Pós-Graduação em Ciência da Computação

“Uma Abordagem MDD para Prover Integridade Topológica e de Rede em Projeto Conceitual de Banco de Dados Espaciais”

Por

Jones Cavalcanti Sarmento

Dissertação de Mestrado

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

RECIFE
2015

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JONES CAVALCANTI SARMENTO

“Uma Abordagem MDD para Prover Integridade Topológica e de Rede em Projeto Conceitual de Banco de Dados Espaciais"

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
2015

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S246a Sarmento, Jones Cavalcanti
Uma abordagem MDD para prover integridade topológica e de rede em projeto conceitual de banco de dados espaciais / Jones Cavalcanti Sarmento. – Recife: O Autor, 2015.
87 f.: il., fig.

Orientador: Robson do Nascimento Fidalgo.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2015.
Inclui referências e apêndices.

1. Banco de dados. 2. Banco de dados geográfico. 3. Modelagem conceitual. I. Fidalgo, Robson do Nascimento (orientador). II. Título.

025.04 CDD (23. ed.) UFPE- MEI 2016-016

Dissertação de Mestrado apresentada por **Jones Cavalcanti Sarmento** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Uma Abordagem MDD para Prover Integridade Topológica e de Rede em Projeto Conceitual de Banco de Dados Espaciais**" orientada pelo **Prof. Robson do Nascimento Fidalgo** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Ana Carolina Brandão Salgado
Centro de Informática / UFPE

_____

Prof. Caudio de Souza Baptista
Departamento de Sistemas e Computação / UFCG

_____

Prof. Robson do Nascimento Fidalgo
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 28 de agosto de 2015.

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Agradecimentos

Ao Senhor Jesus, O Cristo, a honra deste trabalho.

À minha família: dona Leci Maria (mãe), Cristhyane Lima (irmã) e Isabel Sarmento (filha).

À minha querida noiva, Escarlat Luiza.

Aos familiares “Sarmentos” e “Cavalcantis”.

Ao professor Robson Fidalgo pela oportunidade, confiança, disponibilidade e motivação.

Aos meus amigos: Edson Alves, Natália Franco e Antônio Josivaldo.

Aos funcionários do Centro de Informática/UFPE: professores, administração, limpeza, manutenção e suporte técnico.

À Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (Facepe), pelo incentivo financeiro.

E a todos que participaram desse importante período da minha vida, muito obrigado!

Resumo

Model-Driven Development (MDD) é um paradigma que usa modelos como o principal artefato no processo de desenvolvimento de sistemas. Isto é, neste paradigma, modelos não são apenas artefatos de documentação, pois devem corresponder a códigos executáveis. Em projetos de Banco de Dados Espaciais (BDE), existem várias linguagens de modelagens (e.g., OMT-G, MADS, GeoProfile e UML-GeoFrame), as quais permitem representar características espaciais (e.g., Múltiplas Representações e Relacionamento Espacial) por meio de uma ferramenta do tipo *Computer-Aided Software Engineering* (CASE). Embora essas linguagens sejam bem exploradas e difundidas na literatura, constatou-se que estas têm deficiências para modelar e implementar integridade espacial a partir de entidades com múltiplas representações. Assim, de modo a avançar o estado da arte sobre projeto de BDE, este trabalho faz uma análise dos principais trabalhos relacionados, e, de modo a contribuir para superar as deficiências encontradas, propõe uma extensão espacial para a linguagem de modelagem *Enhanced Entity Relationship* (EER). Essa extensão é implementada na ferramenta EERCASE e avalidada por meio de uma análise comparativa com os principais trabalhos relacionados, evidenciando seus pontos fortes e fracos.

Palavras-Chave: Modelagem de Banco de Dados Espaciais. Metamodelo. MDD. Ferramenta CASE.

Abstract

Model-Driven Development (MDD) is a paradigm that uses models as the primary artifact in the systems development process. That is, in this paradigm, models are not only documentation artifacts, since these should be the executable code. In Spatial Databases projects (SDB), there are several modeling languages (e.g., OMT-G, MADS, GeoProfile and UML-GeoFrame), which allow to represent spatial characteristics (e.g., Multiple Representations and Spatial Relationship) by means of a tool type Computer-Aided Software Engineering (CASE). Although these languages are better exploited and widespread in the literature, it was found that they have deficiencies in modeling and implement spatial integrity from entities with multiple representations. Thus, in order to advance the state of the art in SDB project, this paper analyzes the main works related and, in order to contribute to overcome the deficiencies, proposes a spatial extension for modeling language Enhanced Entity Relationship (EER). This extension is implemented in tool EERCASE and evaluated through a comparison with the main work related, highlighting their strengths and weaknesses.

Key Words: Spatial Database Modeling. Metamodel. MDD. CASE tool.

Lista de Figuras

Lista de Quadros

Principais Acrônimos

BDE	Banco de Dados Espaciais
CASE	Computer-Aided Software Engineering
CIM	Computation Independent Model
EER	Enhanced Entity Relationship
EVL	Epsilon Validation Language
GPLv2	GNU General Public License
MADS	Modeling of Application Data with Spatio-temporal features
MDA	Model Driven Architecture
MDD	Model Driven Development
OCL	Object Constraint Language
OMG	Object Management Group
OMT-G	Object Modeling Technique - Geographic
PIM	Platform Independent Model
PSM	Platform Specific Model
QVT	Query View Transformation
SGBD-E	Sistema Gerenciador de Banco de Dados Espacial
SQL	Structured Query Language
SRID	Spatial Reference Identifier
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

Sumário

1. Introdução

Neste Capítulo, apresentam-se o contexto e os motivos que levaram ao desenvolvimento da presente dissertação, assim como seus objetivos e a estrutura de organização dos demais Capítulos.

1.1 Apresentação

Model Driven Development (MDD) (BRAMBILLA; CABOT; WIMMER, 2012) é um paradigma de desenvolvimento de sistemas que tem Modelos como artefatos essenciais para subsidiar a geração de código executável ou interpretável. MDD utiliza ferramentas do tipo *Computer-Aided Software Engineering* (CASE) (ALVES et al., 2014) para produzir seus artefatos a partir de interpretações definidas por meio de uma linguagem de modelagem (e,g. *Enhanced Entity Relationship* – EER e *Unified Modeling Language* - UML).

No contexto de projetos de Banco de Dados Espaciais (BDE) (CÂMARA, 1996; CASANOVA et al., 2005) um modelo é um esquema visual que contém um conjunto de construtores diagramáticos com características espaciais, tais como: tipos de dados espaciais, relacionamentos topológicos e modelagem de rede. Por exemplo, o relacionamento topológico "*IN*" ou "CONTIDO", envolvendo as geometrias Ponto e Polígono, pode ser verificado na relação: um endereço contido em um bairro. Vale ressaltar que um relacionamento espacial (CASANOVA et al., 2005) é uma associação estrutural baseada em integridade referencial (ELMASRI e NAVATHE, 2011), que impõe uma condição espacial entre dois objetos geográficos. Isto é, tais objetos só podem ser associados se a integridade referencial for garantida e a condição espacial, satisfeita.

1.2 Motivação

Embora existam linguagens e ferramentas CASE para modelagem de BDE que oferecem um amplo suporte notacional e tecnológico, se desconhece a existência de uma que: 1) permita personalizar os nomes dos atributos espaciais (e.g., dar os nomes "marco_zero" e "prefeitura" para dois atributos do

tipo ponto em uma mesma classe/entidade “Município)”; 2) modele, em uma mesma classe/entidade, mais de um atributo espacial do mesmo tipo (cf. exemplo anterior); 3) ilustre, via pictogramas distintos, quais são os tipos de dados que participam de uma operação espacial (e.g., o pictograma “Ponto *IN* Polígono” deve ser diferente do pictograma “Polígono *IN* Polígono”); 4) represente, sem ambiguidade, quais são os atributos espaciais que devem participar de cada operação espacial (e.g., deixar claro se a geometria deve ser o objeto do tipo “marco zero” ou do tipo “prefeitura”) ; 5) permita capturar todas as possíveis direções para os segmentos de uma rede (cf. Capítulo 2); e 6) impeça a especificação de relacionamentos topológico ou modelagem de rede inválidos (e.g., “Polígono *IN* Ponto”).

Por fim, outro fator que motivou o desenvolvimento deste trabalho é o fato de que a linguagem de modelagem EER é tão expressiva quanto a UML e bem aceita pela comunidade de banco de dados, no entanto, não foi encontrado nenhum trabalho recente - com menos de 10 anos – o último foi de 1999 (TRYFONA e JENSEN, 1999; TRYFONA et al., 1999), o qual tem as mesmas limitações mencionadas anteriormente.

1.3 Objetivos

O objetivo geral deste trabalho é, com base no paradigma de MDD, definir uma extensão espacial para a linguagem EER e implementar um protótipo de uma ferramenta CASE, para prover integridade topológica e de rede em projeto conceitual de BDE. Para os objetivos específicos, têm-se os seguintes:

1. Identificar quais são os principais tipos de dados espaciais e propor pictogramas não ambíguos e representativos;
2. Identificar quais são as principais operações topológicas e de rede, e também propor pictogramas não ambíguos e representativos;
3. Estender o *Enhanced Entity Relationship Metamodel* (EERMM) (cf. Capítulo 2) de modo que o mesmo passe a dar suporte a atributos e relacionamentos espaciais;
4. Definir um conjunto de regras básicas para impedir a especificação equivocada de relacionamentos espaciais;

5. Utilizar tecnologias baseadas em MDD para implementar um protótipo de ferramenta CASE que permita mostrar que o trabalho proposto é factível.

1.4 Estrutura da Dissertação

Os demais capítulos desta dissertação estão organizados da seguinte forma: no Capítulo 2, apresentam-se os conceitos básicos que estão relacionados a esta pesquisa. No Capítulo 3, as propostas que estão relacionadas ao contexto dessa pesquisa são avaliadas. No Capítulo 4, uma extensão espacial para a linguagem EER é proposta. No Capítulo 5, apresenta-se um protótipo de ferramenta CASE que implementa a linguagem definida neste trabalho e compara-se o trabalho proposto com os demais trabalhos relacionados. Por fim, no Capítulo 6, apresentam-se as considerações finais, contribuições e trabalhos futuros.

2. Conceitos Básicos

Neste Capítulo, são apresentados os conceitos básicos necessários para um melhor entendimento da presente dissertação. Tais conceitos são divididos em: conceitos de cartografia, modelagem conceitual de BDE, Desenvolvimento Dirigido por Modelos, como também o conceito de Integridade Espacial.

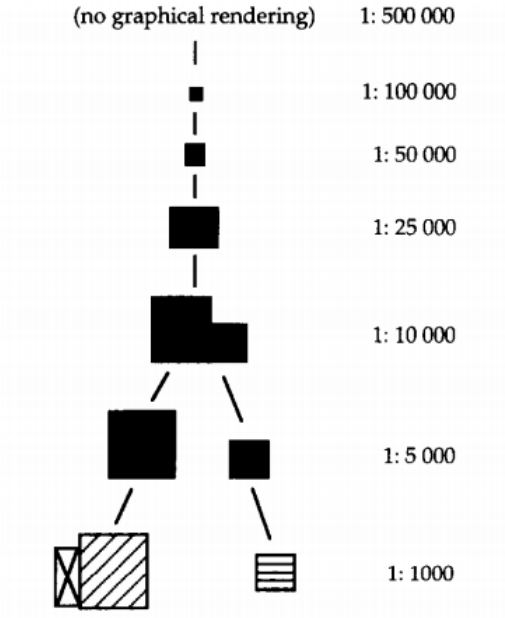
2.1 Conceitos Básicos de Cartografia

Nesta Seção mostram-se os conceitos de Escala e Identificador de Sistema de Referência Espacial (*Spatial Reference System Identifier* - SRID), uma vez que apenas estes são importantes para o entendimento deste trabalho. Contudo, os demais conceitos sobre cartografia (e.g., Elipsóide, Datum e Sistemas de Coordenadas Geográficas) podem ser consultados em (LONGLEY et al., 2013).

2.1.1 Escala e SRID

Segundo Laurini e Thompson (LAURINI e THOMPSON, 1992) escala é uma relação de equivalência entre distâncias impressas em um mapa e as distâncias equivalentes medidas na superfície da terra.

Figura 1 – Representação de Objetos em Diferentes Escalas



Fonte: (FRANK; TIMPF, 1994)

Por exemplo, uma escala de 1:1.000 significa que 1 centímetro do mapa equivale a 1.000 metros na superfície da terra. Na Figura 1, ilustra-se um exemplo de objetos vistos em escalas diferentes. Note que, quanto maior a escala (e.g., 1:1.000 > 1:500.000), os objetos aparecem com mais detalhes.

Um SRID é um identificador que especifica o sistema de coordenadas no qual um objeto espacial é representado. Sem essa identificação, não é possível projetar os dados em um Sistema de Informações Geográficas (SIG). Caso não seja configurado um SRID para um objeto espacial, o Sistema de Gerenciamento de Banco de Dados Espaciais (SGBDE) pode usar um valor padrão (e.g., o PostgreSQL/PostGis usa o valor -1 que significa SRID desconhecido). Ressalta-se, inclusive, que esta não é uma boa prática, visto que a ausência do SRID pode produzir resultados e/ou visualizações inconsistentes. Ademais, dois objetos só podem ser comparados se possuírem o mesmo SRID (OBE e HSU, 2011).

2.2 Modelagem Conceitual de Banco de Dados Espaciais

A modelagem conceitual de banco de dados espaciais é o desenvolvimento de modelos abstratos por meio de uma linguagem de modelagem (e.g., EER ou UML) que contenha construtores espaciais. Estes construtores representam as diferentes formas como o espaço é observado pelo usuário, variando o tipo de dado, a visão, a representação e as relações entre as entidades observadas (CASANOVA et al., 2005). Nesta Seção, cada um dos conceitos citados é descrito, como também a modelagem para estruturas de Redes.

2.2.1 Visões e Tipos de Dados Espaciais

Os atributos de uma entidade definem suas características, como por exemplo: uma entidade “carro” tem como atributos: cor, ano, data de fabricação, etc. Tais atributos podem ser valorados como: “Vermelho” (i.e., tipo caractere) para cor, “2015” (i.e., tipo numérico) para ano, e, por fim “25/06/2015” (i.e., tipo data) para data de fabricação (HEUSER, 2001). Para atributos espaciais, pode-se citar a localização geográfica da fábrica do carro, o

trajeto percorrido, ou o país onde foi fabricado. Estes atributos podem ser capturados como Ponto, Linha e Polígono, respectivamente (LONGLEY et al., 2013).

A utilização dos tipos espaciais depende da percepção do projetista, pois a modelagem de um BDE deve refletir as particularidades do espaço a ser modelado. Tal percepção pode ser modelada por duas visões (i.e., Visão de Campo e Visão de Objetos). Para representar a Visão de Campo, utilizam-se os seguintes elementos: Pontos Irregulares, Isolinhas, Polígonos Adjacentes, Grade de Células, Grade de Pontos e Rede Triangular Irregular (TIN) (GOODCHILD, 1992; LONGLEY et al., 2013). Estes elementos, para serem armazenados em um BDE, precisam ser mapeados em tipos de dados não convencionais. No Quadro 1, mostra-se um exemplo deste mapeamento utilizando o SGBDE PostgreSQL/PostGis.

Quadro 1 – Mapeamento de Visão de Campo para SGBDE

Característica na Visão de Campo	**Tipo de Dado em SGBD-E**
Pontos Irregulares	*Point* ou *Multipoint*
Isolinha	*Linestring* ou *Multilinestring*
Polígonos Adjacentes	*Polygon* ou *Multipolygon*
Grade de Células	Raster
Grade de Pontos	Raster ou *Point* ou *Multipoint*
TIN	*Linestring* ou *Multilinestring*

Fonte: (KEMP, 1993) adaptado

Na Visão de Objeto a realidade é vista por entidades reais, e localizadas por latitude e longitude no espaço geográfico. Seus dados são definidos, em um SGBDE, como por exemplo: Ponto (*Point*), Linha (*Linestring*), *Polygon* (Polygon), Multiponto (*Multipoint*), Multilinha (*Multiline*), Multipolígono (*Multipolygon*) e Complexo (*Geometrycollection*). No Quadro 2, mostra-se um exemplo de mapeamento da visão de objetos em SGBDE PostgreSQL/PostGis.

Quadro 2 – Mapeamento de Visão de Objeto para SGBDE

Característica na Visão de Objeto	Tipo de Dado em SGBD-E
Ponto	*Point*
Coleção de Pontos	*Multipoint*
Linha	*Linestring*
Coleção de Linhas	*Multilinestring*
Polígono	*Polygon*
Coleção de Polígonos	*Multipolygon*
Coleção de Vários Tipos	*Geometrycollection*

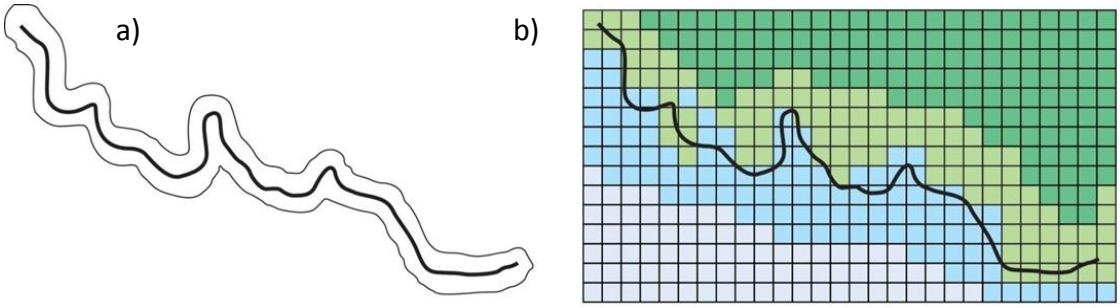
Fonte: (OBE e HSU, 2011) adaptado

A visão de objetos é utilizada para representar objetos espaciais bem definidos. Ou seja, que represente um objeto do mundo real, com formato e localização explícitos. Por exemplo, uma casa, com formato de polígono e localização por coordenadas geográficas. Ao contrário da visão de objetos, a visão de campo não possui, necessariamente, um formato ou localização no espaço, antes, são utilizadas para representar variações no espaço, tais como: temperatura, relevo e poluição atmosférica (LONGLEY et al., 2013).

2.2.2 Múltiplas Representações Espaciais

A representação de um objeto espacial pode ser feita de forma única ou múltipla. Por exemplo, um município pode ser representado por um ponto (i.e., o centroide ou a prefeitura), por um polígono (i.e., os limites do município) ou pelas duas representações ao mesmo tempo. Neste último caso, caracteriza-se uma Múltipla Representação que pode ser usada para representar a variação da escala (i.e., em uma escala baixa é representado por ponto, e em escala alta por polígono) ou por diferentes formatos (i.e., um rio representado por uma linha, ou por um polígono) (CASANOVA et al., 2005). A múltipla representação, embora seja necessária em muitos casos, pode apresentar alguns problemas, tais como: redundância, inconsistências e comportamentos múltiplos para um mesmo fenômeno (CÂMARA, 1996). Um exemplo de múltiplas representações pode ser visto na Figura 2.

Figura 2 - Múltiplas Representações de um Rio



Fonte: (LONGLEY et al., 2013)

Na Figura 2, verificam-se duas visões acerca da realidade geográfica (i.e., Visão de Objeto “Figura 2a” e Visão de Campo “Figura 2b”) com múltiplas representações cada uma. Na Figura 2a, é visto um rio modelado como uma Linha e seu contorno representado por um Polígono. Na Figura 2b um rio é representado por um tipo Grade de Células (i.e., Raster) sendo representado também por uma Linha como o centro do rio.

2.2.3 Relacionamentos Espaciais

Um relacionamento espacial permite impor integridade referencial (ELMASRI e NAVATHE, 2011) a partir da avaliação de uma condição/operação espacial que é aplicada entre dois objetos geográficos (CASANOVA et al., 2005). Segundo Rigaux (RIGAUX; SCHOLL; VOISARD, 2001) uma operação espacial é classificada levando em consideração o tipo de retorno e a quantidade de objetos envolvidos na operação. No Quadro 3, verifica-se esta classificação, vista também em (CASANOVA et al., 2005).

Quadro 3 – Classificação das Operações Espaciais

Operação	Exemplo
Unária Booleana	Convexo, Conectado
Unária Escalar	Comprimento, Área
Unária Espacial	Buffer, Centroide
Binária Booleana	Contém, Cruza (Topológicos); Acima, Ao Norte (Direcionais); Distância Entre (Métrico)
Binária Escalar	Distância
Binária Espacial	Interseção, Diferença
N-ária Espacial	União

Fonte: (RIGAUX; SCHOLL; VOISARD, 2001) adaptado

Segundo a classificação vista no Quadro 3, as operações binárias booleanas são subdivididas em: Topológicas, Métricas e Direcionais. Para fins desta dissertação, serão utilizadas apenas as Topológicas. No Quadro 4, mostram-se pictogramas para representar os relacionamentos espaciais topológicos, definidos em (PARENT et al., 1998) para utilização em linguagem de modelagem.

Quadro 4 - Relacionamentos Topológicos em Linguagem de Modelagem

Tipo Espacial	Ícone	Definição
Disjunto		Os objetos conectados têm geometrias espacialmente disjuntas
Toca		Compartilhamento de geometrias sem interiores em comum
Cruza		Compartilha de alguma parte do interior, de tal forma que a dimensão da parte compartilhada é estritamente inferior à dimensão maior dos objetos conectados
Sobrepõe		Compartilha de alguma parte do interior, de tal forma que a dimensão da parte compartilhada é igual à dimensão dos objetos conectados
Contido		Todo interior de um objeto é parte do interior do outro objeto
Igual		Compartilha de todo interior e de todo o contorno (válido para objetos geográficos com mesma dimensão)

Fonte: (PARENT et al., 1998) traduzido

Nos trabalhos de Egenhofer, Franzosa, Herring (EGENHOFER e FRANZOSA, 1991; EGENHOFER e HERRING, 1990) e de Clementini, Di

Felice, Van Oosterom (CLEMENTINI; DI FELICE; VAN OOSTEROM, 1993) são vistas definições para operações topológicas válidas. Tais operações são agrupadas pelo segundo trabalho, formando um conjunto mínimo de operações topológicas entre Ponto, Linha e Polígono. Este conjunto é formado pelas operações: Toca, Contido, Cruza, Sobrepõe e Disjunto. Vários trabalhos os têm como base para definição de seus relacionamentos espaciais (BORGES; DAVIS; LAENDER, 2001; LISBOA-FILHO et al., 2013; PARENT et al., 1998; RIBEIRO et al., 2013).

Clementini, Di Felice e Van Oosterom definem um conjunto mínimo de vinte e um casos de operações válidas entre Polígono (identificado pela letra “A” de área), Linha (identificado pela letra “L”) e Ponto (identificado pela letra “P”). Esses casos e as respectivas operações definidas são:

- Contido: AA, LL, LA, PA, PL, PP;
- Disjunto: AA, LL, LA, PA, PL, PP;
- Toca: AA, LL, LA, PA, PL;
- Sobrepõe: AA, LL;
- Cruza: LL, LA;

Embora Clementini observe que os casos envolvendo a operação “Igual” podem ser obtidos pela operação “Contém/Contido”, esta dissertação entende que estas operações possuem semânticas diferentes. Com isso, os casos envolvendo esta operação, são: PP, AA e LL; que somados aos vinte e um anteriores, totalizam vinte e quatro casos. Outros dezoito casos são derivados do tipo matricial “Raster”, que embora não seja observado nos trabalhos anteriormente citados, pode ser comparado às geometrias vetoriais (ponto, linha, polígono) por seus limites em forma de Polígono (OBE e HSU, 2011). Sendo assim, são adicionados dezoito casos envolvendo o tipo “Raster”, que são mostrados a seguir pela letra “R”:

- Contido: RA, AR, LR, PR, RR;
- Disjunto: RA, LR, PR, RR;
- Toca: RA, RP, RL, RR;
- Sobrepõe: RA, RR;

- Cruza: RL;
- Igual: RA, RR;

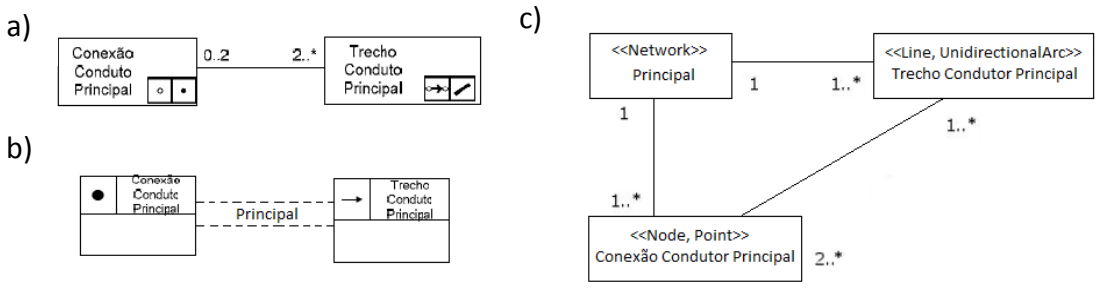
Os dezoito casos envolvendo o tipo “Raster” somados aos vinte e quatro vistos anteriormente, totalizam quarenta e dois casos. Vale ressaltar que, os casos envolvendo a operação “Contido” precisam ser direcionados, por exemplo: um ponto pode estar contido em um polígono, mas um polígono não pode estar contido em um ponto. Esses casos são: LA (i.e., Linha Contido em Polígono), PA (i.e., Ponto Contido em Polígono), PL (i.e., Ponto Contido em Linha), LR (i.e., Linha Contido em Raster), PR (i.e., Ponto Contido em Raster). Os casos envolvendo os tipos Raster e Polígono para operação “Contido” são: RA (i.e., Raster Contido em Polígono) e AR (Polígono Contido em Raster).

2.2.4 Modelagem de Redes

Uma estrutura de Rede pode ser formada por Nós e Arcos ou apenas Arcos. Em uma rede de esgoto, por exemplo, os nós representam os bueiros e os arcos representam a ligação entre dois desses bueiros. Os Arcos de uma rede podem ser direcionados, dependendo da necessidade do objeto da modelagem. A modelagem de Redes é feita a partir de elementos que representem esses dois conceitos (i.e., Nó e Arco) (STEMPLIUC et al., 2009).

Algumas linguagens de modelagem dão suporte à modelagem de Redes, como por exemplo: UML-GeoFrame (Figura 3a), OMT-G (Figura 3b), e GeoProfile (Figura 3c).

Figura 3 - Modelagens de Rede



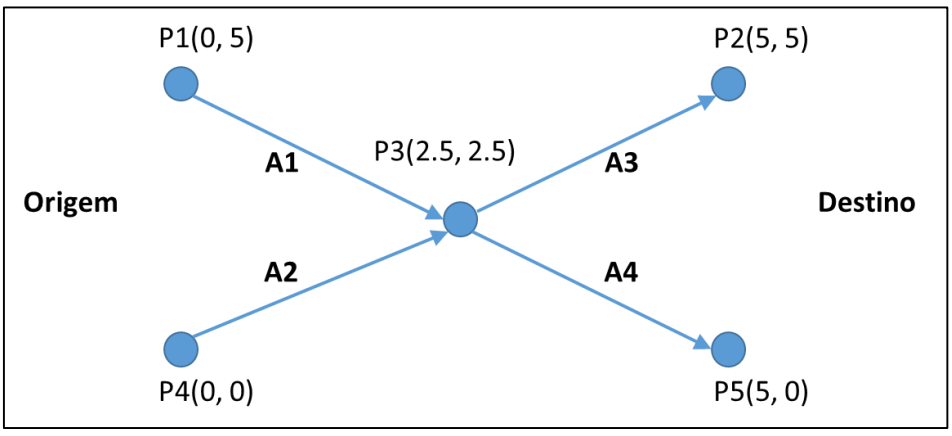
Fonte: (a) (STEMPLIUC et al., 2009), (b) (BORGES; DAVIS; LAENDER, 2001), e (c) (SAMPAIO, 2009) adaptado

Na Figura 3, ilustram-se três modelagens de uma mesma rede (i.e., Principal). Cada arco da rede corresponde a um trecho (i.e., Condutor Principal) e cada nó da rede representa uma conexão entre os trechos.

Uma Rede pode ser mapeada para um BDE. Neste caso, um trecho corresponde a um tipo Linha e um nó a um tipo Ponto. Caso o BDE não ofereça funcionalidades específicas para manipular redes, pode usar uma extensão espacial com estas funcionalidades. Neste trabalho, utiliza-se o BDE PostgreSQL/Postgis com a extensão pgRouting[1], a qual é um projeto *Open Source*, distribuído sob a licença GPLv2[2] e apoiado por comunidades, empresas e organizações. Dentre as funcionalidades que o pgRouting disponibiliza, pode-se destacar: verificar qual a melhor rota entre dois pontos, distância na condução de um automóvel, diversos caminhos alternativos, verificação de falhas na rede, entre outras.

Para utilizar o pgRouting na criação de uma rede, se faz necessário uma tabela contendo um conjunto mínimo de atributos (i.e., Direção, LatitudePonto1, LongitudePonto1, LatitudePonto2, LongitudePonto2, Origem, Destino, GeometriaTrecho). Estes atributos definem a topologia de uma rede em um BDE. Observa-se na Figura 4 um exemplo de uma rede contendo cinco nós (i.e., P1, P2, P3, P4 e P5) e quatro arcos (i.e., A1, A2, A3 e A4). Seus dados podem ser armazenados da forma vista no Quadro 5.

Figura 4 – Rede com Cinco Nós e Quatro Arcos



Fonte: O Autor

[1] http://pgrouting.org
[2] https://www.gnu.org/licenses/gpl-2.0.html - Acessado em 14/07/2015

Quadro 5 – Representação dos Dados da Rede da Figura 4

Id	Dir	Orig	Dest	Lat1	Long1	Lat2	Long2	Geom
A1	FT	A1	A4	0	5	2.5	2.5	LINESTRING(0 5,2.5 2.5)
A2	FT	A2	A3	0	0	2.5	2.5	LINESTRING(0 0,2.5 2.5)
A3	FT	A1	A3	2.5	2.5	5	5	LINESTRING(2.5 2.5,5 5)
A4	FT	A2	A4	2.5	2.5	5	0	LINESTRING(2.5 2.5,5 0)

Fonte: O Autor

Observa-se no Quadro 5, que o arco A1 possui a direção da origem para o destino (i.e., FT "*from the source to target*"); parte do arco A1 para o arco A4 (i.e., colunas "Orig" e "Dest" respectivamente); possui um nó com coordenadas: latitude 0 e longitude 5, e outro nó com coordenadas: latitude 2.5 e longitude 2.5; e possui a geometria "LINESTRING(0 5,2.5 2.5)".

Com esse conjunto mínimo de atributos, é possível verificar, por exemplo, se a rede contém trechos não conectados a nós ou trechos com buracos. Outras funcionalidades requerem outros atributos, como por exemplo: para verificar o caminho mais curto entre dois pontos é preciso o atributo "custo" do perímetro ou trecho em questão e para guardar a direção do trecho é preciso informar o atributo "direção". Vale destacar que os atributos Origem e Destino são as chaves estrangeiras dos pontos da rede, e são criados após o procedimento de criação da topologia (função "pgr_createTopology" no pgRouting). O tributo que informa a direção do trecho (direção) pode ser omitido. Nesse caso, assume-se que o trecho é bidirecional, estando presente apenas nos tipos de rede Bidirecional e Múltipla. Os tipos de Rede direcionadas da Origem para Destino, do Destino para Origem e Unidirecionais não permitem omitir o atributo de direção (PGROUTING, 2013).

2.3 Desenvolvimento Dirigido por Modelos

Model Driven Development (MDD) e *Model Driven Architecture* (MDA) são abordagens utilizadas para auxiliar o desenvolvimento de sistemas. A principal diferença entre estas abordagens dá-se pelo motivo da MDA ser uma proposta do *Object Management Group* (OMG)[3] que segue a notação da UML; e MDD permitir usar qualquer notação. Independente das diferenças entre MDD e MDA, estas abordagens são usadas quando deseja-se especificar: linguagens de modelagem, regras para validar modelos e rotinas para transformações de modelos (BRAMBILLA; CABOT; WIMMER, 2012) .

Sobre linguagens de modelagem, vale destacar que estas têm uma sintaxe concreta (ou notação), uma sintaxe abstrata (ou metamodelo) e uma semântica (ou significado da notação). Neste contexto, ressalta-se que o metamodelo corresponde à gramática da linguagem de modelagem, pois especifica os seus conceitos e as combinações sintaticamente válidas entre eles (BRAMBILLA; CABOT; WIMMER, 2012).

A respeito de regras para validar modelos, é importante ressaltar que podem existir combinações sintaticamente válidas no metamodelo que, por razões de semântica estática, devem ser impedidas. Isto é, para que seja possível estabelecer um significado para determinadas construções sintaticamente corretas, é necessário definir restrições/condições relativas ao uso dos seus elementos sintáticos (FIDALGO et al., 2012). Estas regras podem ser especificadas em *Object Constraint Language* (OCL) (OCL, 2015) ou qualquer outra linguagem semelhante. Por exemplo, regras OCL podem ser escritas para informar os seguintes erros: uma entidade não pode ser fraca dela mesma, uma classe não pode ser superclasse dela mesma, um relacionamento espacial “contém” não pode ser “Ponto contém Polígono” ou “Linha contém Polígono”.

Com relação às transformações de modelos, estas podem ser resumidas em duas abordagens: modelo-modelo (ou M2M) e modelo-texto (ou M2T). Por exemplo, na transformação de um modelo conceitual para um

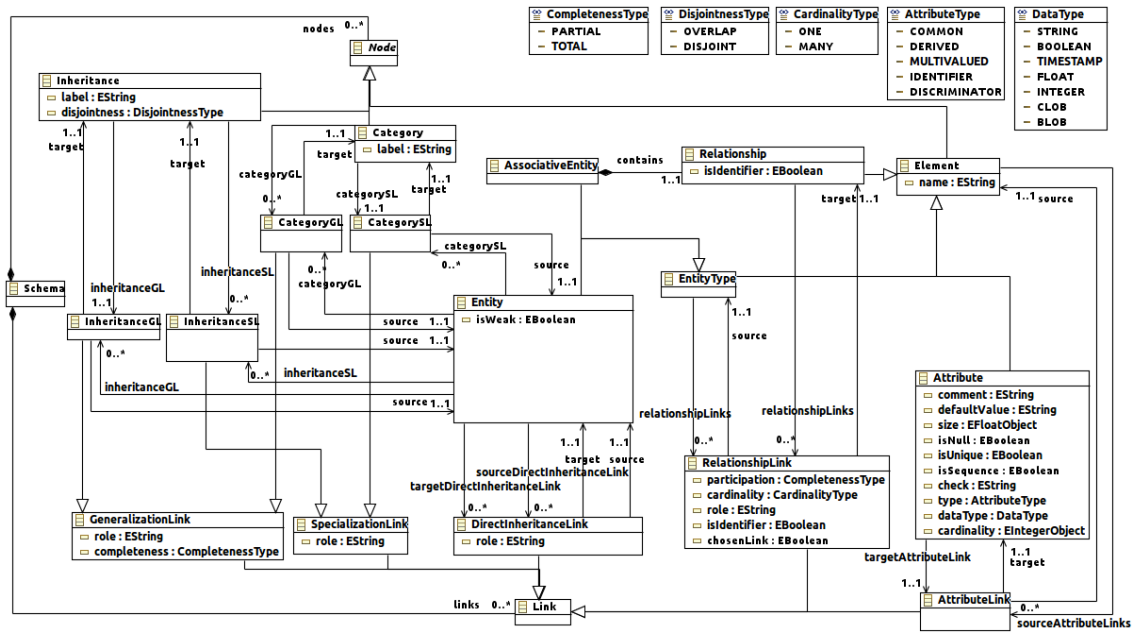
[3] http://www.omg.org/mda/

modelo lógico tem-se a abordagem M2M e na transformação de um modelo lógico para um código SQL a abordagem M2T (BRAMBILLA; CABOT; WIMMER, 2012). Vale ressaltar que as transformações M2T podem incluir a geração de código referente a regras de negócio. Por exemplo, a escrita de um gatilho (*trigger*) para impedir o registro de uma ação caso esta não satisfaça a condição de um relacionamento convencional (um cliente não pode ter mais do que três contas) ou espacial (a geometria de um município deve estar contida na geometria do seu estado).

2.4 O Metamodelo EERMM

No contexto desta dissertação, o termo “linguagem EER” será usado no lugar de “modelo EER”, pois a partir da linguagem EER pode-se diagramar vários modelos EER. Posto isto, este trabalho considera o *Enhanced Entity Relationship Metamodel* (EERMM) (FIDALGO et al., 2012) como o metamodelo da linguagem EER. O EERMM é escolhido, pois é o único metamodelo que cobre todos os conceitos da notação de (ELMASRI e NAVATHE, 2011), a qual é bem aceita pela comunidade de banco de dados e é próxima à notação de (CHEN, 1976). O EERMM é visto na Figura 5.

Figura 5 - Metamodelo EERMM



Fonte: (FIDALGO et al., 2013)

Observa-se na Figura 5 que o EERMM é formado por três metaclasses principais: Esquema (i.e., *Schema*), Nó (i.e., *Node*) e Ligação (i.e., *Link*). A

metaclasse *Schema* representa a área de modelagem do projeto, que é composta por nós e *links*. A metaclasse *Node* representa: os Elementos (Entidade, Relacionamento e Atributo), Categoria e Herança. E a metaclasse *Link* representa as ligações para: Especialização, Generalização, Herança Direta, Relacionamentos e Atributos. Note que apesar de o EERMM ser um metamodelo que observa todos os construtores da linguagem EER, este não é oferece suporte à modelagem de características espaciais.

2.5 Considerações Finais

O metamodelo EERMM é uma especificação da sintaxe concreta da linguagem EER e define a relação entre seus construtores. Este metamodelo é utilizado para a construção da proposta da presente dissertação.

As visões e tipos de dados espaciais são fatores importantes para o entendimento de múltipla representação em relacionamentos espaciais, pois a integridade desses dados depende diretamente dos tipos envolvidos na operação topológica, como também a persistência e integridade dos dados de redes.

É importante diferenciar, por exemplo, linguagens que utilizam a abordagem MDA e MDD. As abordagens baseadas em MDA possuem diferentes níveis de abstração (i.e., CIM, PIM e PSM), o que pode levar o projetista a um esforço adicional para desenvolver um projeto de banco de dados espaciais, se comparado a abordagens MDD.

A integridade de dados espaciais é um fator relevante para manutenção e análises estratégicas. Esta integridade parte da verificação de inconsistências ainda no projeto conceitual, pois é quando modificações são menos custosas do que em ambiente de produção. A seguir, no Capítulo 3, são discutidos trabalhos relacionados, junto aos requisitos que formam a proposta dessa dissertação.

3. Trabalhos Relacionados

Neste Capítulo, faz-se uma análise dos trabalhos relacionados à modelagem de BDE. Como critério de seleção, são consideradas apenas propostas que tenham ferramenta CASE que permita modelar seus conceitos. Ou seja, a ferramenta CASE deve dar evidências que os conceitos propostos são programáveis via um *software*. Com isso, são excluídos importantes trabalhos, tais como: (BÉDARD et al., 2004), (TRYFONA et al., 1999) e (KOSTERS; PAGEL; SIX, 1997). O restante deste Capítulo está estruturado da seguinte maneira: primeiro descreve-se os critérios para avaliação dos trabalhos relacionados. Em seguida, usando os critérios previamente definidos, os principais trabalhos relacionados a esta proposta são avaliados. Por fim, apresenta-se uma avaliação consolidada destes trabalhos.

3.1 Critérios de avaliação

Os critérios para avaliação dos trabalhos referem-se aos conceitos inerentes à sintaxe concreta (ou notação) da linguagem avaliada. Estes critérios são:

1. **Visão de Campo** - verifica se é possível representar os seguintes tipos de dados: Grade de Células (*Grid of Cells*), Polígonos Adjacentes (*Adjacent Polygons*), Isolinha (*Isolines*), TIN (*Triangular Irregular Network*), Grade de Pontos (*Grid of Points*) e Pontos Irregulares (*Irregular Points*)**.** Por exemplo, dados capturados por meio de sensoriamento remoto podem ser representados pelo tipo Grade de Células.
2. **Visão de Objeto** - verifica se é possível representar, no mínimo, os seguintes tipos de dados: Ponto (*Point*), Linha (*Line*) e Polígono (*Polygons*). Por exemplo, um endereço pode ser representado por um tipo geométrico *Point*. Assim também, um campo de futebol pode ser representado por um tipo geométrico *Polygon*.
3. **Múltiplas Representações** - verifica se é possível modelar mais de uma representação espacial em uma classe**.** Por exemplo, uma propriedade

contendo várias construções pode ser representada por mais de uma geometria: a residência pelo tipo *Point*, e o terreno pelo tipo *Polygon*.

4. **Relacionamento Topológico** - verifica se é possível representar operações topológicas em relacionamentos espaciais. Por exemplo, para representar uma relação entre duas geometrias que se tocam, deve-se permitir utilizar a operação *Touch*.
5. **Pictograma em Classe** - verifica se a linguagem utiliza representação espacial diretamente na classe. Por exemplo, um pictograma que representa o tipo de dado Ponto deve ser especificado ao lado do nome da classe.
6. **Pictograma em Atributo** - verifica se é possível inserir pictogramas de tipos espaciais em um atributo. Por exemplo, um pictograma que representa um tipo de dado espacial, deve ser especificado ao lado do nome do atributo.
7. **Modelagem de Redes -** verifica se é possível modelar redes espaciais. Por exemplo, em uma rede Arco-Nó deve-se permitir o uso de uma representação espacial para o Arco (i.e., Linha) e outra para o Nó (i.e., Ponto).
8. **Modelagem Temporal** - verifica se é possível modelar dados e relacionamentos temporais. Por exemplo, em uma entidade/classe que guarda o momento em que o dado foi capturado, deve-se utilizar um construtor que o defina (e.g., *Instant*).
9. **Verificação topológica -** verifica se é possível impedir a especificação equivocada de operações topológicas inválidas. Por exemplo, as operações Polígono IN Ponto e linha IN Ponto são inválidas - e devem ser evitadas ou reportadas como erros -;
10. **Representatividade topológica -** verifica se, por meio de pictogramas, é possível representas os diferentes tipos de dados espaciais envolvidos em uma operação topológica. Por exemplo, a operação Linha CROSS Linha deve ser representada por um pictograma que corresponda a uma linha cruzando outra linha;
11. **Representatividade de Redes -** verifica se, por meio de pictogramas, é possível representar os diferentes tipos de direções que os trechos de uma rede podem ter. Por exemplo, uma rede formada por trechos com

fluxo da “Origem para Destino” deve ter um pictograma expressivo e diferente de uma rede formada por trechos com fluxo do “Destino para a Origem”;

12. **Representatividade de Coleções Geométricas** - verifica se, por meio de pictogramas, é possível representas os tipos de dados espaciais que correspondem às coleções geométricas. Por exemplo, tipos como Multipolígonos (*Multipolygons*), Multipontos (*Multipoints*) e Multilinhas (*Multilinestring*) devem ser representados por pictogramas expressivos;
13. **Nomeação de Atributos Espaciais** – verifica se é possível personalizar o nome de atributos espaciais. Por exemplo, “marco zero” e “prefeitura” devem ser nomes permitidos para atributos espaciais;
14. **Repetição de Tipo Espacial** – verifica se é possível criar, em uma mesma entidade ou classe, mais de um atributo espacial do mesmo tipo. Por exemplo, em uma entidade Cidade, a criação dos atributos “marco zero” e “prefeitura”, ambos do tipo ponto, deve ser permitida;
15. **Desambiguidade de operação** – verifica se, dada uma operação topológica envolvendo duas entidades com mais de uma representação espacial, explicita quais geometrias participam da operação. Por exemplo, considerando duas entidades com múltiplas representações cada uma: Município contendo um Ponto (representando o marco zero) e um Polígono (representando os limites); e Estado contendo Ponto (representando o centroide) e Polígono (representando os limites). Na operação “Município (ponto) IN Estado (Polígono)” deve ficar explícito que o Marco Zero do Município é que deve estar contido nos Limites do Estado, e não os Limites do Município contidos nos Limites do Estado. Nem tão pouco, os Limites do Município contidos no Centroide do Estado.
16. **Operação Topológica sobre Raster** – verifica se é possível o uso do tipo Raster em relacionamentos topológicos. Por exemplo, para verificar se um endereço representado por um Ponto está contido nos limites de um Raster, faz necessária a operação Ponto IN Raster;
17. **Relacionamentos N-ários espaciais** - verifica se é possível operações topológicas sobre relacionamentos N-ários. Por exemplo, considerando o relacionamento N:N:N entre Médico, Paciente e Unidade de Saúde,

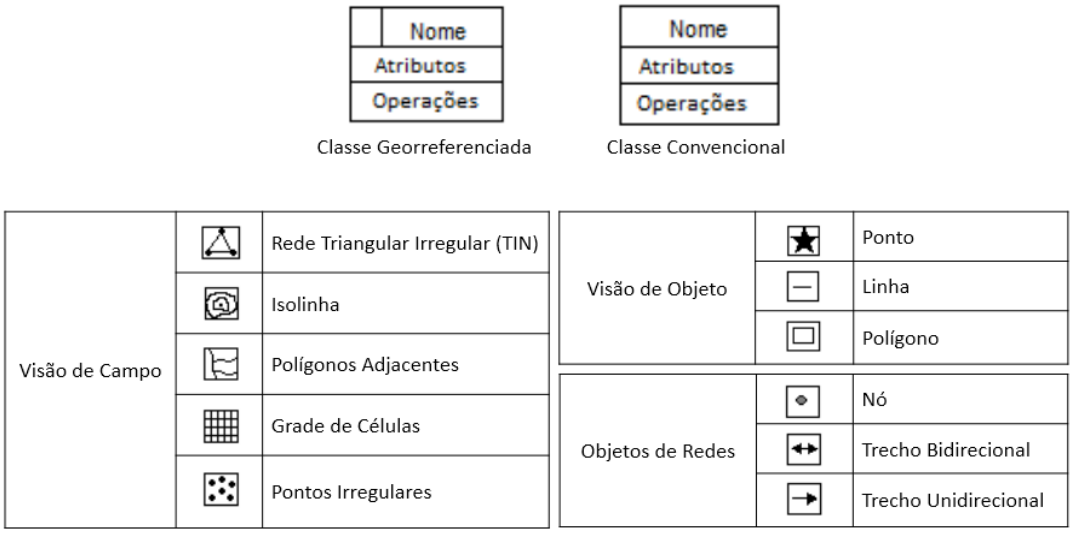
permitir especificar uma condição espacial com a operação IN, tal que a “Residência” do Paciente deve estar dentro da “Região de Atendimento” da Unidade de Saúde.

3.2 OMT-G

A linguagem de modelagem *Object Modeling Technique - Geographic* (OMT-G) (BORGES; DAVIS; LAENDER, 2001; LIZARDO; DAVIS JR, 2014; MARTÍNEZ; FROZZA, 2014) permite modelar uma dada realidade geográfica por meio dos construtores: “todo-parte” topológico (agregação espacial), estrutura de redes, múltiplas representações de objetos e relacionamentos espaciais, como também, atributos alfanuméricos e métodos em cada classe modelada. Basicamente, esta linguagem contém três conceitos: classes, relacionamentos e restrições de integridade espacial; e para o desenvolvimento dos modelos faz uso de três diagramas: diagrama de classe, diagrama de transformação e um diagrama de apresentação (BORGES; DAVIS; LAENDER, 2001). Desses diagramas, apenas o diagrama de classe é utilizado na análise comparativa, pois os outros dois estão fora do escopo dessa pesquisa.

O OMT-G faz uso de dois tipos de classes (i.e., Convencionais e Georreferenciadas). Para classes georreferenciadas, utiliza pictogramas para representar cada tipo espacial, sendo divididos em dados para Visão de Campo, Visão de Objetos e Objetos de Redes. Esses pictogramas são usados no canto superior esquerdo da classe georreferenciada. Na Figura 6, mostram-se os construtores básicos do OMT-G.

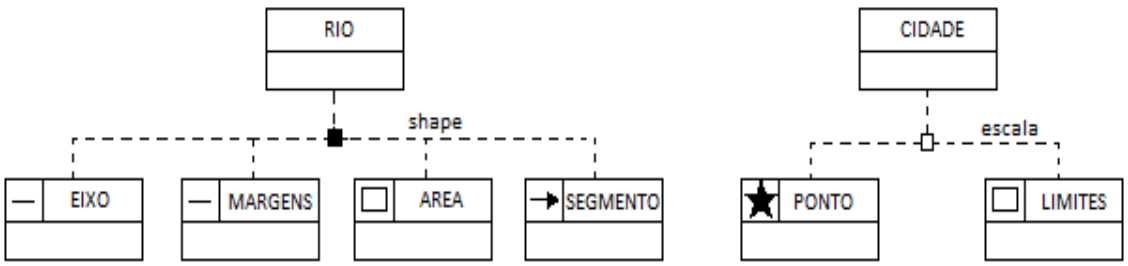
Figura 6 – Construtores do OMT-G: Classe Georreferenciada/Convencional e Pictogramas



Fonte: (BORGES; DAVIS; LAENDER, 2001) adaptado

O conceito de Múltiplas Representações no OMT-G é definido por meio do construtor de Generalização Cartográfica. Esse construtor trata as diferentes representações da geometria modelada, pois em uma determinada escala, a geometria é mais bem representada por Pontos (em escalas pequenas) ou por Polígono (em escala maior). Dois exemplos de múltiplas representações são mostrados na Figura 7.

Figura 7 - Generalização Cartográfica/Múltiplas Representações

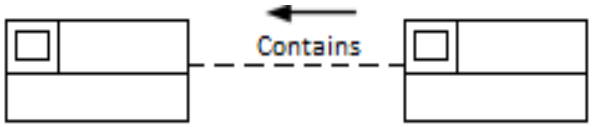


Fonte: (BORGES; DAVIS; LAENDER, 2001) adaptado.

O Relacionamento Espacial no OMT-G é definido como sendo uma linha pontilhada que interliga duas classes georreferenciadas. A definição de uma operação topológica para um relacionamento é feita por uma descrição sobre a linha, a qual é inserida pelo projetista como uma *string* descritiva (i.e., sem sintaxe ou semântica para geração automática de código executável). Esse

relacionamento é realizado apenas entre classes georreferenciadas e pode ser visto na Figura 8.

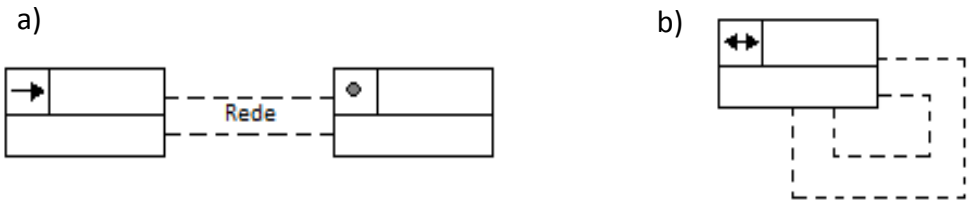
Figura 8 - Relacionamento Espacial



Fonte: (BORGES; DAVIS; LAENDER, 2001)

O OMT-G define duas formas para modelagem de Redes (i.e., Arco-Nó e Arco-Arco). Na Figura 9a, a primeira forma (Arco-Nó) é ilustrada usando duas classes georreferenciadas (uma correspondente a um Arco e outra a um Nó), conectadas por meio de um relacionamento de redes (linha tracejada dupla). A segunda forma (Arco-Arco) é ilustrada na Figura 9b, onde a classe é auto conectada por uma linha tracejada dupla.

Figura 9 - Modelagem de Redes no OMT-G



Fonte: (BORGES; DAVIS; LAENDER, 2001) adaptado

O OMT-G considera nove tipos de relacionamentos espaciais: Toca (*Touch*), Contido (*IN*), Cruza (*Cross*), Sobrepõe (Overlap), Disjunto (*Disjoint*), Adjacente a (*Adjacent to*), Igual (*Equal*), Contém (*Contain*), e Perto (*Near*). Os relacionamentos *Contain* (contém) e *In* (contido) têm a mesma função, pois o que os diferencia é o elemento referenciado (i.e., “A contém B” = “B está contido em A”). No Quadro 6, apresenta-se a avaliação do OMT-G, a qual é feita a partir do uso das ferramentas OMT-G *Design Web* e *Desktop* (LIZARDO; DAVIS JR, 2014; MARTÍNEZ; FROZZA, 2014).

Quadro 6 - Avaliação do OMT-G

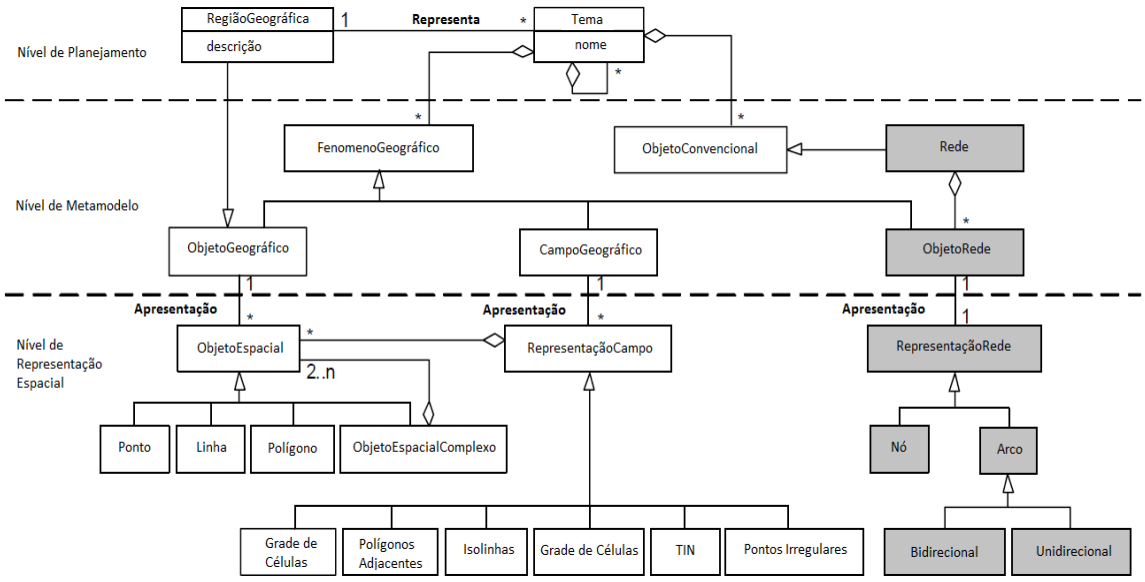
Características de Avaliação	Sim/Não
Visão de Campo	SIM
Visão de Objeto	SIM
Múltiplas Representações	SIM
Relacionamento Topológico	SIM
Pictograma em Classe	SIM
Pictograma em Atributo	NÃO
Modelagem de Redes	SIM
Modelagem Temporal	NÃO
Verificação topológica	NÃO
Representatividade topológica	NÃO
Representatividade de Redes	NÃO
Representatividade de Coleções Geométricas	NÃO
Nomeação de Atributos Espaciais	NÃO
Repetição de Tipo Espacial	NÃO
Desambiguidade de operação	NÃO
Operação Topológica sobre Raster	NÃO
Relacionamentos N-ários espaciais	NÃO

Fonte: O Autor

3.3 UML-GeoFrame

UML-GeoFrame (LISBOA FILHO e IOCHPE, 2008; LISBOA FILHO; JÚNIOR RODRIGUES; DALTIO, 2004; LISBOA FILHO et al., 2007) propõe um conjunto de estereótipos em hierarquia que compõem um perfil UML para modelagem conceitual de BDE. Na Figura 10, pode-se ver como as classes do UML-GeoFrame são organizadas em níveis hierárquicos.

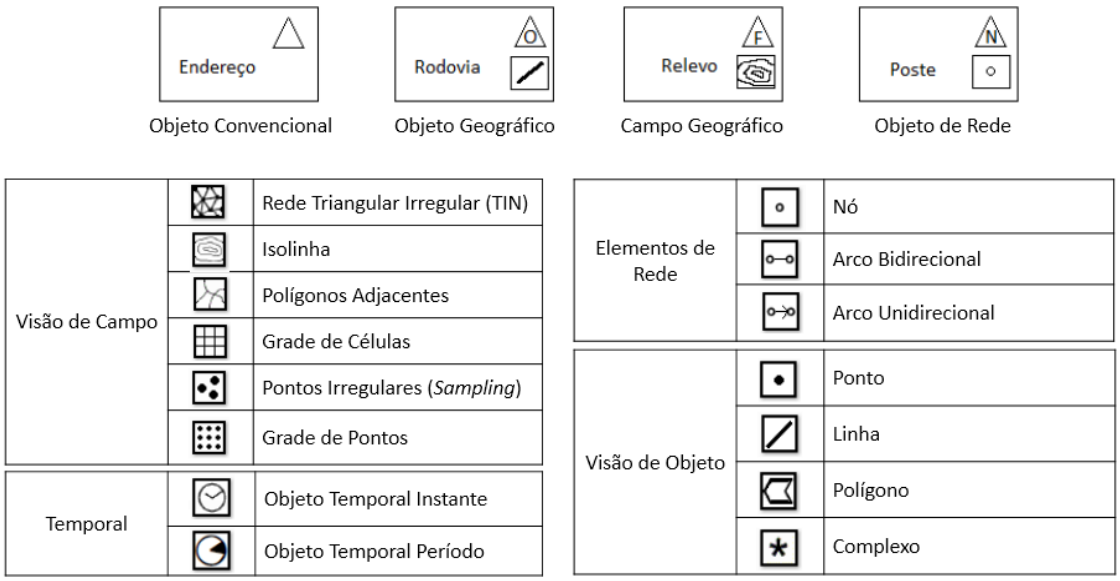
Figura 10 - Hierarquia de Classes do *Framework* GeoFrame



Fonte: (STEMPLIUC et al., 2009)

No nível de representação, é possível identificar os tipos de dados e conceitos que o UML-GeoFrame oferece suporte. Isso é, Visão de Objeto com: Ponto, Linha, Polígono e Complexo; Visão de Campo com: Grade de Células, Polígonos Adjacentes, Isolinhas, Grade de Pontos, Rede Triangular Irregular (TIN) e Pontos Irregulares. Por fim, modelagem de Redes por meio de Nó e Arco, sendo que o último só pode ser direcional e bidirecional. Na Figura 11, mostra-se o conjunto de estereótipos utilizado pelo UML-GeoFrame.

Figura 11 - Classes e Pictogramas em UML-GeoFrame



Fonte: (STEMPLIUC et al., 2009)

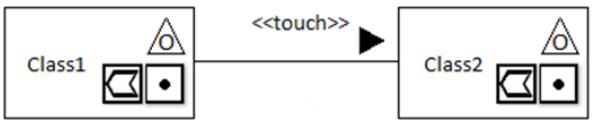
Na Figura 11, observa-se que o UML-GeoFrame representa um objeto convencional, um objeto geográfico, um campo geográfico e um objeto de rede por um pictograma com forma de um triângulo sem desenho adicional, com a letra “O”, com a letra “F” e com a letra “N”, respectivamente. Além desta representação, também é necessário inserir nas classes os pictogramas referentes aos tipos dos dados espaciais.

Ressalta-se que uma classe pode ter mais de um pictograma de tipo de dado espacial (i.e., múltipla representação espacial) e que o projetista pode escrever regras OCL para descrever regras de integridade espacial (cf. Figura 12b). Contudo, é importante notar que as regras OCL, por serem escritas pelos projetistas, não evitam de forma automática que relacionamentos topológicos inválidos sejam especificados (e.g., um polígono contido em um ponto). No Quadro 7, apresenta-se a avaliação do UML-GeoFrame, a qual é feita a partir do uso da ferramenta ArgoCaseGeo[4].

[4] http://www.dpi.ufv.br/projetos/argocasegeo/ Acessado em 09/01/2016

Figura 12 - Relacionamento Espacial no UML-GeoFrame

a)



b)

A	context Class1 inv: Class1.area.touch.Classe2.area inv: Class1.area.touch.Classe2.point inv: Class1.point.touch.Classe2.area inv: Class1.point.in.Classe2.point

Fonte: (RIBEIRO et al., 2013)

Quadro 7 - Avaliação do UML-GeoFrame

Características de Avaliação	**Sim/Não**
Visão de Campo	**SIM**
Visão de Objeto	**SIM**
Múltiplas Representações	**SIM**
Relacionamento Topológico	**SIM**
Pictograma em Classe	**SIM**
Pictograma em Atributo	**NÃO**
Modelagem de Redes	**SIM**
Modelagem Temporal	**SIM**
Verificação topológica	**NÃO**
Representatividade topológica	**NÃO**
Representatividade de Redes	**NÃO**
Representatividade de Coleções Geométricas	**NÃO**
Nomeação de Atributos Espaciais	**NÃO**
Repetição de Tipo Espacial	**NÃO**
Desambiguidade de operação	**NÃO**
Operação Topológica sobre Raster	**NÃO**
Relacionamentos N-ários espaciais	**NÃO**

Fonte: O Autor

3.4 MADS

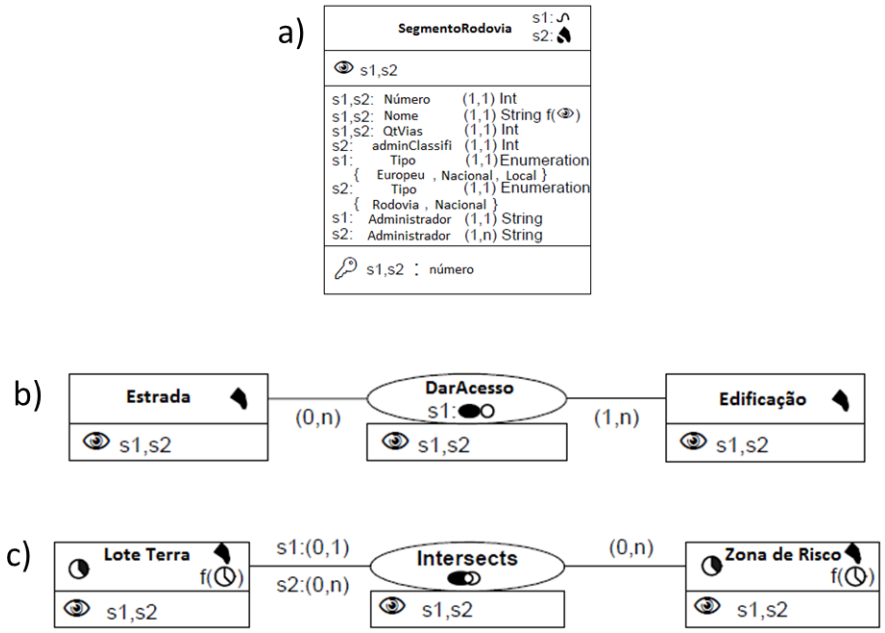
A linguagem *Modeling of Application Data with Spatio-temporal features* (MADS) (PARENT; SPACCAPIETRA; ZIMÁNYI, 2006a, 2006b; PARENT et al., 1998) oferece suporte à Múltiplas Representações Espaciais e Temporais, as quais são feitas a partir do uso de pictogramas em atributos e relacionamentos. Estes pictogramas podem ser vistos na Figura 13. Vale ressaltar que a linguagem MADS é a única entre as demais analisadas que permite os tipos coleção (i.e., *Multipoint*, *Multiline* e *Multipolygon*) identificados em objetos geográficos complexos na Figura 13.

Figura 13 - Pictogramas em MADS

Fonte: (PARENT et al., 1998; SPACCAPIETRA; PARENT; ZIMÁNYI, 2007)

O conceito de Múltiplas Representações visto na Figura 14 na linguagem MADS, é tratado de forma diferente às dos outros trabalhos, pois, para cada objeto espacial, pode-se atribuir mais de uma visão. Por exemplo, o atributo ‘Número’ pertence tanto à visão S1 quanto visão S2; já o atributo ‘Tipo’ pertence apenas à visão S2 (Figura 14a); por sua vez, o relacionamento espacial ‘DarAcesso’ apenas existe na visão S1 (Figura 14b). Por fim, a cardinalidade do relacionamento Intersects é definida como (0,1) para a visão S1 e (0,n) para a visão S2 (Figura 14c).

Figura 14 - Múltiplas Representações em MADS



Fonte: (PARENT; SPACCAPIETRA; ZIMÁNYI, 2006a) traduzido

Diante das características observadas na linguagem de modelagem MADS, no Quadro 8, apresenta-se a avaliação do MADS, a qual é feita a partir do uso da ferramenta MADS *Editor*[5].

Quadro 8 - Avaliação do MADS

Características de Avaliação	**Sim/Não**
Visão de Campo	**NÃO**
Visão de Objeto	**SIM**
Múltiplas Representações	**SIM**
Relacionamento Topológico	**SIM**
Pictograma em Classe	**SIM**
Pictograma em Atributo	**SIM**
Modelagem de Redes	**NÃO**
Modelagem Temporal	**SIM**
Verificação topológica	**NÃO**
Representatividade topológica	**NÃO**
Representatividade de Redes	**NÃO**
Representatividade de Coleções Geométricas	**SIM**
Nomeação de Atributos Espaciais	**SIM**
Repetição de Tipo Espacial	**SIM**
Desambiguidade de operação	**NÃO**
Operação Topológica sobre Raster	**NÃO**
Relacionamentos N-ários espaciais	**PARCIAL**

Fonte: O Autor

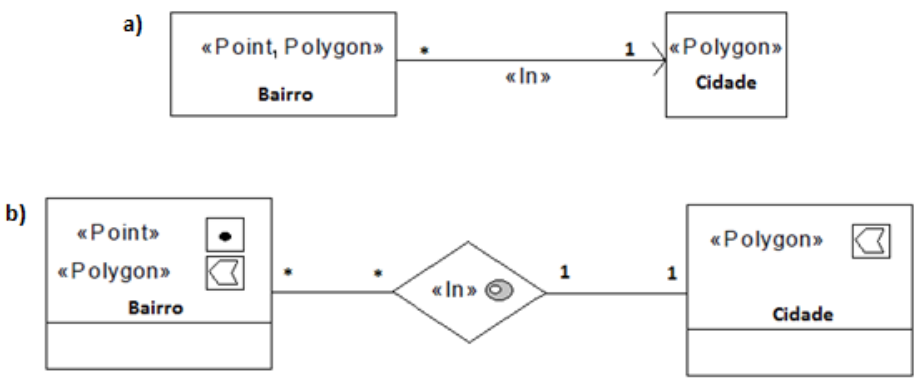
[5] http://cs.ulb.ac.be/mads_tools/ Acessado em 09/01/2016

No Quadro 8, verifica-se que a linguagem MADS possui o critério “Relacionamentos N-ários espaciais” de forma parcial, pois, embora permita relacionamentos espaciais envolvendo mais de uma classe, não fica claro quais geometrias participam da operação em questão; quando a classe possui múltiplas representações com geometrias duplicadas (cf. Figura 36).

3.5 GeoProfile

O GeoProfile (FERREIRA; STEMPLIUC; LISBOA-FILHO, 2014; LISBOA-FILHO et al., 2013; SAMPAIO, 2009) é um perfil UML para modelagem conceitual de BDE que reúne os pontos fortes das linguagens UML-GeoFrame, OMT-G, MADS e Perceptory. O uso de perfis UML permite o aproveitamento de toda a estrutura da UML. Assim, o GeoProfile pode ser implementado em várias ferramentas CASE com suporte a *profiles* UML. Contudo, isto exige a configuração de todos os seus estereótipos, o que pode não ser uma atividade trivial para usuários finais, pois exige um bom conhecimento técnico. Ademais, por ser um perfil, algumas representações gráficas do GeoProfile dependem da ferramenta utilizada. Por exemplo, os conceitos de Múltiplas Representações e Relacionamento Espacial são representados de formas diferentes quando são usadas ferramentas distintas (e.g., *Enterprise Architect*[6] “Figura 15a” e *Visual Paradigm for* UML[7] “Figura 15b”).

Figura 15 - Múltiplas Representações e Relacionamento Espacial em Diferentes Ferramentas CASE em GeoProfile



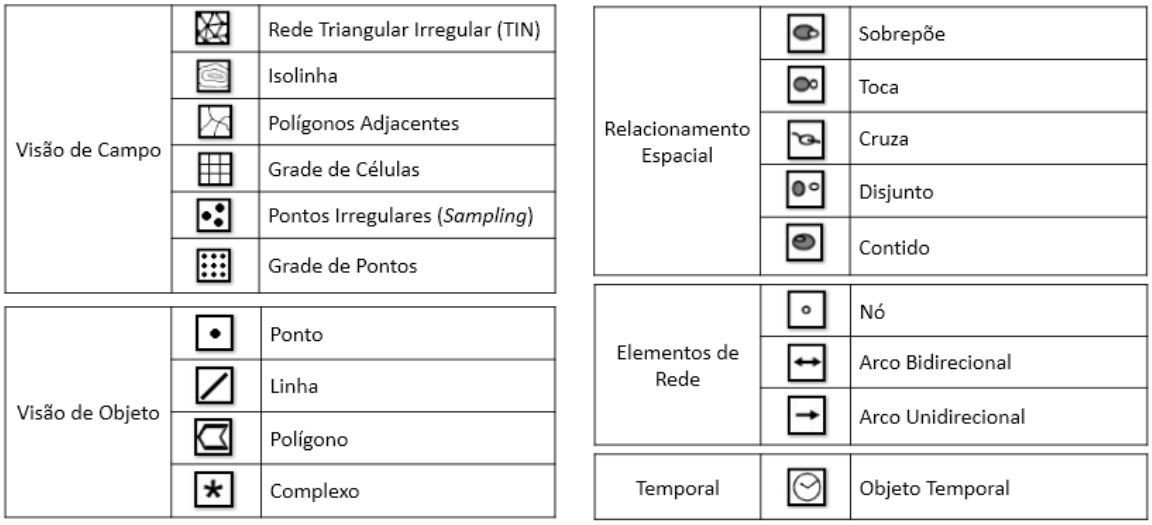
Fonte: (GEOPROFILE, 2015)

[6] http://www.sparxsystems.com.au/ Acessado em 19/03/2015
[7] http://www.visual-paradigm.com/ Acessado em 19/03/2015

Na Figura 15, identificam-se os conceitos de múltiplas representações na classe Bairro (i.e., representado por ponto e por polígono), sendo representado por meio de estereótipos na Figura 15a e por estereótipos e seus respectivos pictogramas na Figura 15b. Além disso, o Relacionamento Espacial na Figura 15a é representado a partir de um estereótipo na associação. Como esta representação não é permitida na ferramenta na *Visual Paradigm for* UML, deve-se colocar o estereótipo e seu respectivo pictograma no losango do relacionamento.

Os pictogramas do GeoProfile são ilustrados na Figura 16. Estas representações são divididas em Visão de Campo e Objetos, Relacionamentos Espaciais, Temporal e Elementos de Redes (LISBOA-FILHO et al., 2013). Ressalta-se que estes pictogramas só aparecem se a ferramenta UML permitir o uso deste recurso.

Figura 16 - Pictogramas no GeoProfile



Fonte: (LISBOA-FILHO et al., 2013)

Note que os pictogramas aplicáveis aos Relacionamentos Espaciais (cf. Figura 13) não cobrem todos os tipos os dados espaciais. Assim, por exemplo, a operação “Linha Cruza Linha” também tem que ser representada pelo pictograma (), o que pode prejudicar o entendimento do modelo.

Como o GeoProfile reúne as principais características das outras linguagens de modelagem espacial, os pictogramas que representam os conceitos de Visão de Campo e Visão de Objetos são baseados na linguagem OMT-G. Os pictogramas que representam os conceitos de relacionamento

espacial e temporal são baseados da linguagem MADS. E os elementos de redes são baseados nos pictogramas da linguagem UML-GeoFrame.

Vale ressaltar que, apenas o GeoProfile define regras para semântica estática e as implementam em OCL (SAMPAIO, 2009). De forma geral, as regras visam evitar: 1) que uma classe tenha estereótipos contraditórios; 2) construções equivocadas de Redes; e 3) relacionamentos topológicos inválidos. Estas regras são implementadas em OCL e são descritas a seguir em linguagem natural:

1. **Regras para evitar estereótipos contraditórios:**
    - Classes com estereótipos de Visão de Campo não podem conter estereótipos de Visão de Objeto;
    - Classes de Redes não podem conter estereótipos de Visão de Campo;
    - Uma classe com um estereótipo “Nó” não pode conter um estereótipo Arco;
    - Uma classe com um estereótipo “Arco Unidirecional” não pode conter um estereótipo “Arco Bidirecional”.
2. **Regras para definir construções válidas para Redes:**
    - Uma modelagem de Rede deve conectar exatamente uma classe com estereótipo Arco com outra classe com um estereótipo Nó.
3. **Regras para definir Relacionamentos Topológicos válidos:**
    - As operações Contido “*IN*” e Disjunto “*Disjoint*” podem ser aplicadas sobre qualquer combinação dos tipos: Ponto, Linha e Polígono;
    - A operação Cruza “*Cross*” só pode ser aplicacada para as combinações Linha-Linha e Linha-Polígono;
    - A operação Sobrepõe “*Overlap*” só pode ser aplicada para as combinações Linha-Linha e Polígono-Polígono;
    - A operação Toca “Touch” só pode ser aplicada para as combinações Polígono-Polígono, Linha-Linha, Linha-Polígono, Ponto-Polígono e Linha-Ponto;

Note que a operação Contido "*IN*" não impede especificar combinações inválidas. Por exemplo, "Polígono Contido em Ponto" e "Polígono Contido em Linha". Ademais, o GeoProfile não tem regras OCL para dados do tipo Raster e, assim como Clementini (CLEMENTINI; DI FELICE; VAN OOSTEROM, 1993), não considera a operação Igual "*Equal*". No Quadro 9, apresenta-se a avaliação das características do perfil GeoProfile.

Quadro 9 - Avaliação do Perfil GeoProfile

Características de Avaliação	**Sim/Não**
Visão de Campo	**SIM**
Visão de Objeto	**SIM**
Múltiplas Representações	**SIM**
Relacionamento Topológico	**SIM**
Pictograma em Classe	**SIM**
Pictograma em Atributo	**NÃO**
Modelagem de Redes	**SIM**
Modelagem Temporal	**SIM**
Verificação topológica	**PARCIAL**
Representatividade topológica	**NÃO**
Representatividade de Redes	**NÃO**
Representatividade de Coleções Geométricas	**NÃO**
Nomeação de Atributos Espaciais	**NÃO**
Repetição de Tipo Espacial	**NÃO**
Desambiguidade de operação	**NÃO**
Operação Topológica sobre Raster	**NÃO**
Relacionamentos N-ários espaciais	**PARCIAL**

Fonte: O Autor

No Quadro 9, verifica-se que o GeoProfile possui, de forma parcial, os critérios: "Verificação Topológica" e Relacionamentos N-ários espaciais. A verificação topológica restringe alguns casos impossíveis (e.g., Polígono sobrepor Linha). Contudo, não evita, por exemplo, que um Ponto contenha um Polígono. E, de forma similar ao visto na linguagem MADS, não deixa claro qual (ou quais) geometrias participam do relacionamento n-ário espacial em classes com múltiplas representações.

3.6 Considerações Finais

Nesta seção, resume-se o que foi visto nos trabalhos relacionados, destacando suas principais características. Para isto, no Quadro 10, apresenta-se uma matriz que consolida os pontos fortes e fracos de cada trabalho segundo as características de avaliação definidas no início deste capítulo.

Quadro 10 - Avaliação das Propostas

Conceitos \ Trabalhos	OMT-G	GeoProfile	UML-GeoFrame	MADS
Visão de Campo	SIM	SIM	SIM	NÃO
Visão de Objeto	SIM	SIM	SIM	SIM
Múltiplas Representações	SIM	SIM	SIM	SIM
Relacionamento Topológico	SIM	SIM	SIM	SIM
Pictograma em Classe	SIM	SIM	SIM	SIM
Pictograma em Atributo	NÃO	NÃO	NÃO	SIM
Modelagem de Redes	SIM	SIM	SIM	NÃO
Modelagem Temporal	NÃO	SIM	SIM	SIM
Verificação topológica	NÃO	PARCIAL	NÃO	NÃO
Representatividade topológica	NÃO	NÃO	NÃO	NÃO
Representatividade de Redes	NÃO	NÃO	NÃO	NÃO
Representatividade de Coleções Geométricas	NÃO	NÃO	NÃO	SIM
Nomeação de Atributos Espaciais	NÃO	NÃO	NÃO	SIM
Repetição de Tipo Espacial	NÃO	NÃO	NÃO	SIM
Desambiguidade de operação	NÃO	NÃO	NÃO	NÃO
Operação Topológica sobre Raster	NÃO	NÃO	NÃO	NÃO
Relacionamentos N-ários espaciais	NÃO	PARCIAL	NÃO	PARCIAL

Fonte: O Autor

No Quadro 10, observa-se que os conceitos de Visão de Objetos, Múltiplas Representações, Relacionamento Espacial e Pictograma em Classe

são abordados por todas as linguagens; caracterizando-os em requisitos triviais.

Os critérios: Pictograma em Atributo, Verificação topológica, Representatividade Topológica, Representatividade de Redes, Representatividade de Coleções Geométricas, Nomeação de Atributos Espaciais, Repetição de Tipo Espacial, Desambiguidade de operação, Operação Topológica sobre Raster e Relacionamentos N-ários espaciais, formam parte das contribuições deste trabalho.

Destes critérios citados, apenas o GeoProfile permite Verificação Topológica de forma parcial, implementada por meio de regras em OCL (cf. Seção 3.5). Nenhuma linguagem possui os critérios: Representatividade topológica, Representatividade de Redes, Desambiguidade de operação e Operação Topológica sobre Raster**.** Observa-se também que, apenas a linguagem MADS define e implementa os critérios: Pictograma em Atributo, Representatividade de Coleções Geométricas, Nomeação de Atributos Espaciais, Repetição de Tipo Espacial. O critério “Relacionamentos N-ários espaciais” é suportado apenas pelas linguagens GeoProfile e MADS de forma parcial (Geoprofile dá suporte apenas na ferramenta *Visual Paradign*). Ademais, os valores que correspondem a “SIM” e “NÃO” para todos os trabalhos relacionados, já denotam sua adesão ao requisito.

4. Uma Extensão Espacial para a Linguagem EER

Com o objetivo de superar as deficiências encontradas nos trabalhos relacionados, uma extensão para a linguagem EER denominada *Spatial Enhanced Entity Relationship* (SEER) é proposta neste Capítulo. Para completar este propósito, as seções deste Capítulo estão organizadas da seguinte maneira: na Seção 4.1 apresenta-se uma visão geral dos principais conceitos da SEER. Na Seção 4.2 discute-se a sintaxe concreta (notação gráfica) da proposta. Na Seção 4.3 especifica-se a sintaxe abstrata (metamodelo) da SEER. Na Seção 4.4 considerando apenas o contexto espacial, define-se uma semântica estática para a SEER. Na Seção 4.5 mostram-se transformações para SQL de um modelo em SEER. Por fim, na Seção 4.6 apresentam-se as considerações finais do capítulo.

4.1 Visão geral da SEER

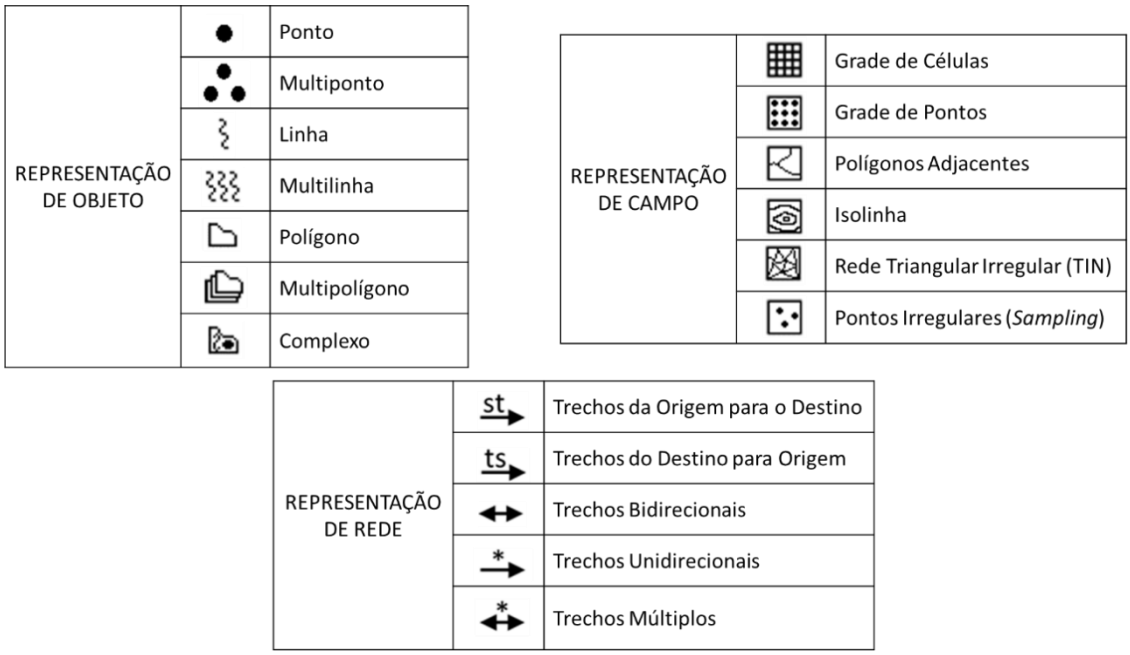
A SEER permite modelar características espaciais utilizando os construtores da linguagem EER segundo a notação de Elmasri e Navathe (ELMASRI e NAVATHE, 2011). Dentre as principais características da SEER, pode-se citar o suporte a: múltiplas representações espaciais, relacionamentos topológicos, modelagem de redes, visão de objeto e visão de campo. Como a SEER é uma linguagem diagramática, esta é formada por três elementos básicos (BRAMBILLA, 2012): I) sintaxe concreta (i.e., uma extensão da notação EER de ELMASRI e NAVATHE, 2011), II) sintaxe abstrata (i.e., uma extensão do metamodelo EERMM) e III) semântica estática (i.e., o significado para cada elemento espacial gráfico e suas regras de boa formação). As próximas seções detalham cada um desses três elementos.

4.2 Sintaxe Concreta

A SEER estende a notação de Elmasri e Navathe (ELMASRI e NAVATHE, 2011) a partir da definição de pictogramas em atributos e relacionamentos. No primeiro caso, os pictogramas são usados para especificar os tipos de dados dos atributos. No segundo, os pictogramas são

usados para definir condições espaciais em relacionamentos. Na Figura 17, mostra-se os pictogramas dos tipos de dados espaciais. Na Figura 18, mostra-se os pictogramas referentes aos relacionamentos topológicos. Ressalta-se que um atributo só pode ter um dos pictogramas apresentados na Figura 17 e que um relacionamento pode ter mais de um dos pictogramas ilustrados na Figura 18.

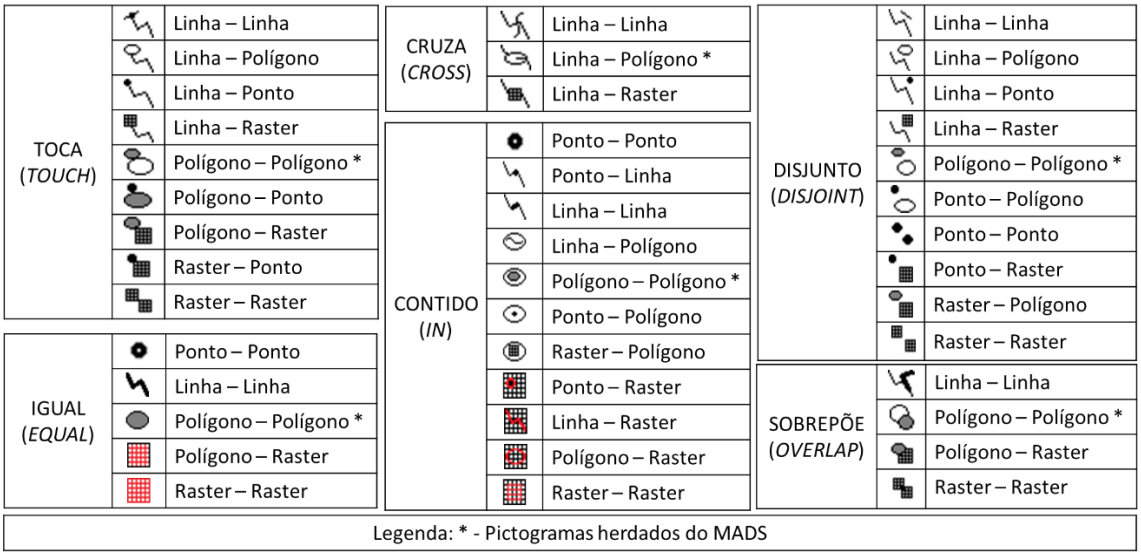
Figura 17 – Representações de Dados na SEER



Fonte: O Autor

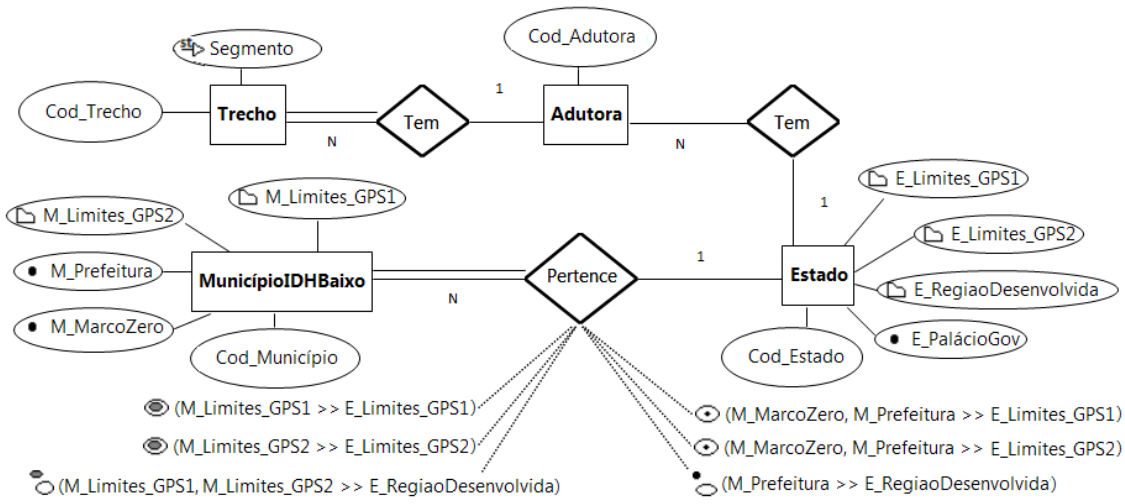
Visando reduzir equívocos na especificação de operações topológicas, a SEER, com base nas operações topológicas propostas por Clementini, Di Felice e Van Oosterom (cf. Subseção 2.2.3), especifica um pictograma para cada combinação válida entre os tipos de dados que podem ocorrer em uma operação topológica. Assim, diferente dos trabalhos relacionados (cf. Capítulo 3) que têm um número reduzido de pictogramas para operações topológicas, a SEER possui quarenta e dois pictogramas (cf. Figura 18), os quais são exclusivos e não podem ser sobrecarregados, isto é, cada pictograma tem uma sintaxe e uma semântica específica. Por exemplo, a sintaxe da operação "👁 - Ponto Contido em Polígono" exige que o primeiro e segundo argumentos sejam, respectivamente, um ponto e um polígono. Por sua vez, não existe nenhuma outra operação cuja semântica seja "um ponto contido em um polígono". Na Figura 19, ilustra-se uma visão geral da notação da SEER.

Figura 18 – Representações para Relacionamentos Espaciais



Fonte: O Autor

Figura 19 – Exemplo de Modelo SEER



Fonte: O Autor

Na Figura 19, os pictogramas referentes aos tipos de dados espaciais são definidos nos atributos, o que, diferente dos trabalhos que usam pictogramas em classes, permite deixar explícito os nomes desses atributos. Além disso, quando é necessário impor operações espaciais sobre relacionamentos, estes recebem um ou mais construtores denominados de “Condição Espacial”, os quais são definidos da seguinte maneira: **pictograma da operação topológica (coleção de atributos espaciais >> atributo espacial).** Os termos entre

parênteses são definidos pela seguinte expressão regular: "**^[\w\-]{1,30}[\,[\w\-]{0,30}]*>>[\w\-]{1,30}$"** em Regex[8]**.** Note que: 1) a seta dupla ">>" separa o lado esquerdo do lado direito da operação; 2) o lado esquerdo deve ter um, mas pode ter vários atributos espaciais (separados por vírgulas); 3) o lado direito só pode ter um atributo espacial e 4) o nome de um atributo espacial tem entre 1 e 30 caracteres, os quais só podem ser alfanuméricos, um traço "-" ou um sublinhado "_". Por exemplo, na Figura 19, o relacionamento "Pertence" contém várias Condições Espaciais, as quais, como dito no parágrafo anterior, devem ser especificadas respeitando a sintaxe e a semântica da operação topológica representada pelo pictograma. Por exemplo, para a Condição Espacial "👁 (M_Limites_GPS1 >> E_Limites_GPS1)", os lados esquerdo e direito devem ser do tipo Polígono e os atributos "M_Limites_GPS1" e "E_Limites_GPS1" não podem ser trocados de posição (apesar de ambos serem polígonos), pois a Condição Espacial nunca seria verdadeira (i.e., o limite de um estado não pode estar contido no limite de um município).

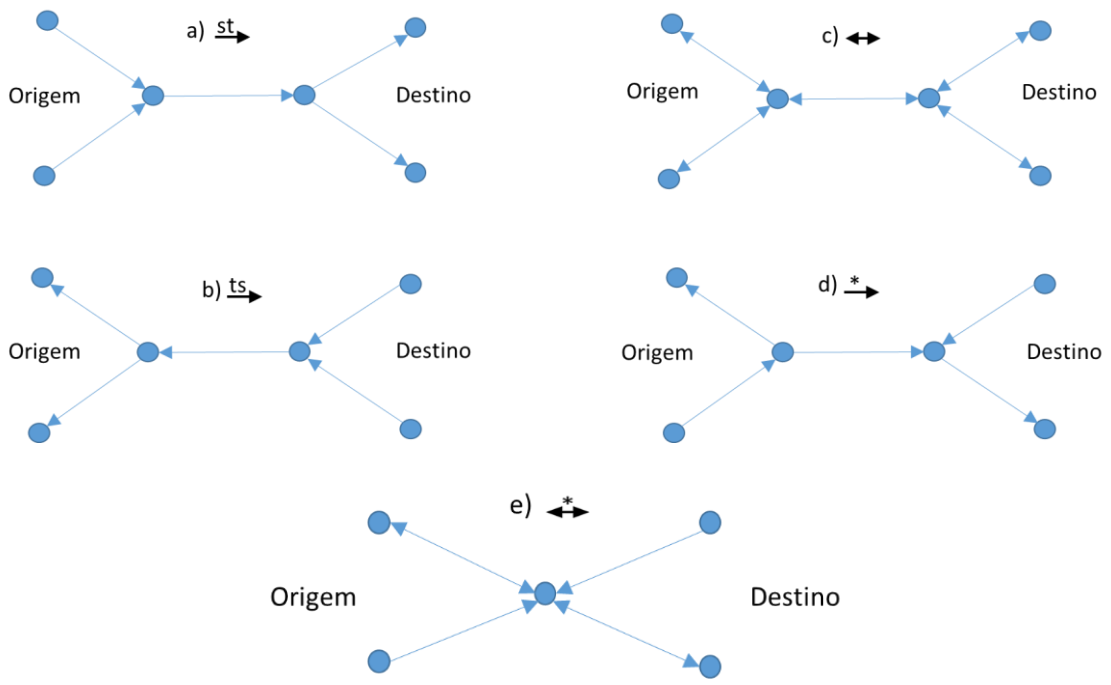
Ainda na Figura 19, observa-se que a Condição Espacial "👁 (M_Prefeitura, M_MarcoZero >> E_Limites_GPS1)" tem mais de um atributo no lado esquerdo. Nestes casos, combina-se cada um dos atributos com o único atributo do lado direito. Para situações que exijam mais de um atributo do lado direito, deve-se especificar outra Condição Espacial. Por exemplo, para combinar os atributos "M_Prefeitura" e "M_MarcoZero" com os atributos "E_Limites_GPS1" e "E_Limites_GPS2", são definidas duas Condições Espaciais. Ressalta-se que a SEER adota esta convenção para não ter que definir um operador para produto cartesiano e outro para mapeamento ordenado pela posição dos atributos (i.e., primeiro atributo do lado esquerdo com o primeiro atributo do lado direito e assim por diante). Ademais, dado que as operações espaciais são mutuamente exclusivas, também vale ressaltar que, independente da operação espacial, não podem haver combinações "lado esquerdo - lado direito" repetidas. Por exemplo, uma vez que a combinação "M_Prefeitura, E_Limites_GPS1" já existe na Condição Espacial "👁 (M_Prefeitura, M_MarcoZero >> E_Limites_GPS1)", esta não pode aparecer em outra Condição Espacial do relacionamento "Pertence", pois isto geraria

[8] http://aurelio.net/regex/guia/

uma contradição, visto que é uma incoerência dizer que a sede da prefeitura deve estar contida e ser disjunta ao limite do seu estado. Por fim, deve-se notar que o construtor Condição Espacial evita ambiguidade, pois este deixa explícito quais são os atributos que devem participar da operação topológica.

A SEER também permite modelar estruturas de redes. Na Figura 20, ilustram-se os tipos de redes que a SEER oferece suporte. Observe que: na Figura 20a, os segmentos da rede são direcionais e partem da origem para o destino; na Figura 20b, os segmentos também são direcionais, mas, ao contrário da Figura 20a, estes partem do destino para a origem; na Figura 20c, todos os segmentos da rede são bidirecionais; na Figura 20d, a rede tem segmentos nos dois sentidos, mas todos são unidirecionais. Por último, na Figura 20e, a rede possui todas as direções e não há um padrão de direção para cada segmento.

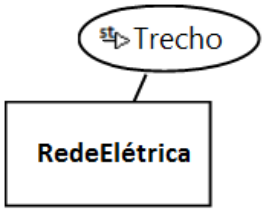
Figura 20 - Tipos de Redes na SEER



Fonte: O Autor

Na SEER, uma Rede formada apenas por arcos é feita a partir da definição de um dos pictogramas de Redes em um atributo. Na Figura 21, mostra-se um exemplo de uma rede formada apenas por arcos unidirecionais que partem da origem para o destino (i.e., “st” *Source to Target*). Na Figura 22, ilustra-se como modelar uma rede do tipo Arco-Nó na SEER.

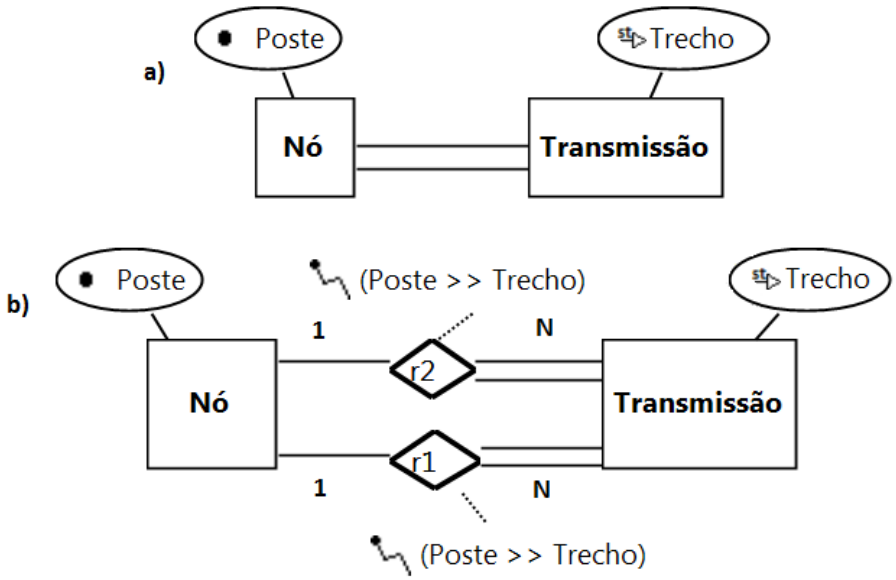
Figura 21 – Exemplo de como modelar uma rede básica na SEER



Fonte: O Autor

Na Figura 22a, a linha dupla denota que um segmento da rede deve possuir dois nós (uma origem e um destino). Na Figura 22b, ilustra-se outra forma para o mesmo cenário apresentado na Figura 22a. Note que na Figura 22b, cada segmento deve ser ligado a dois nós por meio de relacionamentos espaciais “Toca”, tornando esta modelagem menos abstrata do que a apresentada na Figura 22a.

Figura 22 - Exemplos de como modelar uma rede Arco-Nó na SEER



Fonte: O Autor

4.3 Sintaxe Abstrata

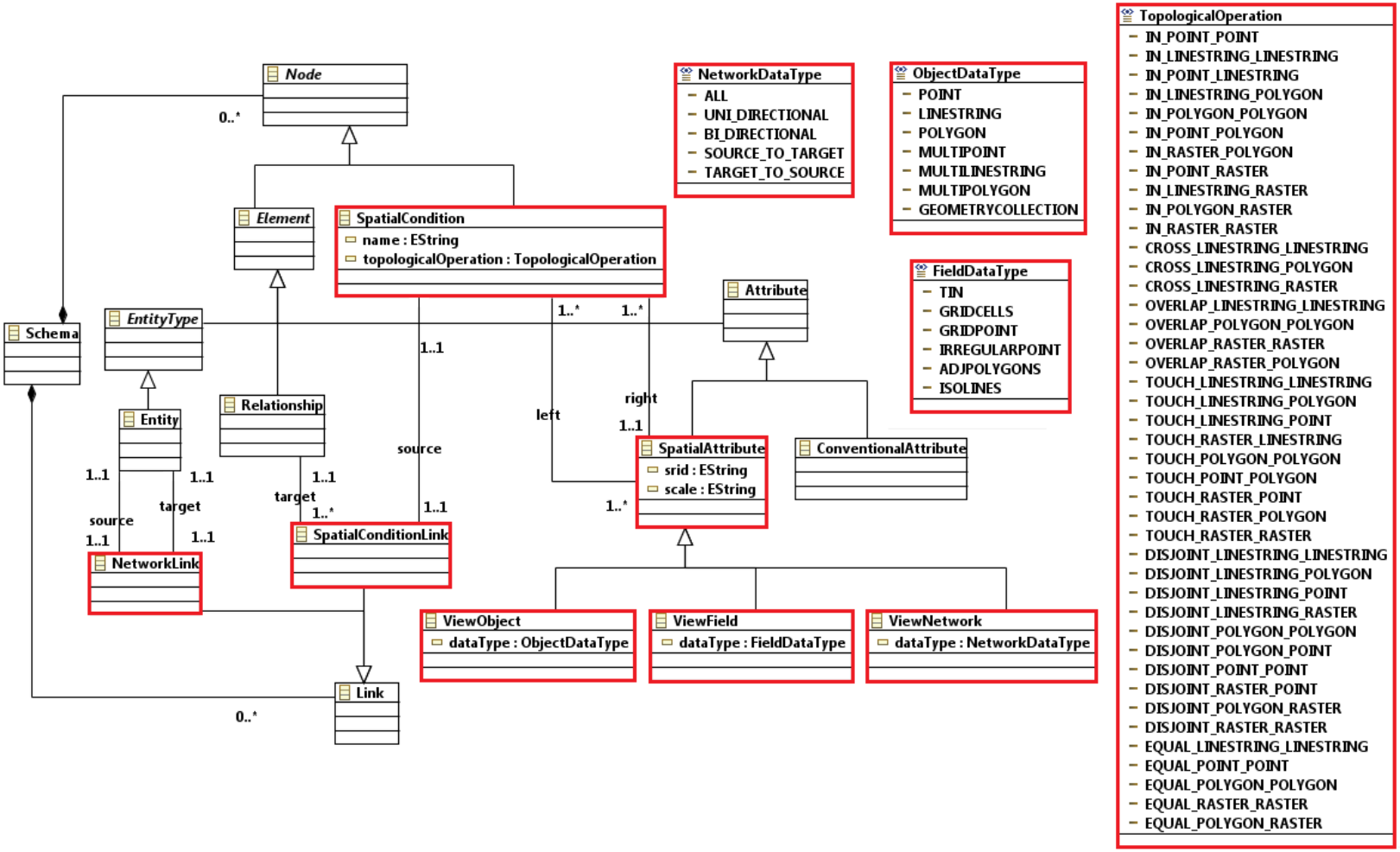
A sintaxe abstrata da SEER é definida a partir de uma extensão do metamodelo EERMM (cf. Subseção 2.4) a qual é denominada *Spatial* EERMM (SEERMM). Ou seja, o SEERMM descreve os conceitos da SEER e as relações válidas entre estes conceitos. Na Figura 23, apresentam-se as modificações que o SEERMM fez sobre o EERMM. Para isso, são criadas sete metaclasses e quatro enumerações, todas destacadas na Figura 23.

As metaclasses criadas são: "*SpatialAttribute*" (Atributo Espacial), "*ViewObject*" (Visão de Objetos), "*ViewField*" (Visão de Campo), "*ViewNetwork*" (Visão de Redes), "*SpatialConditionLink*" (Ligação de Condição Espacial), "*NetworkLink*" (Ligação de Redes), e "*SpatialCondition*" (Condição Espacial).

As enumerações criadas são: "*ObjectDataType*" (Tipo de Dado para Visão de Objeto), "*FieldDataType*" (Tipo de Dado para Visão de Campo), "*NetworkDataType*" (Tipo de Dado para Redes), e "*TopologicalOperation*" (Operações Topológicas).

A metaclasse "*Attribute*" é especializada em duas outras metaclasses: "*SpatialAttribute"* e "*ConventionalAttribute"*. Esta especialização visa: 1) separar os tipos espaciais dos convencionais; 2) impedir que um atributo convencional participe de uma condição espacial; 3) impedir que um atributo espacial seja identificador ou discriminador; e 4) permitir que apenas atributos espaciais tenham SRID e Escala. Por conseguinte, a metaclasse "*SpatialAttribute*" é especializada nas metaclasses "*ViewField*", "*ViewObject*" e "*ViewNetwork*". Cada uma dessas tem um atributo chamado "*dataType*", o qual é específico para o seu tipo de dado. Por exemplo, o "*dataType"* de *"ViewField"* deve ser do tipo *"FieldDataType".* Ressalta-se que os elementos das enumerações dos tipos de dados espaciais correspondem aos pictogramas da SEER vistos na Figura 17.

Figura 23 – Metamodelo SEERMM



Fonte: O Autor

A metaclasse “*SpatialCondition”* refere-se ao construtor de Condição Espacial, o qual contém os meta-atributos “*name*” e “*topologicalOperation*”, sendo o último do tipo “*TopologicalOperation*”. Igualmente as enumerações dos tipos de dados, os elementos da enumeração “*TopologicalOperation*” correspondem aos pictogramas da SEER exibidos na Figura 18. De modo a permitir que o lado esquerdo de uma Condição Espacial tenha vários atributos espaciais, tem-se a ligação “*left”,* com cardinalidade “1..*” entre “*SpatialCondition”* e “*SpatialAttribute”.* Por sua vez, para garantir que o lado direito só tenha um atributo espacial, tem-se a ligação *“right”* com cardinalidade “1..1”, entre “*SpatialCondition”* e “*SpatialAttribute”.*

Para indicar que uma Condição espacial deve ser aplicada em um Relacionamento, tem-se a metaclasse “*SpatialConditionLink*”. Note que as cardinalidades das ligações “*source*” e “target” é “1..1”, o que impede associar uma Condição Espacial a mais de um Relacionamento. Contudo, a cardinalidade oposta é “1..*”, sendo possível assim, conectar mais de uma Condição espacial a um Relacionamento.

Por fim, para dar suporte à modelagem de redes do tipo Arco-Nó, tem-se a metaclasse “*NetworkLink*” e as ligações “*source*” e “*target*” com a metaclasse “Entity”. Note que estas ligações denotam que uma rede é formada por apenas duas entidades, as quais não podem participar de outras redes.

4.4 Semântica Estática

A especificação de operações válidas no metamodelo SEERMM (cf. enumeração *TopologicalOperation* “Figura 23”) permite prover integridade espacial. Por consequência, é reduzido o número de regras para semântica estática. Na SEER, a Semântica Estática é formada por onze regras subdivididas em: Tipos de Dados e Topologia, Modelagem de Redes, Cartografia, Cardinalidade e Estruturação da Condição Espacial. Estas podem ser vistas a seguir:

Tipos de Dados e Topologia

1. **Verificação dos tipos de dados em Condições Espaciais.** Baseando-se na enumeração *“TopologicalOperation”* são especificados

quarenta e dois testes (um para cada elemento da enumeração), os quais garantem que, dada a escolha de uma operação topológica, seus tipos de dados são corretamente posicionados nos lados esquerdo e direito da operação. Por exemplo, a operação "(GeoLinha >> GeoPolígono)" deve conter, necessariamente, uma Linha do lado esquerdo e um Polígono do lado direito.

2. **Verificação dos tipos Visão de Campo em operações topológicas**. Com exceção do tipo Raster "*Grid of Cells*", os tipos que formam o conceito de Visão de Campo não participam de operação espacial. Estes tipos não possuem referências cartográficas reais que permitem representar um objeto no espaço geográfico. Tal característica é vista apenas no tipo Raster (e.g., imagem de satélite georreferenciada). Diante disso, não faz sentido permitir que os tipos que compõem a Visão de Campo (com exceção do "*Grid of Cells*"), participem de relacionamentos topológicos. Em função disso, é realizado um teste para evitar esse equívoco.

Modelagem de Redes

3. **Verificação de diferentes tipos de Rede em uma mesma entidade.** Uma vez que o tipo "Múltiplos" (cf. Figura 20e) de redes comporta todos os possíveis casos de direção, não faz sentido permitir que haja mais de um atributo para representação de redes em uma mesma entidade. Sendo assim, caso o projetista queira definir outro tipo, deve-se criar outra rede.

4. **Verificação dos tipos envolvidos em uma Rede.** Uma "Ligação de Rede" (*NetworkLink*) deve conter um atributo de Rede (*NetworkDatatype*) em uma entidade, e na outra entidade relacionada, deve conter apenas um atributo do tipo Ponto, configurando com isso uma rede Arco-Nó. Esta regra evita ambiguidade entre a ligação arco-nó por chaves, como também garante a quantidade mínima dos tipos envolvidos.

Cartografia

5. **Verificação de SRID diferentes**. Uma vez que objetos espaciais possuem SRID diferentes (cf. Subseção 2.1.1), eles não podem ser comparados, pois não comportam referências geográficas a partir de um mesmo “DATUM”, como também, as implementações não permitem comparações com SRID distintos.

6. **Verificação de escalas diferentes.** A captura de um objeto espacial em diferentes escalas pode ocasionar inconsistências em operações topológicas. Sendo assim, é realizada uma verificação das escalas envolvidas em um relacionamento espacial, com função apenas informativa.

Cardinalidade

7. **Verificação da operação “Contido” (*IN)* em relacionamentos 1:N.** Com exceção da operação “Contido” (*IN)*, todas as demais são comutativas. Assim, considerando apenas geometrias convexas, pode-se concluir que, em relacionamentos binários espaciais 1:N, os objetos espaciais do lado ”N” sempre devem estar contidos no objeto espacial do lado “1” e que, consequentemente, o contrário sempre é um erro.

8. **Verificação da operação “Contido” (*IN*) em relacionamento 1:1.** Considerando as premissas citadas na regra anterior para operação “Contido” (*IN*), em relacionamentos binários espaciais 1:1, os objetos espaciais do lado da participação “Total” (i.e., o lado que recebe a chave da relação) devem estar contidos no objeto espacial do lado com participação “Parcial”. Por exemplo, uma Câmara Municipal está contida em apenas um Município, e para ser cadastrada, deve-se informar necessariamente a chave de seu respectivo Município. Contudo, o Município não precisa obrigatoriamente, informar a Câmara Municipal para ser cadastrado.

Estruturação da Condição Espacial

9. **Verificação de objetos espaciais diferentes**. Os objetos espaciais que participam da Condição Espacial devem ser de entidades diferentes, com exceção do alto-relacionamento, pois não faz sentido

garantir geometrias que já estão na própria entidade (e.g., “ (**GeoEntidade1** >> **GeoEntidade1**)”).

10. **Verificação da origem dos objetos do lado esquerdo da Condição Espacial.** Com o objetivo de manter integridade e consistência em modelos complexos, uma Condição Espacial gerencia uma operação espacial entre duas entidades apenas. Com isso, o conjunto de objetos espaciais do lado esquerdo da Condição Espacial deve ser da mesma entidade. Um exemplo desse equívoco pode ser visto da seguinte forma: (e.g., “(**GeoEntidade1, GeoEntidade2** >> GeoEntidade3)”). Caso o projetista queira definir mais de um objeto de diferentes entidades do lado esquerdo da Condição Espacial, pode inserir uma outra condição como opção para isso.

11. **Verificação de contradições.** Um mesmo relacionamento não pode ter duas Condições Espaciais contendo os mesmos atributos, pois as operações topológicas são mutuamente exclusivas. Um exemplo pode ser visto da seguinte forma: (e.g., “(GeoLinha >> GeoPolígono)” e “ (GeoLinha >> GeoPolígono)”). Neste exemplo, verifica-se que a geometria GeoLinha não pode estar contida e ser disjunta ao mesmo tempo no GeoPolígono.

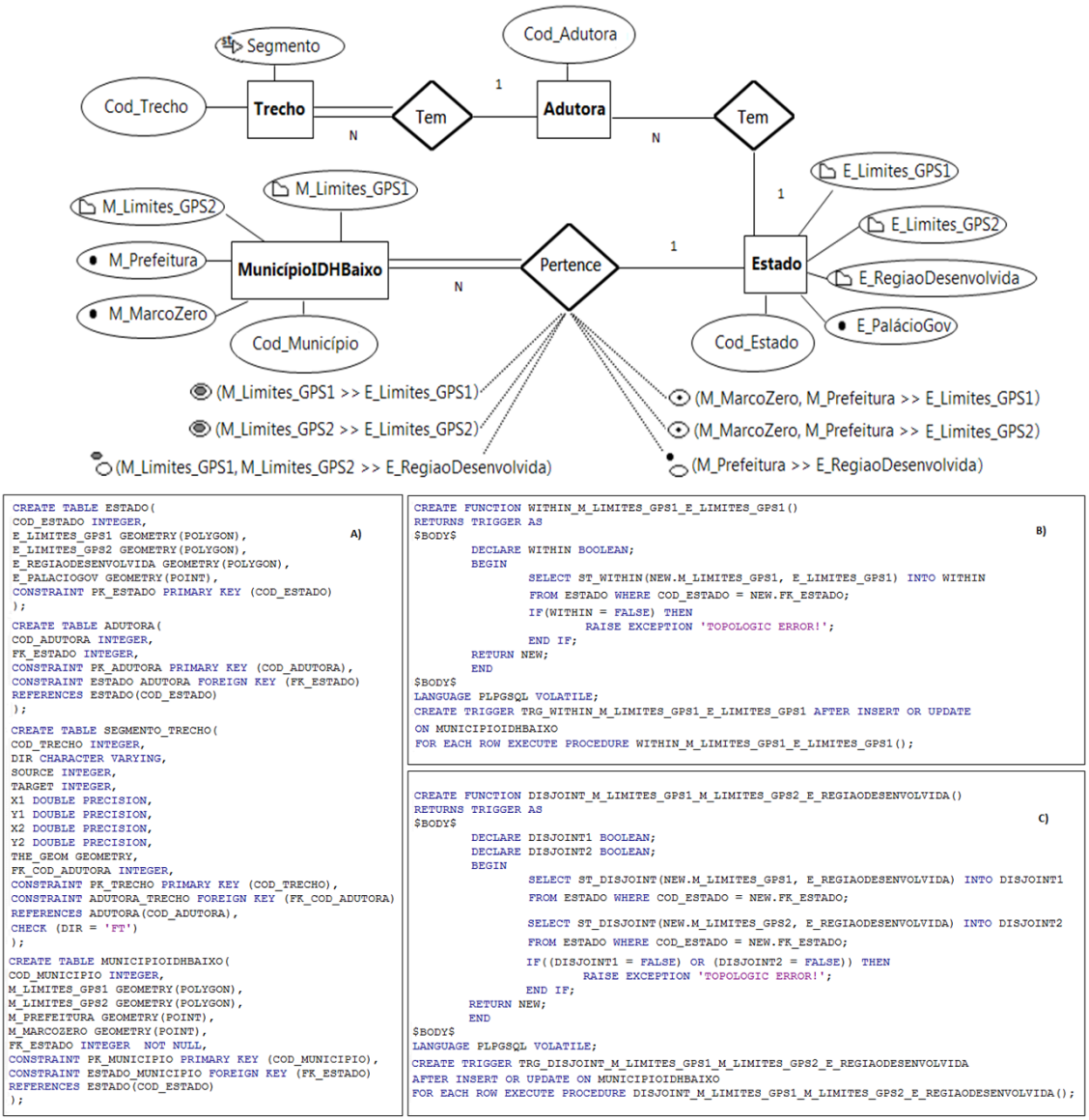
4.5 Transformações para SQL

Embora a geração de código esteja fora do escopo deste trabalho, na Figura 24, mostra-se um exemplo de comandos SQL (de acordo com a sintaxe do PostgreSQL 9.3/PostGis 2.0) que podem ser escritos para implementar o diagrama da Figura 19.

Na Figura 24a, verificam-se os comandos SQL para a criação das tabelas. Para a Rede, observa-se a tabela “SEGMENTO_TRECHO”, onde são definidos atributos para prover integridade e funcionalidades específicas da Rede (cf. Subseção 2.2.4). Já os comandos das Figura 24b e 29c são *triggers* que representam uma Condição Espacial cada. Embora o modelo visto na Figura 19 possua seis Condições Espaciais, são criadas apenas duas *Triggers* (i.e., evento disparado em forma de gatilho), pois com esses dois exemplos é possível replicar às outras. A *Trigger* da Figura 24b refere-se à Condição

Espacial “ (M_Limites_GPS1 >> E_Limites_GPS1) e a *trigger* da Figura 24c é referente à Condição Espacial “ (M_Limites_GPS1, M_Limites_GPS2 >> E_RegiaoDesenvolvida).

Figura 24 – Códigos SQL dos conceitos da SEER



A)

```
CREATE TABLE ESTADO(
COD_ESTADO INTEGER,
E_LIMITES_GPS1 GEOMETRY(POLYGON),
E_LIMITES_GPS2 GEOMETRY(POLYGON),
E_REGIAODESENVOLVIDA GEOMETRY(POLYGON),
E_PALACIOGOV GEOMETRY(POINT),
CONSTRAINT PK_ESTADO PRIMARY KEY (COD_ESTADO)
);
CREATE TABLE ADUTORA(
COD_ADUTORA INTEGER,
FK_ESTADO INTEGER,
CONSTRAINT PK_ADUTORA PRIMARY KEY (COD_ADUTORA),
CONSTRAINT ESTADO ADUTORA FOREIGN KEY (FK_ESTADO)
REFERENCES ESTADO(COD_ESTADO)
);
CREATE TABLE SEGMENTO_TRECHO(
COD_TRECHO INTEGER,
DIR CHARACTER VARYING,
SOURCE INTEGER,
TARGET INTEGER,
X1 DOUBLE PRECISION,
Y1 DOUBLE PRECISION,
X2 DOUBLE PRECISION,
Y2 DOUBLE PRECISION,
THE_GEOM GEOMETRY,
FK_COD_ADUTORA INTEGER,
CONSTRAINT PK_TRECHO PRIMARY KEY (COD_TRECHO),
CONSTRAINT ADUTORA_TRECHO FOREIGN KEY (FK_COD_ADUTORA)
REFERENCES ADUTORA(COD_ADUTORA),
CHECK (DIR = 'FT')
);
CREATE TABLE MUNICIPIOIDHBAIXO(
COD_MUNICIPIO INTEGER,
M_LIMITES_GPS1 GEOMETRY(POLYGON),
M_LIMITES_GPS2 GEOMETRY(POLYGON),
M_PREFEITURA GEOMETRY(POINT),
M_MARCOZERO GEOMETRY(POINT),
FK_ESTADO INTEGER  NOT NULL,
CONSTRAINT PK_MUNICIPIO PRIMARY KEY (COD_MUNICIPIO),
CONSTRAINT ESTADO_MUNICIPIO FOREIGN KEY (FK_ESTADO)
REFERENCES ESTADO(COD_ESTADO)
);
```

B)

```
CREATE FUNCTION WITHIN_M_LIMITES_GPS1_E_LIMITES_GPS1()
RETURNS TRIGGER AS
$BODY$
        DECLARE WITHIN BOOLEAN;
        BEGIN
                SELECT ST_WITHIN(NEW.M_LIMITES_GPS1, E_LIMITES_GPS1) INTO WITHIN
                FROM ESTADO WHERE COD_ESTADO = NEW.FK_ESTADO;
                IF(WITHIN = FALSE) THEN
                        RAISE EXCEPTION 'TOPOLOGIC ERROR!';
                END IF;
        RETURN NEW;
        END
$BODY$
LANGUAGE PLPGSQL VOLATILE;
CREATE TRIGGER TRG_WITHIN_M_LIMITES_GPS1_E_LIMITES_GPS1 AFTER INSERT OR UPDATE
ON MUNICIPIOIDHBAIXO
FOR EACH ROW EXECUTE PROCEDURE WITHIN_M_LIMITES_GPS1_E_LIMITES_GPS1();
```

C)

```
CREATE FUNCTION DISJOINT_M_LIMITES_GPS1_M_LIMITES_GPS2_E_REGIAODESENVOLVIDA()
RETURNS TRIGGER AS
$BODY$
        DECLARE DISJOINT1 BOOLEAN;
        DECLARE DISJOINT2 BOOLEAN;
        BEGIN
                SELECT ST_DISJOINT(NEW.M_LIMITES_GPS1, E_REGIAODESENVOLVIDA) INTO DISJOINT1
                FROM ESTADO WHERE COD_ESTADO = NEW.FK_ESTADO;

                SELECT ST_DISJOINT(NEW.M_LIMITES_GPS2, E_REGIAODESENVOLVIDA) INTO DISJOINT2
                FROM ESTADO WHERE COD_ESTADO = NEW.FK_ESTADO;
                IF((DISJOINT1 = FALSE) OR (DISJOINT2 = FALSE)) THEN
                        RAISE EXCEPTION 'TOPOLOGIC ERROR!';
                END IF;
        RETURN NEW;
        END
$BODY$
LANGUAGE PLPGSQL VOLATILE;
CREATE TRIGGER TRG_DISJOINT_M_LIMITES_GPS1_M_LIMITES_GPS2_E_REGIAODESENVOLVIDA
AFTER INSERT OR UPDATE ON MUNICIPIOIDHBAIXO
FOR EACH ROW EXECUTE PROCEDURE DISJOINT_M_LIMITES_GPS1_M_LIMITES_GPS2_E_REGIAODESENVOLVIDA();
```

Fonte: O Autor

4.6 Considerações Finais

Este Capítulo apresenta a extensão SEER, composta por sua sintaxe concreta (i.e., construtores e pictogramas), sintaxe abstrata (i.e., metamodelo SEERMM), semântica estática (i.e., formada por onze regras) e um exemplo de transformação para SQL dos seus conceitos (cf. Figura 24).

As quarenta e duas operações espaciais diretamente no metamodelo (i.e., enumeração *TopologicalOperations*), junto às regras de semântica estática (ver Seção 4.4), permitem integridade de operações topológicas, mesmo em relacionamentos entre entidades com múltiplas representações.

O construtor “Condição Espacial” (*SpatialCondition*) definido no metamodelo “SEERMM”, gerencia a operação espacial, a posição e tipo dos objetos espaciais envolvidos no relacionamento espacial; sendo possível usá-la para integridade de dados.

Com exceção da operação “Contido” (*IN*), as operações topológicas “Toca” (*Touch*), “Sobrepõe” (*Overlap*), “Igual” (*Equal*), “Disjunto” (*Disjoint*) e “Cruza” (*Cross*) são comutativas. Com isso, a SEER, diferente dos trabalhos relacionados, trata as seguintes operações: 1) Ponto Contido em Polígono, 2) Ponto Contido em Linha, 3) Linha Contida em Polígono, 4) Ponto Contido em Raster e 5) Linha Contida em Raster.

A SEER define cinco tipos diferentes de estruturas de redes (cf., Seção 4.2). Cada tipo de rede definido no modelo pelo usuário, pode ser definido em banco de dados em forma da restrição *CHECK*. A verificação é inserida na coluna que corresponde à direção do trecho em questão. Por exemplo, no trecho com sentido “Origem para Destino” são inseridos os caracteres “FT” (*FromTo*) na coluna “DIR” (i.e., direção).

5. Ferramenta SEERCASE e análise comparativa entre os trabalhos relacionados

Este Capítulo apresenta a ferramenta SEERCASE, bem como uma análise comparativa entre a SEER e os trabalhos relacionados. Ao final, é feita uma comparação conforme o conjunto de critérios definidos no Capítulo 3.

5.1 Implementação da Ferramenta SEERCASE

A ferramenta SEERCASE é uma implementação dos conceitos da SEER na ferramenta EERCASE (ALVES et al., 2014). O seu desenvolvimento é realizado sobre a plataforma *Eclipse Graphical Modeling Project* (GMP)[9], a qual fornece um conjunto de ferramentas necessárias para criação de editores gráficos baseados em *Eclipse Modeling Framework* (EMF). Este ambiente faz uso do XML *Metadata Interchange* (XMI) para armazenamento, manipulação, recuperação e intercâmbio de metadados. Dentre as principais linguagens neste ambiente, pode-se destacar as seguintes: *Eclipse Validation Language* (EVL) para validação semântica do modelo e a linguagem Emfatic (define um metamodelo em forma textual para EMF).

5.1.1. Eclipse Modeling Framework (EMF)

O EMF é um framework para construção de ferramentas CASE e outras aplicações baseadas em modelos de dados estruturados. A partir de uma especificação de um modelo descrito em XMI, o EMF fornece ferramentas e suporte em tempo de execução para produzir um conjunto de classes Java para o modelo e um conjunto de classes que habilitam a visualização e manipulação por meio de um editor básico.

O EMF (Core) consiste em três fundamentos:

1. **EMF:** Inclui o metamodelo Ecore[10] para descrever modelos e fornecer suporte em tempo de execução dos modelos, incluindo:

[9] http://www.eclipse.org/modeling/gmp/ - Acessado em 04/07/2015.
[10] http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html#details

notificação de alteração, suporte a persistência pelo padrão XMI e uma API para manipulação de objetos EMF.

2. **EMF.edit:** Inclui classes genéricas reusáveis para construir editores de modelos EMF. Este contém classes para fornecer conteúdo e *labels,* propriedades para suporte e outras classes que permitem visualizar modelos EMF usando padrões JFace.
3. **EMF.codegen:** Ferramenta capaz de gerar tudo o que é necessário para construir um editor completo para um modelo EMF.

5.1.2. Eclipse Validation Language (EVL)

A *Object Constraint Language* (OCL) é uma linguagem padrão para captura e definição de restrições em linguagens de modelagem. No entanto, esta linguagem possui limitações para modelos desenvolvidos com o uso do *framework* EMF. Posto isso, a linguagem EVL torna-se uma opção mais eficiente no lugar da OCL, sendo usada para avaliar e especificar restrições em modelos que utilizam objetos EMF. Na Figura 25, verificam-se as principais sentenças da sintaxe concreta da linguagem EVL.

Figura 25 – Sintaxe Concreta EVL

```
1 context <name> {
2       (guard (:expression)|({statementBlock}))?
3
4       (constraint|critique) <name> {
5
6       (guard (:expression)|({statementBlock}))?
7
8       (check (:expression)|({statementBlock}))?
9
10      (message (:expression)|({statementBlock}))?
11      }
12
13 }
```

Fonte: (KOLOVOS et al., 2010)

A sintaxe concreta da EVL vista na Figura 25 é formada por um contexto (linha 1) que deve satisfazer uma ou mais restrições/críticas (linha 4). A palavra reservada “*guard*” (linhas 2 e 6) refere-se a outras restrições definidas em outros contextos/restrições, e devem ser satisfeitas antes da restrição definida neste contexto. A palavra reservada “*check*” (linha 8) implica em uma restrição para o contexto em questão, retornando verdadeiro se for

satisfeita ou falso, caso contrário. A palavra reservada “*message*” (linha 10) refere-se a uma mensagem exibida para o usuário.

5.1.3. Linguagem Emfatic

Emfatic é uma linguagem de representação de modelos EMF que usa uma sintaxe textual similar a linguagem Java. Os construtores que compõem a SEER são definidos nesta linguagem no arquivo “.emf”, resultando no metamodelo SEERMM (cf. Figura 23). A partir do arquivo “.emf” é possível gerar o modelo Ecore (arquivo “.ecore”) e vice-versa, que corresponde aos elementos da ferramenta SEERCASE. Na Figura 26, verifica-se um exemplo de código Emfatic.

Figura 26 – Exemplo de Código Emfatic

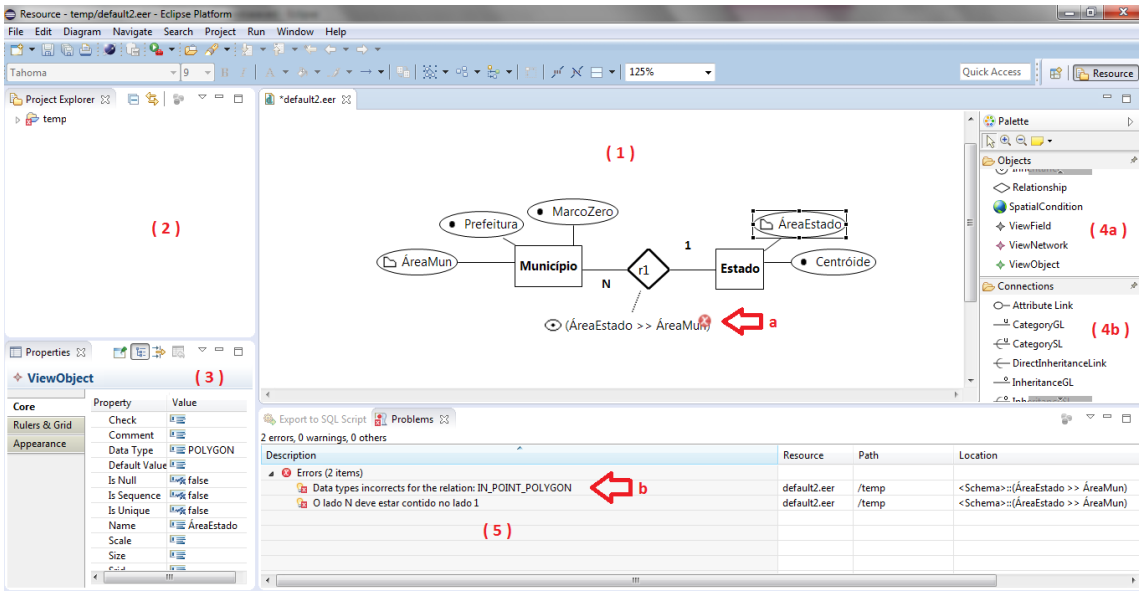
```
@namespace(uri="seer", prefix="seer")
package eer;

@gmf.diagram(foo="bar", rcp="false", onefile="true", diagram.extension="eer")
class Schema {
  val Node[*]#schema nodes;
  val Link[*]#schema links;
}

@gmf.node( label.icon="false", border.width="1", border.color="0,0,0")
abstract class Node {
  ref Schema[1]#nodes schema;
}

@gmf.node(label="label",  figure="ellipse" ,resizable="false", size="30,30")
class Inheritance extends Node {
  attr String label;
  attr DisjointnessType disjointness;
  ref InheritanceGL[1]#target inheritanceGL;
  ref InheritanceSL[*]#target inheritanceSL;
}
```

Fonte: O Autor

5.2 Utilização da Ferramenta SEERCASE

A ferramenta SEERCASE é composta por cinco áreas (Figura 27) que correspondem a funcionalidades distintas. Na área 1, tem-se o editor para projeto conceitual de BDE. Neste exemplo, verifica-se um modelo com dois erros (seta b) que ferem a semântica estática (cf. Seção 4.4), pois a área de um Estado não pode estar dentro da área do Município, sendo isto verificado pela cardinalidade. O segundo erro é referente aos tipos envolvidos na operação, uma vez que a operação denotada pelo pictograma "  " exige que sejam os tipos Ponto e Polígono. Os erros na SEERCASE são informados para o projetista em forma de texto (área 5, seta "b") e por pictograma vermelho em forma de "X" (área 1, seta "a").

Figura 27 - Ferramenta SEERCASE



Fonte: O Autor

Na área 2, observada na Figura 27, situa-se o "*Project Explorer*" onde estão localizados os artefatos do projeto gerados pelo projetista. Na área 3, apresenta-se a aba de configuração das propriedades do componente que está selecionado. Na área 4, exibe-se a paleta de objetos disponíveis, dividida em Objetos (4a) e Conexões (4b).

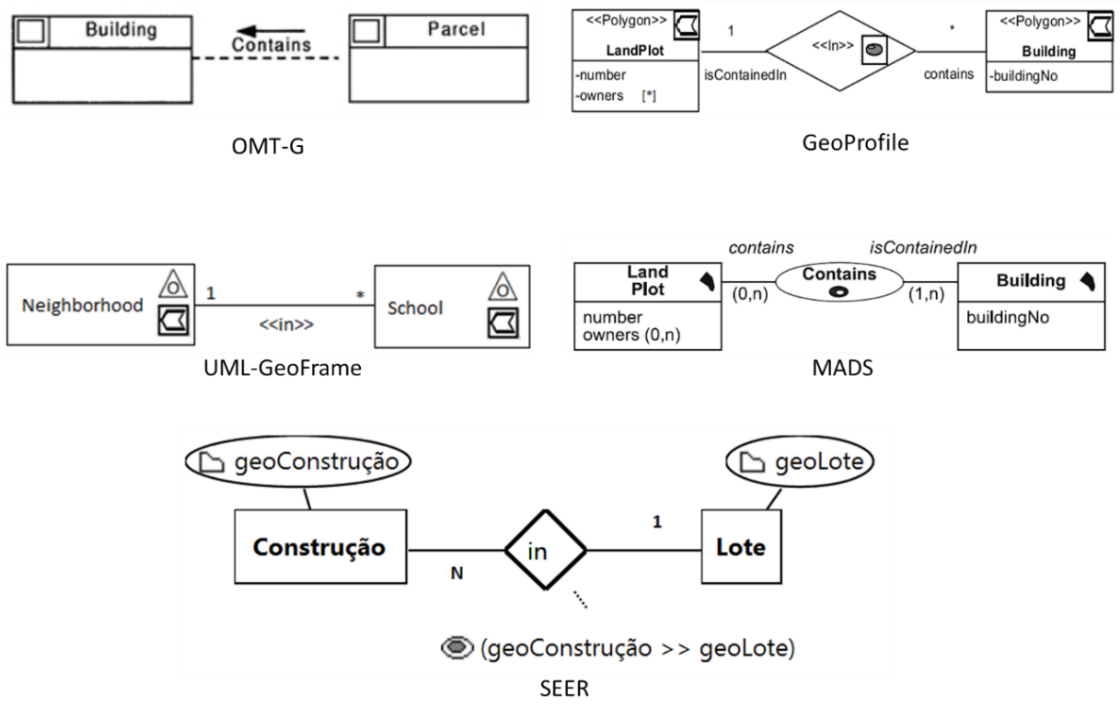
O conjunto de regras que forma a semântica estática da SEER (cf. Seção 4.4) é implementado na ferramenta SEERCASE utilizando a linguagem *Eclipse Validation Language* (EVL) (cf. Apêndice 2). Esta implementação pode

ser feita por OCL, contudo, optou-se por EVL por questões de compatibilidade com *framework* EMF. Com isso, ao salvar o projeto, a ferramenta trata os casos indevidos e informa para o projetista um *feedback* em forma de mensagem e pictograma.

5.3 Representações a partir dos trabalhos relacionados

A seguir, são mostrados vários exemplos que permitem comparar o trabalho realizado com os trabalhos relacionados. Na Figura 28, observam-se exemplos de modelagem do Relacionamento Espacial “Contido/Contém”.

Figura 28 - Relacionamento Espacial a partir dos Trabalhos Relacionados

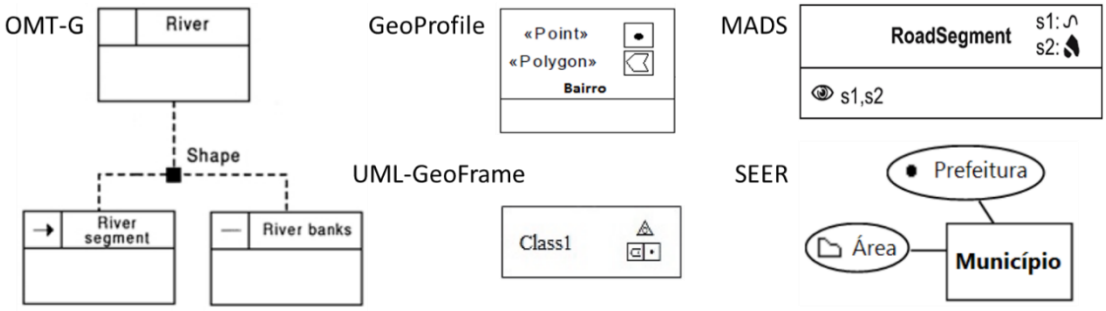


Fonte: O Autor

Na Figura 28, verificam-se que as linguagens OMT-G e UML-GeoFrame não possuem pictogramas para representar relacionamentos espaciais. Já no perfil GeoProfile e na linguagem MADS, a representação é feita por um pictograma com um círculo contendo um ponto, não denotando as formas participantes do relacionamento (i.e., Polígono e Polígono). Na SEER, a representação do Relacionamento espacial é feita por um pictograma que informa os tipos envolvidos.

Na Figura 29, ilustram-se exemplos de modelos contendo o conceito de Múltiplas Representações nas diferentes linguagens. Nesta figura, verifica-se que o modelo da linguagem OMT-G representa o conceito de Múltiplas Representações pelo conceito de superclasse e subclasse. Este método, embora expresse a múltipla representação, cria uma tabela para cada tipo espacial. Isto é, a quantidade de classes em OMT-G é maior que a quantidade de classes nas outras linguagens para o mesmo modelo.

Figura 29 - Múltiplas Representações a partir dos Trabalhos Relacionados



Fonte: O Autor

No exemplo de Múltiplas Representações do UML-GeoFrame exibido na Figura 29, tem-se a utilização de um triângulo para informar o tipo da entidade (i.e., Visão de Objeto e/ou Visão de Campo e/ou Redes), junto a representações do tipo. Com isso, há uma redundância de informação (classe que corresponde à visão de objeto), uma vez que esta informação pode ser obtida apenas pelos tipos.

No exemplo de Múltiplas Representações do GeoProfile mostrado na Figura 29, verifica-se que este perfil representa duas formas para uma mesma geometria (i.e., nome do tipo em forma descritiva e o pictograma). No entanto, essa duplicidade de informação é vista apenas em algumas ferramentas (e.g., *Visual Paradigm*).

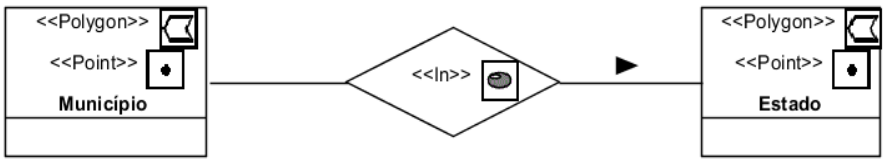
No exemplo de Múltiplas Representações do MADS visto na Figura 29, verifica-se que este utiliza visões (s1 e s2), que se relacionam com os atributos conforme regras do negócio.

No exemplo de Múltiplas Representações da SEER visto na Figura 29, verifica-se que é possível inserir mais de um tipo de dado espacial na mesma

entidade, como também, personalizar os nomes dos atributos e, se necessário, inserir mais de um atributo com o mesmo tipo.

Nas Figura 30, 35, 36 e 37 observam-se modelos contendo o conceito de Relacionamento Espacial entre Entidades georreferenciadas com Múltiplas Representações cada uma. É importante ressaltar esse tipo de modelagem devido à ambiguidade causada no relacionamento espacial e os tipos de dados que participam da operação topológica em questão. Este conceito não foi encontrado para o OMT-G e, portanto, foi omitido.

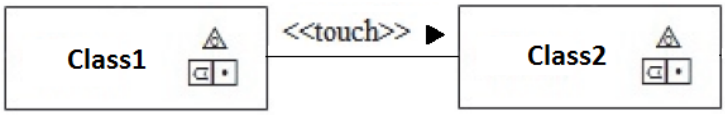
Figura 30 - Relacionamento Espacial entre Entidades com múltiplas Representações em GeoProfile



Fonte: (GEOPROFILE, 2015)

Na Figura 30, observa-se um exemplo de relacionamento espacial entre entidades com múltiplas representações, utilizando o GeoProfile. Neste exemplo, o “Município” está Contido em “Estado”. No entanto, não fica explícito qual geometria participa da operação “Contido”; se é o ponto ou o polígono do município.

Figura 31 - Relacionamento Espacial entre Entidades com múltiplas Representações em UML-GeoFrame



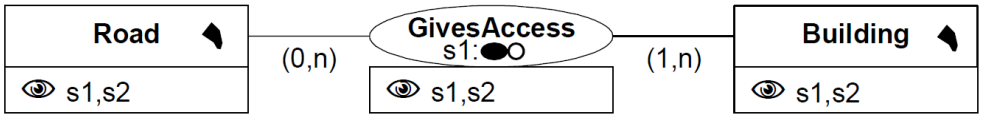

```
context Class1
inv: Class1.area.touch.Classe2.area
inv: Class1.area.touch.Classe2.point
inv: Class1.point.touch.Classe2.area
inv: Class1.point.in.Classe2.point
```

Fonte: (RIBEIRO et al., 2013)

Na Figura 31, observa-se um exemplo de Múltipla Representação para a linguagem UML-GeoFrame. Nela, observam-se duas classes com duas geometrias (Polígono e Ponto) relacionadas por meio de um relacionamento espacial “Toca”. Similar ao GeoProfile, não é possível identificar qual das

geometrias se relacionam. Contudo, para evitar ambiguidade na informação, a linguagem define uma expressão em OCL que permite ao projetista descrever o relacionamento válido. No exemplo da Figura 31, embora a OCL denote os casos válidos entre as geometrias, não é possível identificar se é apenas a Área que toca a Área, se é a Área que toca o Ponto, ou se ocorrem as duas situações ao mesmo tempo. Assim também, esta regra é imposta pelo projetista no momento do desenvolvimento.

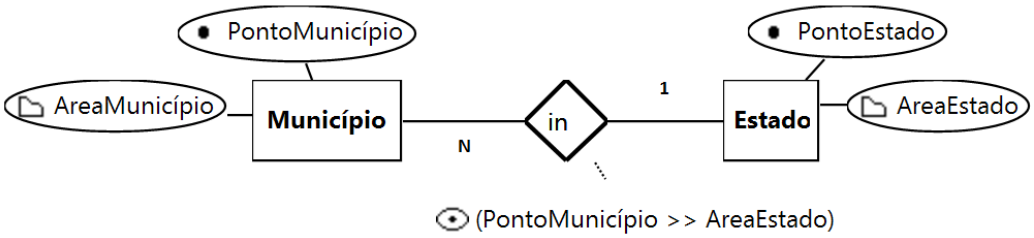
Figura 32 - Relacionamento Espacial entre Entidades com múltiplas Representações em MADS



Fonte: (PARENT; SPACCAPIETRA; ZIMÁNYI, 2006a)

Na Figura 32, apresenta-se um exemplo de Múltiplas Representações na linguagem MADS. Nela, fica claro que apenas as geometrias que correspondem a visão “s1” se tocam. No entanto, se for definida uma outra geometria para visão s1 (e.g., uma Linha), ocasiona também uma ambiguidade. Este problema é tratado na SEER (Figura 33), onde fica explícito qual o tipo geométrico, o sentido e a operação do relacionamento espacial entre entidades com múltiplas representações. Desta forma é possível evitar a ambiguidade vista em modelagem de relacionamentos espaciais envolvendo múltiplas representações.

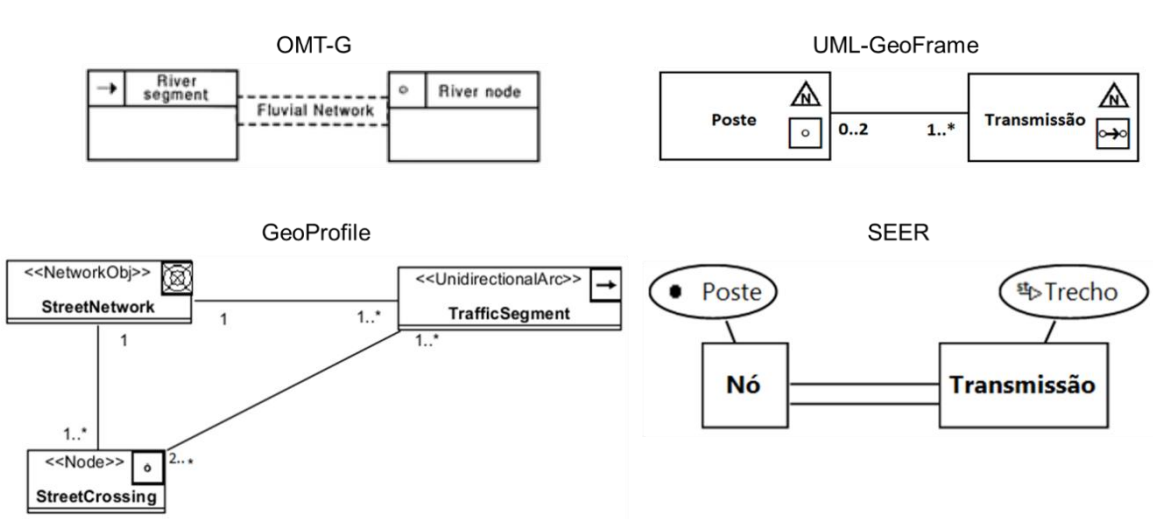
Figura 33 - Relacionamento Espacial entre Entidades com múltiplas Representações em SEER



Fonte: O Autor

Na Figura 34, observam-se modelagens de Redes nos trabalhos relacionados juntamente com a SEER. Com exceção da linguagem MADS, todas as outras linguagens permitem modelagem de Redes. Nesta figura, observa-se que a linguagem OMT-G define uma Rede por uma ligação tracejada dupla entre duas entidades que possuem os tipos Arco e Nó. Nesse exemplo, o arco é Unidirecional, podendo ser Bidirecional dependendo do objeto modelado (cf. Capítulo 3).

Figura 34 - Modelagens de Redes em Diferentes Linguagens



Fonte: O Autor

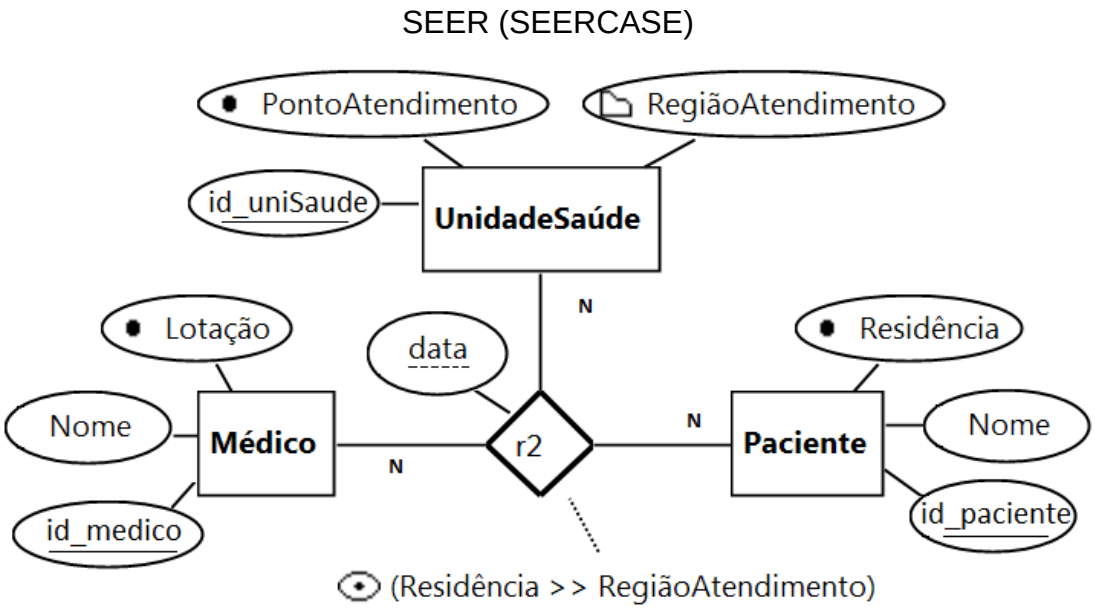
Na Figura 34, verifica-se um exemplo para modelagem de Redes com o perfil GeoProfile. Nele, mostra-se uma rede formada por arcos representando as ruas (*TrafficSegment*) e nós representando os cruzamentos entre elas (*StreetCrossing*). Este modelo é o único que precisa de três estereótipos para representar uma estrutura de rede. Por sua vez, no modelo de Rede da Figura 34 com a linguagem UML-GeoFrame, têm-se arcos representados pelos

trechos das linhas de transmissão e nós representados pelos postes, onde um trecho de transmissão é ligado a dois postes e um poste pode conter vários trechos ligados a este. Ainda na Figura 34, observa-se uma modelagem de Redes na linguagem SEER, em que duas entidades (Nó e Transmissão) são ligadas entre si por meio de uma linha dupla (*Link* de Redes). Este exemplo abstrai a necessidade de especificar as cardinalidades.

5.4 Representações a partir da SEER

A fim de expressar a SEER a partir de uma modelagem própria, é desenvolvido um exemplo envolvendo um relacionamento ternário, com múltiplas representações e representações geográficas nas três entidades envolvidas. Cada linguagem de modelagem desenvolve o modelo mostrado na Figura 35 através de sua respectiva ferramenta CASE.

Figura 35 - Relacionamento Ternário em SEER



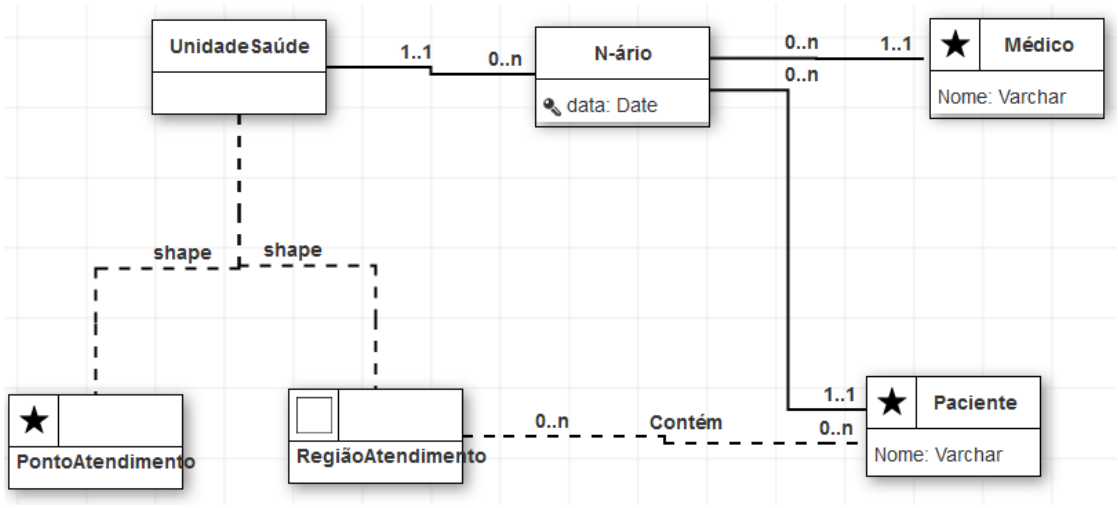
Fonte: O Autor

No exemplo da Figura 35, tem-se um Relacionamento Ternário com as entidades: UnidadeSaúde, Médico e Paciente. A “UnidadeSaúde” contém uma representação geográfica do tipo “Ponto” que corresponde à localização geográfica do atendimento (PontoAtendimento), bem como uma representação do tipo Polígono para a região (RegiãoAtendimento) atendida por esta unidade de saúde. A entidade “Paciente” é formada por uma localização geográfica representada por um tipo “Ponto”, e o nome do paciente (*String*) que recebeu

atendimento. A entidade “Médico” é formada por uma representação geográfica do tipo Ponto para localização da lotação do médico que realizou o procedimento, junto ao atributo do tipo *String* para o seu nome. O relacionamento “r2” possui um atributo descritivo do tipo data que corresponde ao dia que houve o atendimento, bem como uma condição espacial indicando que a residência do paciente deve estar dentro da região de atendimento da unidade de saúde. Nas Figura 36, 41 e 42 são observados exemplos que correspondem ao modelo da Figura 35, sendo desenvolvidas pelas ferramentas dos trabalhos relacionados.

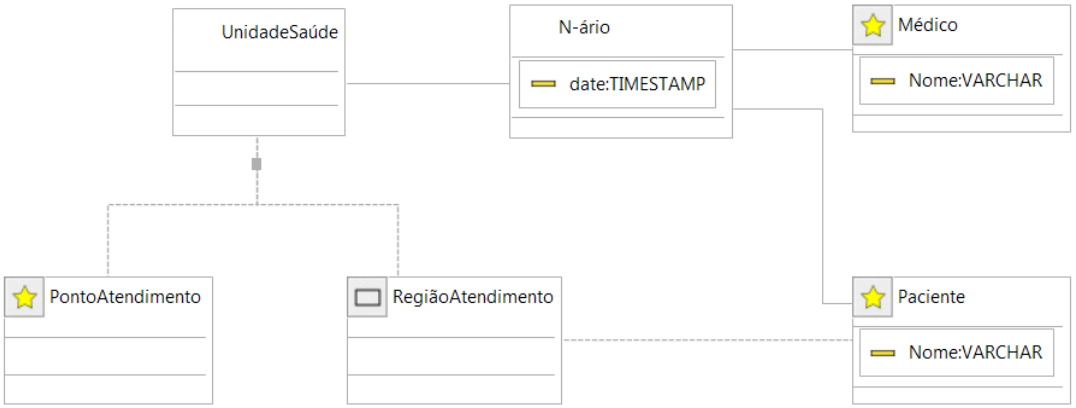
Nas Figura 36 e 41, observa-se que o OMT-G não dá suporte a relacionamentos n-ários, embora esse tipo de associação seja previsto pela linguagem OMT (RUMBAUGH et al., 1991). Diante dessa limitação, no exemplo modelado é criada uma tabela como meio de relacionar as três tabelas participantes (N-ário) com cardinalidade NxNxN.

Figura 36 – Exemplo em OMT-G (OMT-G *Design WEB*)



Fonte: O Autor

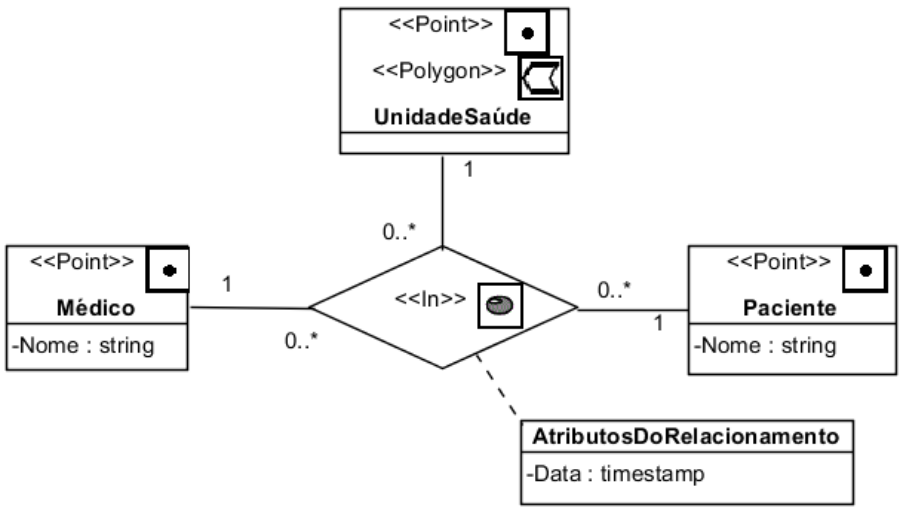
Figura 37 – Exemplo em OMT-G (OMT-G *Design Desktop*)



Fonte: O Autor

No OMT-G, a múltipla representação é feita pelo conceito de generalização, similar ao conceito de herança que, no exemplo, são definidas como duas sub entidades (i.e., PontoAtendimento e RegiãoAtendimento) para garantir as representações dos diferentes tipos. Nas Figura 36 e 41, observa-se que para representar o relacionamento espacial (Residência dentro Região Atendimento), é feita uma ligação entre a sub entidade “RegiãoAtendimento” e “Paciente”. Com isso, é criada uma relação entre chaves no banco de dados de forma não prevista no modelo da Figura 35.

Figura 38 – Exemplo em GeoProfile (*Visual Paradigm*)



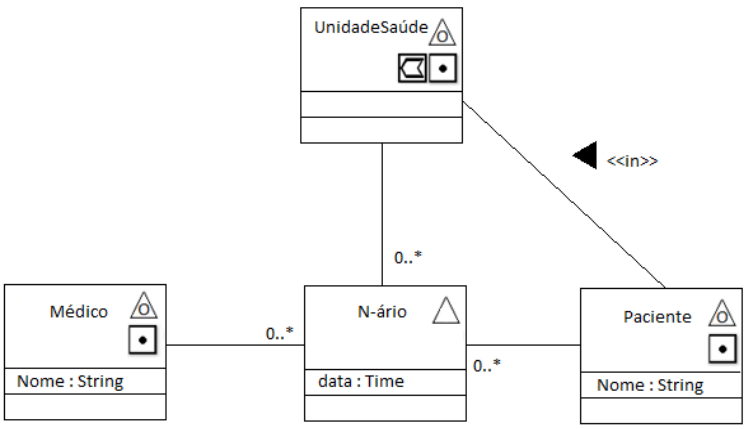
Fonte: O Autor

Na Figura 38, observa-se uma modelagem com o perfil GeoProfile correspondente ao modelo identificado na Figura 35. Este modelo foi

desenvolvido na ferramenta *Visual Paradigm,* versão *Professional Edition* 12.1. Verifica-se que o GeoProfile produz um relacionamento ternário de forma similar ao visto pela SEER, sem perdas na estruturação. Contudo, verifica-se neste exemplo que o GeoProfile não deixa claro qual o ponto que está contido pela “RegiaoAtendimento”, se é o ponto da lotação do médico ou o endereço do Paciente. A ambiguidade aumenta se for inserida uma segunda representação geográfica para o endereço do Paciente (e.g., Polígono). Desta forma, não fica explícito qual dos polígonos faz parte do relacionamento espacial. Assim, também não é possível nomear as geometrias das classes.

Na Figura 39, observa-se uma modelagem com a linguagem UML-GeoFrame correspondente ao modelo identificado na Figura 35. O modelo mostrado na Figura 39 é projetado utilizando a ferramenta ArgoCaseGeo versão 3.0. Nele, verifica-se que o relacionamento ternário é representado com a criação de uma quarta tabela (N-ário), embora este conceito seja previsto pelo diagrama de classe da UML.

Figura 39 – Exemplo em UML-GeoFrame (ArgoCaseGEO)

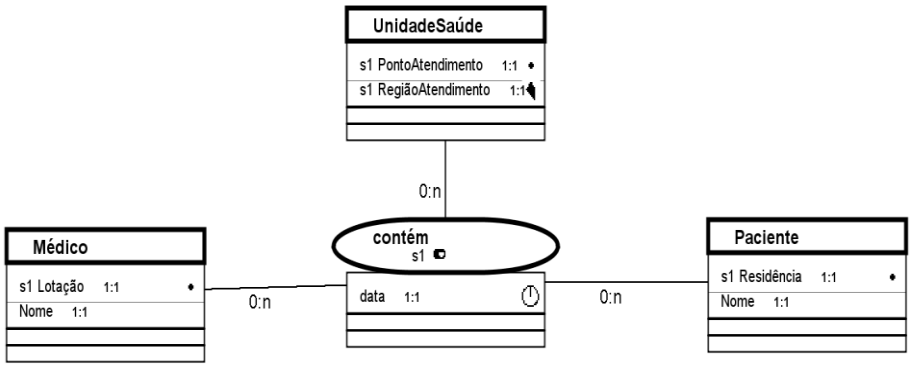


Fonte: O Autor

Para representar o relacionamento espacial em UML-GeoFrame (Figura 43), é feita uma ligação entre o paciente e a unidade de saúde como um artifício, implicando em uma relação não prevista no exemplo da Figura 35. Outra observação é a ambiguidade para o relacionamento espacial “*IN*”, pois não fica claro qual geometria participa dessa relação, se a geometria do endereço do paciente está dentro da área de atendimento, se está dentro do ponto de atendimento ou se ambas. Para isso, pode-se inserir uma OCL

(RIBEIRO et al., 2013) e deixar explícita a relação. Contudo, nessa abordagem há uma margem para erros, uma vez que o projetista impõe a regra manualmente. Na Figura 40, verifica-se um exemplo utilizando a linguagem MADS correspondente ao modelo visto na Figura 35 feito pela SEER.

Figura 40 – Exemplo em MADS (MADS *Editor* 2.2)



Fonte: O Autor

Na Figura 40, observa-se que linguagem MADS trata o modelo de forma similar ao visto na SEER (Figura 35). Contudo, o MADS não deixa claro qual dupla de geometrias participa da operação “Contém”, pois todas as geometrias que participam do relacionamento fazem parte do grupo de representações da visão “s1”.

5.5 Considerações Finais

No Quadro 9, verifica-se um resumo das características dos trabalhos relacionados juntamente com a SEER.

Quadro 11 - Avaliação final dos trabalhos relacionados junto ao SEER

Conceitos \ Trabalhos	OMT-G	GeoProfile	UML-GeoFrame	MADS	SEER
Visão de Campo	SIM	SIM	SIM	NÃO	SIM
Visão de Objeto	SIM	SIM	SIM	SIM	SIM
Múltiplas Representações	SIM	SIM	SIM	SIM	SIM
Relacionamento Topológico	SIM	SIM	SIM	SIM	SIM
Pictograma em Classe	SIM	SIM	SIM	SIM	NÃO
Pictograma em Atributo	NÃO	NÃO	NÃO	SIM	SIM
Modelagem de Redes	SIM	SIM	SIM	NÃO	SIM
Modelagem Temporal	NÃO	SIM	SIM	SIM	NÃO
Verificação topológica	NÃO	PARCIAL	NÃO	NÃO	SIM
Representatividade topológica	NÃO	NÃO	NÃO	NÃO	SIM
Representatividade de Redes	NÃO	NÃO	NÃO	NÃO	SIM
Representatividade de Coleções Geométricas	NÃO	NÃO	NÃO	SIM	SIM
Nomeação de Atributos Espaciais	NÃO	NÃO	NÃO	SIM	SIM
Repetição de Tipo Espacial	NÃO	NÃO	NÃO	SIM	SIM
Desambiguidade de operação	NÃO	NÃO	NÃO	NÃO	SIM
Operação Topológica sobre Raster	NÃO	NÃO	NÃO	NÃO	SIM
Relacionamentos N-ários espaciais	NÃO	PARCIAL	NÃO	PARCIAL	SIM

Fonte: O Autor

Verifica-se no Quadro 9 que a SEER não possui pictogramas em classe, como também o conceito de modelagem temporal. A SEER não permite pictograma na classe por uma questão de opção, pois, esta abordagem não permite especificar uma descrição ao tipo espacial, ao contrário da abordagem escolhida (i.e., Pictogramas em Atributos), onde é possível especificar tanto uma descrição para o tipo espacial, como também especificar duas vezes o

mesmo tipo. A SEER não implementa o conceito de Modelagem Temporal por ficar entendido que, para ser garantido, deve-se levar em consideração fatores de implementação que transcendem a especificação de pictogramas apenas.

Ainda no Quadro 9, verifica-se que as contribuições da SEER estão presentes na especificação e implementação dos seguintes conceitos: Verificação Topológica, Representatividade Topológica, Representatividade de Redes, Desambiguidade de Operação, Operação Topológica em Raster e Relacionamentos N-ários Espaciais. Com exceção do MADS, outros importantes conceitos não são suportados pelas linguagens e são definidos pela SEER: Representatividade de Coleções Geométricas, Nomeação de Atributos Espaciais e Repetição de Tipos Espaciais.

6. Conclusão

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

6.1 Considerações Finais

Esta dissertação propõe uma extensão da linguagem EER para modelagem conceitual de BDE. Tal extensão chama-se SEER e é implementada no protótipo SEERCASE. Dentre os pontos fortes da SEER, pode-se citar o suporte para modelar, sem ambiguidade, Múltiplas Representações e Relacionamentos Espaciais. Vale destacar que é definido um conjunto básico de regras que visa reduzir a especificação equivocada de condições espaciais em relacionamentos topológicos. Para isso, são adicionadas características espaciais ao metamodelo EERMM (o metamodelo estendido chama-se SEERMM) e definidas regras para a semântica estática da linguagem SEER. Essas regras partem do conjunto mínimo de relacionamentos topológicos válidos definidos por Clementini (CLEMENTINI; DI FELICE; VAN OOSTEROM, 1993).

6.2 Contribuições

Uma extensão espacial para a linguagem EER. A SEER permite modelar projetos de BDE com integridades espaciais a partir da notação de Elmasri e Navathe (ELMASRI e NAVATHE, 2011).

Uma extensão espacial do metamodelo EERMM. O SEERMM é uma extensão espacial do EERMM. Essa extensão inclui a inserção dos relacionamentos topológicos válidos diretamente em metamodelo, atributos espaciais e modelagem de redes (cf. Figura 23).

Regras que visam evitar erros básicos de semântica estática durante a modelagem conceitual do BDE. Dado que um metamodelo apenas impede erros sintáticos, são definidas onze regras básicas de semântica estática que visam reduzir equívocos envolvendo tipos de dados, construções de redes, conceitos de cartografia, cardinalidade e estruturação da condição espacial.

Pictogramas para expressar precisamente os tipos de dados espaciais que são válidos em condições espaciais topológicas. Os pictogramas encontrados na literatura não denotam precisamente quais são os tipos de dados espaciais envolvidos em uma operação espacial. Para superar esta limitação, são definidos quarenta e dois pictogramas para este fim (cf. Seção 4.2).

Definição do construtor "Condição Espacial". Dado que uma Condição espacial para ser definida precisa que sejam especificados: duas geometrias (no mínimo), uma operação topológica, o sentido que a operação topológica deve ser testada, um SRID e uma Escala, o construtor "Condição Espacial" é criado para gerenciar estas informações. Ademais, vale ressaltar que este construtor também evita especificações de relacionamentos espaciais ambíguos quando as entidades envolvidas têm Múltiplas Representações Espaciais.

Um conjunto de construtores para modelagem de Redes. São definidos cinco tipos diferentes de trechos de Redes (da origem para destino, do destino para origem, bidirecional, unidirecional e múltipla). Estes tipos formam os atributos de Redes inseridos no metamodelo SEERMM (enumeração *NetworkDataType*) e são usados para definir uma rede com restrições e direção.

Desenvolvimento do protótipo SEERCASE. Para a implementação dos conceitos, foi desenvolvido um protótipo de ferramenta CASE chamado SEERCASE. Esta implementação é feita utilizando tecnologias de MDD para construção de ferramentas CASE (cf. Seção 5.1).

6.3 Trabalhos Futuros

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

- **Inserir os conceitos de modelagem Temporal.** Um estudo voltado para fatores temporais, levando em consideração que esses conceitos impactam diretamente em possíveis regras estruturais não convencionais;

- **Verificar e implementar outros relacionamentos espaciais**. Faz-se necessário um estudo para implementação de relacionamentos espaciais métricos e direcionais vistos em (RIGAUX; SCHOLL; VOISARD, 2001);
- **Implementar na SEERCASE um módulo para, dado um diagrama SEER, gerar código SQL.** Realizar um estudo sobre as diferentes tecnologias de SGBDE e como fazer tradutores de diagramas em código executável/interpretável.
- **Implementar na SEERCASE um módulo de engenharia reversa**. Desenvolver um estudo para gerar modelos espaciais conceituais a partir de estruturas físicas de banco de dados.

Referências

ALVES, E. et al. EERCASE: uma ferramenta para apoiar o estudo do projeto conceitual de banco de dados. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2014, Dourados. **Anais**... [S.l.]: [s.n.], 2014. v. 3, n. 1.

BÉDARD, Y; LARRIVÉE, S; PROULX, M. Modeling geospatial databases with plug-ins for visual languages: a pragmatic approach and the impacts of 16 years of research and experimentations on perceptory. In: CONCEPTUAL modeling for advanced application domains. [S.l.]: Springer, 2004. p. 17–30.

BORGES, K. A. V.; DAVIS, C. A.; LAENDER, A. H. F. OMT-G: an object-oriented data model for geographic applications. **GeoInformática**, [S.l.], v. 5, n. 3, p. 221– 260, 2001.

BRAMBILLA, M.; CABOT, J.; WIMMER, M. Model-driven software engineering in practice. **Synthesis Lectures on Software Engineering**, [S.l.], v. 1, n. 1, p. 1–182, 2012.

CÂMARA, G. **Anatomia de sistemas de informação geográfica**. UNICAMP: Instituto de Computação, 1996.

CASANOVA, M. A et al. Bancos de dados geográficos. **MundoGEO**, Curitiba, 2005.

CHEN, P. P-S. The entity-relationship model-toward a unified view of data. **ACM Transactions on Database Systems (TODS)**, [S.l.], v. 1, n. 1, p. 9–36, 1976.

CLEMENTINI, E.; DI FELICE, P.; VAN OOSTEROM, P. A small set of formal topological relationships suitable for end-user interaction. In: ADVANCES in Spatial Databases. Berlin: Springer, 1993. p. 277-295.

EGENHOFER, M. J.; FRANZOSA, R. D. Point-set topological spatial relations. **International Journal of Geographical Information System**, [S.l.], v. 5, n. 2, p. 161–174, 1991.

EGENHOFER, M. J.; HERRING, J. Categorizing binary topological relations between regions, lines, and points in geographic databases. **The 9**, [S.l.], v. 9, p. 91–94, 1990.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. [S.l.]: Pearson Education, 2011.

FERREIRA, T. B.; STEMPLIUC, S. M.; LISBOA FILHO, J. Geographical data modeling with the UML geoprofile and MDA transformations on the enterprise architect tool. In: IBERIAN CONFERENCE ON INFORMATION SYSTEMS AND TECHONOLOGIES, 9., 2014, Barcelona. **[Trabalhos apresentados]**... [S.l.: s.n.], 2014.

FIDALGO, R. D. N. et al. EERMM: a metamodel for the enhanced entity-relationship model. In: CONCEPTUAL Modeling. [S.l.]: Springer, 2012. p. 515–524.

FIDALGO, R. D. N. et al. Metamodeling the enhanced entity-relationship model. **Journal of Information and Data Management**, [S.l.], v. 4, n. 3, p. 406, 2013.

FRANK, A. U.; TIMPF, S. Multiple representations for cartographic objects in a multi-scale tree: an intelligent graphical zoom. **Computers & Graphics**, [S.l.], v. 18, n. 6, p. 823–829, 1994.

GEOPROFILE: um perfil UML para modelagem conceitual de banco de dados geográficos. Viçosa: UFV, 2015. Disponível em: <http://www.dpi.ufv.br/projetos/geoprofile/>

GOODCHILD, M. F. Geographical data modeling. **Computers & Geosciences**, [S.l.], v. 18, n. 4, p. 401–408, 1992.

HEUSER, C. A. **Projeto de banco de dados**. Porto Alegre: Sagra Luzzatto, 2001.

KEMP, K. K. **Environmental modeling with GIS:** a strategy for dealing with spatial continuity. 1993. Tese (Doutorado)- Department of Geography, University of Santa Barbara, Santa Barbara, 1993. NCGIA Technical Report 93-3.

KOLOVOS, D. et al. The épsilon book. **Structure**, [S.l.], v. 178, p. 1–10, 2010.

KOSTERS, G.; PAGEL, B.-U.; SIX, H.-W. GIS: application development with GeoOOA. **International Journal of Geographical Information Science**, [S.l.], v. 11, n. 4, p. 307–335, 1997.

LAURINI, R.; THOMPSON, D. **Fundamentals of spatial information systems**. [S.l.]: Academic press, 1992. v. 37.

LISBOA FILHO, J. et al. Design and implementation of the valid time for spatio-temporal databases. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, 9., 2007, Madeira, Portugal. **[Trabalhos apresentados]**... [S.l.: s.n.], 2007. p. 569-573.

LISBOA FILHO, J.; IOCHPE, C. Modeling with a UML profile. In: ENCYCLOPEDIA of GIS. [S.l.]: Springer, 2008. p. 691–700.

LISBOA FILHO, J.; JÚNIOR RODRIGUES, M. F.; DALTIO, J. **ArgoCASEGEO**: uma ferramenta CASE de código-aberto para modelo UML-geoframe. Viçosa: UFV/DPI, 2004.

LISBOA FILHO, J. et al. Domain and Model Driven Geographic Database Design. In: DOMAIN Engineering. [S.l.]: Springer, 2013. p. 375–399.

LIZARDO, L. E. O.; DAVIS JR, C. A. OMT-G Designer: a web tool for modeling geographic databases in OMT-G. In: ADVANCES in Conceptual Modeling. [S.l.] Springer, 2014. p. 228–233.

LONGLEY, P. A. et al. **Sistemas e ciência da informação geográfica**. [S.l.]: Bookman, 2013.

MARTÍNEZ, Á. O. T.; FROZZA, A. A. OMT-G Design: uma ferramenta para modelagem de dados espaciais. In: ESCOLA REGIONAL DE BANCO DE DADOS, 10., 2014, São Francisco do Sul. **[Trabalhos apresentados]**... [S.l.: s.n.], 2014.

OBE, R.; HSU, L. **PostGIS in action**. [S.l.]: Manning Publications, 2011.

OCL. **Object Constraint Language**, 2015. Disponível em: <http://www.omg.org/spec/OCL/2.3.1/PDF/>

PARENT, C. et al. Modeling spatial data in the MADS conceptual model. In: SYMPOSIUM ON SPATIAL DATA HANDLING, 8., 1998, Vancouver. **Anais**... Vancouver: British Columbia, 1998.

PARENT, C.; SPACCAPIETRA, S.; ZIMÁNYI, E. **Conceptual modeling for traditional and spatio-temporal applications**: the mads approach. [S.l.]: Springer Science & Business Media, 2006a.

PARENT, C.; SPACCAPIETRA, S. ZIMÁNYI, E. The MurMur project: modeling and querying multi-representation spatio-temporal databases. **Information Systems**, [S.l.], v. 31, n. 8, p. 733–769, 2006b.

PGROUTING Manual, 2013. Disponível em: <http://docs.pgrouting.org/2.0/en/pgRoutingDocumentation.pdf>

RIBEIRO, A. A. D. A. et al. Extending OCL to specify and validate integrity constraints in UML geoframe conceptual data model. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, 15., 2013, France. **Proceedings**... [S.l.: s.n.], 2013. p. 286–293.

RIGAUX, P.; SCHOLL, M.; VOISARD, A. **Spatial databases**: with application to GIS. [S.l.]: Morgan Kaufmann, 2001.

RUMBAUGH, J. et al. **Object-oriented modeling and design**. [S.l.]: Prentice-hall, 1991. v. 199.

SAMPAIO, G. B. **GeoProfile**: um perfil uml para modelagem conceitual de bancos de dados geográficos. Viçosa: Universidade Federal de Viçosa, 2009.

SPACCAPIETRA, S.; PARENT, C.; ZIMÁNYI, E. Spatio-temporal and multirepresentation modeling: a contribution to active conceptual modeling. In: ACTIVE conceptual modeling of learning. [S.l.]: Springer, 2007. p. 194–205.

STEMPLIUC, S. M. et al. Extending the UML-geoframe data model for conceptual modeling of network applications. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, 11., Milan. **[Trabalhos apresentados]**... [S.l.: s.n.], 2009. p. 164-170.

TRYFONA, N. et al. **A methodology and a tool for spatiotemporal database design**. [S.l.: s.n.], 1999.

TRYFONA, N.; JENSEN, C. S. Conceptual data modeling for spatiotemporal applications. **Geoinformática**, [S.l.], v. 3, n. 3, p. 245–268, 1999.

Apêndice A

Este apêndice contém os comandos SQL referente ao exemplo da SEER visto na Figura 35 (cf. Seção 5.3).

Quadro 12 – Exemplo de Comandos SQL para Relacionamento Ternário

```
--TABELA UNIDADESAUDE
CREATE TABLE UNIDADESAUDE(
ID_UNISAUDE INTEGER,
PONTOATENDIMENTO GEOMETRY(POINT, 4326),
REGIAOATENDIMENTO GEOMETRY(POLYGON, 4326),
CONSTRAINT PK_UNISAUDE PRIMARY KEY (ID_UNISAUDE)
)
--TABELA MEDICO
CREATE TABLE MEDICO(
ID_MEDICO INTEGER,
NOME CHARACTER VARYING(20),
LOTACAO GEOMETRY(POINT,4326),
CONSTRAINT PK_MEDICO PRIMARY KEY (ID_MEDICO)
)

--TABELA PACIENTE
CREATE TABLE PACIENTE(
ID_PACIENTE INTEGER,
NOME CHARACTER VARYING(20),
RESIDENCIA GEOMETRY(POINT, 4326),
CONSTRAINT PK_PACIENTE PRIMARY KEY (ID_PACIENTE)
)

--TABELA R2
CREATE TABLE R2 (
FK_UNISAUDE INTEGER, FK_MEDICO INTEGER,
FK_PACIENTE INTEGER,
DATA TIMESTAMP,
CONSTRAINT PK_R2 PRIMARY KEY (FK_UNISAUDE, FK_MEDICO, FK_PACIENTE,
DATA),
CONSTRAINT UNISAUDE_R2 FOREIGN KEY (FK_UNISAUDE) REFERENCES UNIDADESAUDE
(ID_UNISAUDE),
CONSTRAINT MEDICO_R2 FOREIGN KEY (FK_MEDICO) REFERENCES MEDICO
(ID_MEDICO),
CONSTRAINT PACIENTE_R2 FOREIGN KEY (FK_PACIENTE) REFERENCES PACIENTE
(ID_PACIENTE)
)
```

```
CREATE FUNCTION WITHIN_RESIDENCIA_REGIAOATENDIMENTO()
RETURNS TRIGGER AS
$BODY$
    DECLARE WITHIN BOOLEAN;
    BEGIN
        SELECT ST_WITHIN(RESIDENCIA, REGIAOATENDIMENTO) INTO WITHIN FROM
        PACIENTE,
        UNIDADESAUDE WHERE ID_PACIENTE = NEW.FK_PACIENTE AND ID_UNISAUDE
        = NEW.FK_UNISAUDE;
        IF(WITHIN = FALSE) THEN
        RAISE EXCEPTION 'TOPOLOGIC ERROR!';
        END IF;
RETURN NEW;
    END
$BODY$
LANGUAGE PLPGSQL VOLATILE;

--CRIA A TRIGGER
CREATE TRIGGER TRG_WITHIN_RESIDENCIA_REGIAOATENDIMENTO AFTER INSERT OR
UPDATE ON R2 FOR EACH ROW EXECUTE PROCEDURE
WITHIN_RESIDENCIA_REGIAOATENDIMENTO();

-- EXEMPLO PRÁTICO:
INSERT INTO UNIDADESAUDE VALUES (1, ST_GEOMFROMTEXT('POINT(2.5 2.5)',
4326), ST_GEOMFROMTEXT('POLYGON((0 0, 0 5, 5 5, 5 0, 0 0))', 4326));

INSERT INTO PACIENTE VALUES (1, 'MARIA', ST_GEOMFROMTEXT('POINT(6 2.5)',
4326));

INSERT INTO MEDICO VALUES (1, 'JOAO', ST_GEOMFROMTEXT('POINT(5 5)',
4326));

-- TESTE "ERRO"
INSERT INTO R2 VALUES (1, 1, 1, '2015-08-17');
```

Apêndice B

Este apêndice contém algumas regras em EVL referente a implementações da semântica estática (cf. Seção 4.4).

Quadro 13 – Implementação de parte da Semântica Estática em EVL

```
context SpatialCondition{

constraint EqualsSRID{
        check{
        var msg : String;
                    if((self.source.size() > 0) and (self.target <> null)){
                      for (sources in self.source){
                      if(sources.srid <> self.target.srid){
                      msg := 'Attributes must have SRID equals. ';
                      return false;
                      }
                      }
             }
return true;
}
         message : msg
}

constraint Cardinality{
        check{
        var msg : String;

for(links in RelationshipLink.allInstances()->select(r | r.cardinality = CardinalityType#MANY)) {
if((links.target = self.conditionLink.target) and
(self.conditionLink.target.relationshipLink.size()==2)){
         if(links.source == self.target.attributeLinkTarget.source){
         if( (self.topologicalOperation == TopologicalOperation#IN_POINT_POINT) or
         (self. topologicalOperation == TopologicalOperation#IN_LINESTRING_LINESTRING) or
     (self. topologicalOperation == TopologicalOperation#IN_POINT_LINESTRING) or (self.
     topologicalOperation == TopologicalOperation#IN_LINESTRING_POLYGON) or (self.
     topologicalOperation ==TopologicalOperation#IN_POLYGON_POLYGON) or (self.
     topologicalOperation == TopologicalOperation#IN_POINT_POLYGON) or
     (self. topologicalOperation == TopologicalOperation#IN_LINESTRING_RASTER) or (self.
     topologicalOperation ==TopologicalOperation#IN_RASTER_POLYGON) or (self.
     topologicalOperation == TopologicalOperation#IN_POLYGON_RASTER) or (self.
     topologicalOperation == TopologicalOperation#IN_RASTER_RASTER)){
     msg := 'Cardinality N can not contain cardinality 1';
return false;
}}}}}message : msg
}
```

```
constraint ValidTopologicalOperations{
        check{
        var msg : String;
if((self.source.size() > 0) and (self.target <> null)){
        for (sources in self.source){
        if(self. topologicalOperation == TopologicalOperation#IN_POINT_LINESTRING){
        if( ( (self.target.dataType = ObjectDataType#POINT) or (self.target.dataType =
    ObjectDataType#MULTIPOINT)) and ((sources.datatype = ObjectDataType#LINESTRING) or
    (sources.dataType = ObjectDataType#MULTILINESTRING))){
                    msg := 'Line can\'t be within of Point. ';
                    return false;
                        }
                    }
        else if(self. topologicalOperation == TopologicalOperation#IN_LINESTRING_POLYGON){
                    if(((self.target.dataType = ObjectDataType#LINESTRING) or
    (self.target.dataType = ObjectDataType#MULTILINESTRING)) and ((sources.dataType =
    ObjectDataType#POLYGON) or (sources.dataType = ObjectDataType#MULTIPOLYGON)) ){
                    msg := 'Polygon can\'t be within of Line. ';
                    return false;
                        }
                    }
        else if(self. topologicalOperation == TopologicalOperation#IN_POINT_POLYGON){
                    if(((self.target.dataType = ObjectDataType#POINT) or(self.target.dataType=
    ObjectDataType#MULTIPOINT)) and ((sources.dataType = ObjectDataType#POLYGON) or
    (sources.dataType = ObjectDataType#MULTIPOLYGON)) ){
                    msg := 'Polygon can\'t be within of Point. ';
                    return false;
                        }
        }else if(self. topologicalOperation == TopologicalOperation#IN_POINT_RASTER){
                if(( (self.target.dataType = ObjectDataType#POINT) or(self.target.dataType=
        ObjectDataType#MULTIPOINT)) and (sources.dataType=FieldDataType#GRIDCELLS)){
                    msg := 'Raster can\'t be within of Point. ';
                    return false;
                        }
        }else if(self. topologicalOperation == TopologicalOperation#IN_LINESTRING_RASTER){
                    if(( (self.target.dataType = ObjectDataType#LINESTRING) or
    (self.target.dataType = ObjectDataType#MULTILINESTRING)) and (sources.dataType =
    FieldDataType#GRIDCELLS)){
                    msg := 'Raster can\'t be within of Line. ';
                    return false;
                                }
                            }
                        }

            }
               return true;
         }
         message : msg
} } //end SpatialCondition
```