguida, as transformações devem ser executadas para que o código SFS/SQL possa ser gerado automaticamente. Ao executar a transformação, as instâncias das classes do meta-modelo são utilizadas como parâmetros nos templates do Acceleo, que são executados em cascata para a geração do código. O resultado da transformação de modelo para texto pode ser vista na figura 7.

O código SQL gerado foi testado e não apresentou nenhum erro quando executado no SGBD MySQL. No entanto, quando foi executado no PostgreSQL, que utiliza a extensão PostGIS para poder manipular dados do tipo geográfico, houve erro e não foi possível criar o banco de dados. Tal erro ocorre pelo fato do PostgreSQL utilizar uma função, *AddGeometryColumn*, para criar tipos de dados geográfico em suas tabelas. Na utilização do SGBD *Oracle Spatial* também ocorreu um erro, isto porque o *Oracle Spatial* utiliza o tipo MDSYS.SDO_GEOMETRY para todo campo geográfico criado em uma tabela.

A solução para os problemas apresentados seriam a criação de novos cartuchos

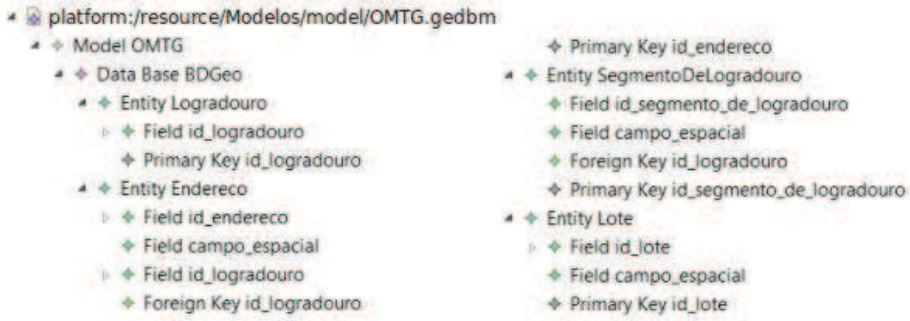

Figure 6. A instância do modelo.

contendo regras de transformação que tratariam as especificidades de cada SGBD, permitindo que seja gerado código SQL específico para cada um desses sistemas. Outra solução possível seria a intervenção manual no arquivo de texto gerado, sendo necessário para adaptá-lo para o Oracle Spatial apenas alterar o tipo dos campos geográficos gerados para MDSYS.SDO_GEOMETRY.

```
CREATE DATABASE BDGeo;

CREATE TABLE BDGeo.Logradouro (
    id_logradouro int(0) ,
    CONSTRAINT id_logradouro PRIMARY KEY (id_logradouro)
);

CREATE TABLE BDGeo.Lote (
    id_lote int(0) ,
    campo_espacial POLYGON,
    CONSTRAINT id_lote PRIMARY KEY (id_lote)
);

CREATE TABLE BDGeo.Endereco (
    id_endereco int(0) ,
    campo_espacial POINT,
    id_logradouro int(0) ,
    CONSTRAINT id_endereco PRIMARY KEY (id_endereco),
    CONSTRAINT FK_id_logradouro FOREIGN KEY (id_logradouro)
          REFERENCES Logradouro (id_logradouro)
);
```

```
CREATE TABLE BDGeo.SegmentoDeLogradouro (
    id_segmento_de_logradouro int,
    campo_espacial LINESTRING,
    id_logradouro int(0) ,
    CONSTRAINT id_segmento_de_logradouro PRIMARY KEY
        (id_segmento_de_logradouro),
    CONSTRAINT FK_id_logradouro2 FOREIGN KEY
        (id_logradouro) REFERENCES Logradouro (id_logradouro)
);

CREATE TABLE BDGeo.Lot_SDL (
    id_lot_sdl int(0) ,
    id_lote int(0) ,
    id_segmento_de_logradouro int(0) ,
    CONSTRAINT id_lot_sdl PRIMARY KEY (id_lot_sdl),
    CONSTRAINT FK id lote FOREIGN KEY (id lote) REFERENCES
        Lote (id lote),
    CONSTRAINT FK_id_segmento_de_logradouro FOREIGN KEY
        (id_segmento_de_logradouro)
    REFERENCES SegmentoDeLogradouro (id_segmento_de_logradouro)
);
```

Figure 7. O código gerado.

4. Trabalhos Relacionados

Nesta seção são apresentados alguns trabalhos relacionados que utilizam o MDA em conjunto com a modelagem de banco de dados geográficos e a geração de codificação automática. O ArgoCASEGEO + TerraLib [Gazola et al. 2006] é uma integração entre o ArgoCASEGEO, que é uma ferramenta CASE para projetos de banco de dados que utiliza o modelo UML-GeoFrame [Lisboa and Iochpe 1999] e a biblioteca TerraLib, que é codificada em C++ e é específica para Small GIS. A integração utiliza a MDA através da utilização da extensão da UML para especificação de modelos independentes de plataformas e a geração da codificação automática é dada através de um processo através de uma arquitetura de três camadas.

A integração, apesar de utilizar o modelo UML-GeoFrame como meta-modelo de estereótipos geográficos; gerar códigos para diversas plataformas; e gerar codificação também para projeto de banco de dados relacionais, ela possui um conjunto de regras única que utiliza um *parser* para a geração da codificação. A ferramenta ainda possui uma arquitetura de geração de código com três camadas que depende do ArgoCASEGEO, que

consequentemente está atrelada ao UML-GeoFrame, dificultando o processo de extensão para um outro modelo, e.g. OMT-G.

A metodologia utilizada neste artigo utiliza uma especificação padronizada pela OMG como regras de transformação, a M2T, e cartuchos de regras que podem ser inseridos, sem a necessidade de se modificar regras já existentes. Ainda, não existe o atrelamento a nenhum modelo ou linguagem de modelagem, visto que cada modelo MDA da metodologia é implementado separadamente.

Já o OMT-G Design [Schaly and Frozza 2010] é uma ferramenta de apoio a projeto de banco de dados geográficos, que utiliza o modelo OMT-G, para o ambiente de desenvolvimento Eclipse. A ferramenta utiliza o *Graphical Modeling Framework*, para geração da parte gráfica e o *Xpand*, que é uma linguagem baseada em templates M2T, para geração de codificação automática do esquema do banco de dados. Apesar da ferramenta utilizar a metodologia MDA, a ferramenta está atrelada ao modelo conceitual OMT-G e não utiliza um meta-modelo genérico.

A extensão proposta nesse artigo, permite que seja gerada a codificação automática para o esquema do banco de dados, sem a necessidade de se refazer ou criar uma nova ferramenta, desde que a linguagem de modelagem ou modelo escolhido seja aderente ao meta-modelo da metodologia utilizada na ferramenta. Isso impede o atrelamento à um único modelo ou linguagem, oferecendo flexibilidade ao projetista de banco de dados.

5. Conclusão

Este artigo apresentou uma extensão geográfica para uma metodologia MDA de projeto de banco de dados relacionais que gera codificação automática para a especificação SFS/SQL e para o padrão ANSI SQL 92/99/03. Para que fosse possível estender a metodologia foi proposta uma modificação no meta-modelo genérico original, onde foi necessário estender os tipos dos campos para ser adaptável ao SFS/SQL. Da mesma forma, foi apresentada as transformações M2T para a geração de codificação automática.

Foi apresentado também uma ferramenta para o ambiente de desenvolvimento Eclipse utilizando o EMF, onde o meta-modelo modificado foi implementado usando o Ecore e as regras de transformação foram implementadas usando o Acceleo. Um simples exemplo modelado usando o OMT-G, para um banco de dados geográfico de municípios foi instanciado na ferramenta para a geração do código SFS/SQL.

A metodologia flexibiliza as tarefas do projetista de banco de dados pois permite a escolha entre diversas linguagens de modelagens relacionais e geográficas. Além disso, por utilizar a MDA, o meta-modelo pode ser extensível a outros modelos, sem a necessidade de se refazer as regras de transformação para a codificação automática. Também é possível adicionar novos cartuchos de geração de código para outras especificações e manter a metodologia flexível para diversos modelos.

A ferramenta desenvolvida, apesar de não possuir uma parte gráfica, pode ser acoplada a outras ferramentas já desenvolvidas, gerando um conjunto de transformações do modelo específico para o meta-modelo adotado pela ferramenta proposta. Também é possível criar a parte gráfica utilizando o *Graphical Modeling Framework* (GMF), integrante do EMF, usado como base de desenvolvimento da ferramenta, visto que eles

compartilham o mesmo meta-modelo. Uma outra vantagem da ferramenta desenvolvida é que ela foi pensada para utilização em plataformas livres e utiliza de diversas tecnologias livres em seu desenvolvimento, dessa forma, a ferramenta desenvolvida também segue os princípios do software livre e é, portanto, gratuita.

Como trabalhos futuros será desenvolvido a parte gráfica para as linguagens de modelagem relacional e a integração com uma ferramenta existente para modelagem geográfica. Dois conjuntos de transformação utilizando o QVT devem ser descritos para que os conceitos sejam mapeados do modelo escolhido para o meta-modelo utilizado na metodologia.

References

Booch, G., Rumbaugh, J., and Jacobson, I. (2000). *UML: Guia do Usuário*. Editora Campus.

Elmasri, R. and Navathe, S. (2005). *Sistemas de banco de dados*. Pearson Addison Wesley.

Gazola, A., Sampaio, G. B., and Filho, J. L. (2006). Argocasegeo + terralib: Bancos de dados geográficos para aplicações small gis. In *I Workshop de Computação e Aplicações. Anais do XXVI Congresso da SBC*, volume 1.

Laender, A., Davis, C., Brauner, D., Câmara, G., Queiroz, G., Borges, K., Ferreira, K., Ligiane, V. L., and Carvalho, M. (2005). *Bancos de Dados Geográficos*. MundoGEO.

Lisboa, F. and Iochpe, C. (1999). Specifying analysis patterns for geographic databases on the basis of a conceptual framework. In *Proceedings of the 7th ACM GIS*.

Mellor, S. J., Scott, K., Uhl, A., and Weise, D. (2005). *MDA Destilada: Princípios de Arquitetura Orientada por Modelos*. Ciência Moderna.

Obeo (2012). Acceleo: MDA generator - home. http://www.acceleo.org/.

OMG (2008). MOF model to text transformation language, v 1.0.

OMG (2011). Meta object facility (MOF) Query/View/Transfomation specification.

Rosa, A., Gonçalves, I., and Pantoja, C. E. (2013). A mda approach for database modeling. *Lecture Notes on Software Engineering*, 1(1):26–30.

Rosa, A. and Pantoja, C. E. (2013). Uma ferramenta mda para modelagem de banco de dados relacionais. In *IX Escola Regional de Banco de Dados*.

Schaly, K. W. and Frozza, A. A. (2010). Uma ferramenta para gerar bancos de dados geográficos a partir de diagramas omt-g. In *XI Escola Regional de Banco de Dados*.

Steinberg, D., Budinsky, F., Merks, E., and Paternostro, M. (2008). *Emf: Eclipse Modeling Framework*. Pearson Education.