

**MAPEAMENTO DE ESQUEMAS CONCEITUAIS  
GEOGRÁFICOS PARA ESQUEMAS GML E  
ESQUEMAS FÍSICOS DE BANCOS DE DADOS  
ESPACIAIS**

ANDRÉ CAVALCANTE HORA

MAPEAMENTO DE ESQUEMAS CONCEITUAIS  
GEOGRÁFICOS PARA ESQUEMAS GML E  
ESQUEMAS FÍSICOS DE BANCOS DE DADOS  
ESPACIAIS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: CLODOVEU AUGUSTO DAVIS JÚNIOR

CO-ORIENTADORA: MIRELLA MOURA MORO

Belo Horizonte

Novembro de 2010

© 2010, André Cavalcante Hora.  
Todos os direitos reservados.

H811m Hora, André Cavalcante  
Mapeamento de Esquemas Conceituais Geográficos  
para Esquemas GML e Esquemas Físicos de Bancos de  
Dados Espaciais / André Cavalcante Hora. — Belo  
Horizonte, 2010  
xii, 69 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de  
Minas Gerais

Orientador: Clodoveu Augusto Davis Júnior

Co-orientadora: Mirella Moura Moro

1. Computação - Teses. 2. Sistemas de informação  
geográfica - Teses. 3. Banco de dados - Teses.  
I. Orientador. II. Coorientadora. I. Título.

CDU 519.6\*72 (043)



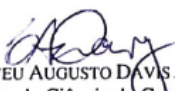
UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

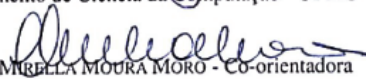
### FOLHA DE APROVAÇÃO

Mapeamento de esquemas conceituais geográficos para esquemas GML e  
esquemas físicos de bancos de dados espaciais

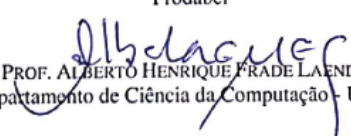
**ANDRÉ CAVALCANTE HORA**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

  
PROF. CLODOVEU AUGUSTO DAVIS JÚNIOR - Orientador  
Departamento de Ciência da Computação - UFMG

  
PROFA. MIRELLA MOURA MORO - Co-orientadora  
Departamento de Ciência da Computação - UFMG

  
DRA. KARLA ALBUQUERQUE DE VASCONCELOS BORGES  
Prodabel

  
PROF. ALBERTO HENRIQUE FRAIDE LAENDER  
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 26 de novembro de 2010.

# Agradecimentos

Agradeço ao Professor Clodoveu e à Professora Mirella pela oportunidade e disponibilidade, pelas orientações inteligentes e por contribuírem com seus ensinamentos no desenvolvimento desse trabalho.

Aos meus queridos pais, Dito e Tereza, e irmãos, Vitor e Vanessa, pela eterna confiança. À Livia pelo carinho, paciência e compreensão. Aos familiares de Maceió pelo incentivo; aos tios e primos de BH pelo acolhimento. A todos os colegas e amigos de BH e Campina Grande pelo apoio.

Às funcionárias da pós-graduação, sempre tão atenciosas e eficientes. Às agências financiadoras, CNPq e FAPEMIG, pela concessão da bolsa de mestrado e suporte.

Muito obrigado a todos!

# Resumo

A modelagem conceitual geográfica, assim como a modelagem conceitual tradicional, é uma atividade de expressiva importância para o projeto de aplicações geográficas. Os modelos conceituais geográficos provêm primitivas para representar a geometria e a topologia dos dados geográficos, que geralmente são armazenados em documentos GML ou em bancos de dados geográficos. O GML também é muito utilizado hoje em dia para a troca de informação entre aplicações geográficas ou na Web. Os bancos de dados geográficos oferecem recursos para a manipulação de dados geográficos, incluindo funções geométricas e topológicas, porém não facilitam a implementação de restrições de integridade espaciais.

Esta dissertação define o mapeamento de esquemas conceituais geográficos OMT-G em esquemas GML e esquemas físicos de bancos de dados espaciais. São propostas regras e algoritmos de mapeamento que preservam a estrutura e a semântica do esquema conceitual, tais como classes e relacionamentos espaciais e não-espaciais. Os esquemas GML são gerados através de uma estratégia para reduzir o uso de referências e aumentar o aninhamento dos elementos. A redução de referências é importante para diminuir o tempo de validação de documentos GML e o aninhamento dos elementos é importante para aumentar a eficiência de consultas a documentos GML. É realizada uma comparação do trabalho proposto com outros relacionados, avaliando as técnicas de mapeamento e os tempos de processamento de consultas em documentos GML. Os resultados mostram que o mapeamento proposto gera resultados eficientes com relação ao tempo de processamento das consultas. Já os esquemas físicos são gerados considerando, além das primitivas que compõem os diagramas de classe do OMT-G, restrições de integridade espaciais que podem ser deduzidas a partir da análise dos diagramas. Isso resulta na criação de tabelas com geometria, índices espaciais, funções para verificação da consistência dos relacionamentos e classes espaciais, entre outros.

**Palavras-chave:** Modelagem de dados geográficos, Esquemas XML, Esquemas GML, Bancos de Dados Espaciais.

# Abstract

Geographic conceptual modeling, along with traditional conceptual modeling, is fundamentally important in the design of geographic applications. Geographic conceptual models provide primitives to represent the geometry and the topology of geographic data, which are generally stored in GML documents or spatial databases. GML is also widely used nowadays in data exchange among geographic applications or through the Web. Geographic databases provide support for spatial data handling, including geometric and topologic functions, but do not facilitate the implementation of spatial integrity constraints.

This dissertation defines the mapping of OMT-G geographic conceptual schemas into GML schemas and physical geographic database schemas. For that purpose, rules and mapping algorithms are proposed in order to preserve the structures and semantics of conceptual schema, such as spatial and non-spatial classes and relationships. GML schemas are generated using a strategy that is able to reduce the use of references and increase the nesting of elements. Reducing references is important because it leads to faster validation of GML documents, while nesting improves the efficiency of queries over GML documents. A comparison between the proposed method and related work is presented, assessing the mapping techniques and query processing times over queries on GML documents. Results show that our mapping method was able to generate time-efficient results. Physical schemas are generated considering both the OMT-G class diagram primitives and spatial integrity constraints that can be deduced from analyzing the conceptual diagrams. The resulting schemas include tables with geometry columns, spatial indexes, spatial-consistency-checking functions, and other elements.

**Keywords:** Geographic data modeling, XML schemas, GML schemas, Spatial databases.

# Lista de Figuras

2.1	Primitivas do OMT-G . . . . .	6
2.2	Exemplo de elemento (esquerda) e elemento aninhado (direita) . . . . .	8
2.3	Esquemas XML que descrevem os elementos da Figura 2.2 . . . . .	10
2.4	Elemento com representação geográfica (esquerda) e seu esquema GML (direita) . . . . .	10
2.5	Esquema SQL com a criação de tabela e índice espacial . . . . .	11
3.1	Esquema GML com os elementos <i>root</i> e <i>spatialDomain</i> . . . . .	17
3.2	Esquema GML representando uma classe espacial do tipo Polígono . . . . .	18
3.3	Esquema OMT-G Bairro-Quadra . . . . .	20
3.4	Esquema GML <i>flat</i> que representa o esquema da Figura 3.3 utilizando duas restrições <i>key</i> e uma <i>keyref</i> . . . . .	21
3.5	Esquema GML <i>aninhado</i> que representa o esquema da Figura 3.3 utilizando duas restrições <i>key</i> (descrito na segunda linha e terceira coluna da Tabela 3.1) . . . . .	22
3.6	Esquemas GML representando um Nó e um Arco . . . . .	23
3.7	Exemplo de um esquema OMT-G . . . . .	25
3.8	Exemplo de um esquema OMT-G sem elementos <i>epn1</i> . . . . .	26
3.9	Grafo direcionado <i>G</i> e seu grafo de componentes fortemente conectados com os elementos <i>epn2</i> : 1, 2, 3, 9 e 10 . . . . .	26
3.10	Média da quantidade de raízes, <i>keyref</i> e profundidade dos esquemas dos tipos (1) e (2) . . . . .	29
3.11	Estrutura hierárquica do esquema gerado pela abordagem proposta ( <i>O2G</i> ) com seus principais elementos . . . . .	33
3.12	Estrutura hierárquica do esquema gerado por <i>Relacionado-2</i> com seus principais elementos . . . . .	34
3.13	Estrutura hierárquica do esquema gerado por <i>Relacionado-1</i> com seus principais elementos . . . . .	34



3.14	Estrutura hierárquica do esquema gerado por <i>flat</i> com seus principais elementos . . . . .	35
3.15	Consultas Q1 (a,b,c) em XQuery e Q6 (d,e,f) em GQL para cada esquema	37
4.1	Exemplo de um esquema OMT-G . . . . .	40
4.2	Visão geral do mapeamento de esquemas OMT-G para esquemas físicos . .	43
4.3	Exemplo de comandos DDL para uma classe espacial . . . . .	45
5.1	Esquema OMT-G para uma aplicação geográfica urbana . . . . .	52
5.2	Esquema GML iniciando o mapeamento por <i>Logradouro</i> . . . . .	55
5.3	Esquema GML iniciando o mapeamento por <i>Região</i> . . . . .	56
5.4	Consultas/resultados SQL e XQuery: recuperar os bairros contidos em uma determinada região . . . . .	57
5.5	Consultas/resultados SQL e XQuery: recuperar os trechos de um determinado logradouro . . . . .	57
5.6	Consultas/resultados SQL e XQuery: recuperar os bairros que são cruzados por um determinado trecho . . . . .	58
5.7	Consultas/resultados SQL e XQuery: recuperar os cruzamentos contidos em uma determinada região . . . . .	58
6.1	Etapas para a geração de documentos GML . . . . .	62
A.1	Esquema XML para representação de esquemas OMT-G . . . . .	69

# Lista de Tabelas

2.1	Principais características dos trabalhos . . . . .	15
3.1	Mapeamento dos relacionamentos convencionais, topológicos e agregações em estruturas GML . . . . .	19
3.2	Documentos permitidos para cada tipo de generalização dado um esquema OMT-G com uma superclasse $A$ e duas subclasses $B$ e $C$ . . . . .	23
3.3	Resumo do mapeamento entre as primitivas OMT-G e GML . . . . .	24
3.4	Quantidade de raízes, keyref e profundidade em cada esquema gerados através de (1) e (2) . . . . .	30
3.5	Descrição das consultas não-espaciais (Q1-Q5) e espaciais (Q6-Q10) . . . .	36
3.6	Tempo de processamento das consultas em segundos . . . . .	38
4.1	Tipos geométricos: OMT-G e OGC . . . . .	41
4.2	Mapeamento dos construtores do modelo OMT-G . . . . .	43
4.3	<i>Scripts</i> e seus dados . . . . .	44

# Sumário

<b>Agradecimentos</b>	<b>v</b>
<b>Resumo</b>	<b>vi</b>
<b>Abstract</b>	<b>vii</b>
<b>Lista de Figuras</b>	<b>viii</b>
<b>Lista de Tabelas</b>	<b>x</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Contexto . . . . .	1
1.2 Motivação . . . . .	2
1.3 Objetivo . . . . .	3
1.4 Organização da Dissertação . . . . .	4
<b>2 Conceitos e Trabalhos Relacionados</b>	<b>5</b>
2.1 O Modelo OMT-G . . . . .	5
2.1.1 Visão Geral . . . . .	5
2.1.2 Restrições de Integridade Espaciais . . . . .	6
2.2 XML e GML . . . . .	8
2.3 Sistemas de Gerência de Bancos de Dados Espaciais . . . . .	10
2.4 Trabalhos Relacionados . . . . .	12
<b>3 Mapeamento para Esquemas GML</b>	<b>16</b>
3.1 Regras de Mapeamento . . . . .	16
3.1.1 Mapeamento do Esquema OMT-G e do Domínio Espacial . . .	16
3.1.2 Mapeamento de Classes Convencionais e Espaciais . . . . .	17
3.1.3 Mapeamento de Relacionamentos Convencionais e Espaciais . .	18
3.1.4 Mapeamento de Generalizações e Generalizações Conceituais . .	20

3.1.5	Mapeamento de Atributos e Restrições . . . . .	23
3.2	Algoritmo de Mapeamento . . . . .	24
3.2.1	Descrição . . . . .	24
3.2.2	Comparação de Mapeamentos . . . . .	28
3.3	Comparação com Outras Abordagens . . . . .	30
3.3.1	Geração de Esquemas GML . . . . .	30
3.3.2	Consultas nos Documentos GML . . . . .	35
3.3.3	Resultados . . . . .	36
<b>4</b>	<b>Mapeamento para Esquemas Físicos</b>	<b>39</b>
4.1	Mapeamento para Esquemas Lógicos . . . . .	40
4.2	Mapeamento para Esquemas Físicos . . . . .	42
4.2.1	Componentes Físicos . . . . .	43
4.2.2	Criação das Tabelas e Índices Espaciais . . . . .	44
4.2.3	Implementação das Restrições de Integridade Espaciais . . . . .	45
<b>5</b>	<b>Estudo de Caso</b>	<b>51</b>
5.1	Esquema OMT-G . . . . .	51
5.2	Esquemas GML e Físicos . . . . .	51
5.3	Consultas . . . . .	53
<b>6</b>	<b>Conclusões e Trabalhos Futuros</b>	<b>59</b>
	<b>Referências Bibliográficas</b>	<b>63</b>
	<b>Apêndice A Ferramentas e Representação de Esquemas OMT-G</b>	<b>68</b>

# Capítulo 1

## Introdução

### 1.1 Contexto

A modelagem conceitual geográfica, assim como a modelagem para bancos de dados tradicionais, é uma atividade de expressiva importância. Ela permite a independência do esquema físico que será utilizado e a documentação do projeto, assim como a reutilização do esquema (ou de parte dele), uma vez que a realidade geográfica modelada se repete para diferentes aplicações. Entretanto, modelos conceituais para aplicações geográficas, como o OMT-G [4], têm necessidades adicionais quando comparados com a modelagem para aplicações convencionais, tanto com relação à abstração de conceitos e classes, quanto no que se refere aos tipos de classes representáveis e seus relacionamentos [6]. O modelo OMT-G utiliza as primitivas definidas para o diagrama de classes UML (*Unified Modeling Language*) e introduz características geográficas com o objetivo de aumentar a capacidade de representação semântica daquele modelo. Deste modo, o OMT-G provê primitivas para modelar a geometria e a topologia dos dados geográficos, utilizando classes e relacionamentos espaciais [4]. O OMT-G é atualmente utilizado por diversas organizações governamentais, industriais e acadêmicas no Brasil.

Esquemas conceituais de modo geral podem ser mapeados para esquemas de bancos de dados [11], bem como para esquemas XML (*eXtended Markup Language*) [1, 12, 13, 17, 21, 28, 38, 40]. A linguagem XML vem se difundindo como um padrão para representação, troca e armazenamento de dados. Razões para isso incluem sua estrutura auto-descritiva e seu formato textual e não-proprietário, que facilita a criação de documentos por pessoas e *softwares*. A existência de diversas linguagens para definição e manipulação de documentos XML, tais como XPath [45],

XQuery [47], XML Schema [46], DTD (*Document Type Definition*) [44] e Relax NG <sup>1</sup>, torna o seu uso ainda mais atraente para o gerenciamento de dados.

Documentos XML não estão restritos à manipulação de dados convencionais. Dados geográficos também podem ser codificados e manipulados em XML. Nesse caso, utiliza-se a linguagem GML (*Geography Markup Language*) [8, 34], que é um padrão do OGC (*Open Geospatial Consortium* <sup>2</sup>), para armazenamento e troca de informação geográfica baseado na XML. Como a GML é fundamentada na XML, as abordagens e tecnologias para o armazenamento e consultas em documentos XML podem ser aplicadas aos documentos GML [48]. Nesse contexto, analogamente ao padrão XML Schema, a linguagem GML provê o GML Schema, que é utilizado para definição de esquemas GML.

Já os sistemas de gerência de bancos de dados (SGBD) convencionais possuem extensões espaciais para tratar os dados inseridos em um contexto geográfico. Exemplos de sistemas de gerência de bancos de dados espaciais são Oracle Spatial<sup>3</sup>, PostGIS<sup>4</sup> e MySQL Spatial Extensions<sup>5</sup>.

O trabalho desenvolvido nesta dissertação se insere no contexto da transformação de esquemas conceituais geográficos OMT-G em esquemas GML e esquemas físicos de bancos de dados visando sua utilização nas mais variadas aplicações GIS (*Geographic Information System*) e nos sistemas de gerência de bancos de dados espaciais.

## 1.2 Motivação

XML e GML são linguagens muito utilizadas para a troca e armazenamento de dados tanto em aplicações convencionais quanto em aplicações geográficas. Documentos XML também são o padrão para publicação e troca de dados na Web. De modo geral, todas as aplicações que manipulam XML também podem manipular GML. Existem diversas aplicações que lidam diretamente com dados em GML. Por exemplo, informações geográficas publicadas no WebGIS<sup>6</sup> utilizam o formato GML. Os dados armazenados por gerenciadores de bancos de dados como o Oracle Spatial e PostGIS podem ser exportados ou importados para XML ou GML. Documentos GML são também utilizados nas especificações de serviços Web da OGC, tais como *Web Map Service* (WMS<sup>7</sup>)

---

<sup>1</sup>Relax NG: <http://relaxng.org>

<sup>2</sup>OGC: <http://www.opengeospatial.org>

<sup>3</sup>Oracle Spatial: <http://www.oracle.com/technetwork/database/options/spatial>

<sup>4</sup>PostGIS: <http://postgis.refractory.net>

<sup>5</sup>MySQL Spatial Extensions: <http://dev.mysql.com/doc/refman/5.0/en/spatial-extensions.html>

<sup>6</sup>WebGIS: <http://www.webgis.com>

<sup>7</sup>Web Map Service: <http://www.opengeospatial.org/standards/wms>

e *Web Feature Service* (WFS<sup>8</sup>), que são recursos importantes para o estabelecimento de infraestruturas de dados espaciais [24]. Entretanto, dadas a complexidade e as peculiaridades das aplicações geográficas, criar uma estrutura de banco de dados por meio de esquemas GML não é uma atividade simples. Além disso, é mais fácil para os projetistas de aplicações especificar e entender os conceitos e os relacionamentos de um sistema usando modelos conceituais que em termos de XML ou GML [31]. É mais natural modelar utilizando um modelo conceitual geográfico, aproveitando a natureza visual dos diagramas de classes e outras primitivas, antes de tentar codificar diretamente as estruturas de banco de dados em esquemas GML.

Já os sistemas de gerência de bancos de dados espaciais são a opção natural para a representação física de esquemas conceituais geográficos. Uma característica diferenciada da informação geográfica é a existência de restrições de integridade espaciais, visto a natureza topológica e geométrica dos dados. Desta forma, além das funcionalidades e da infraestrutura para a manipulação da informação geográfica, os sistemas de gerência de bancos de dados espaciais oferecem apoio através de suas funções topológicas e geométricas para a validação das restrições de integridade espaciais. Apesar de o modelo OMT-G ser utilizado por desenvolvedores GIS em várias organizações, o mapeamento para esquemas físicos de bancos de dados espaciais ainda não foi completamente definido.

## 1.3 Objetivo

O objetivo deste trabalho é definir o mapeamento de esquemas conceituais geográficos OMT-G para (1) esquemas GML e (2) esquemas físicos de bancos de dados espaciais. Para (1) são propostas regras e um algoritmo de mapeamento [19, 20] para a transformação entre esquemas OMT-G e esquemas GML, estendendo propostas existentes na literatura. Para (2) são propostas regras para o mapeamento de esquemas OMT-G em esquemas físicos de bancos de dados espaciais com o padrão *Simple Features Specification for SQL*<sup>9</sup>. Ambos os esquemas gerados preservam as primitivas do esquema original, tais como classes, relacionamentos, generalizações e atributos. Com este mapeamento, pretende-se prover as bases para a implementação de uma ferramenta de projeto OMT-G, capaz de gerar os esquemas físicos (tabelas, índices, restrições de integridade, *triggers*, etc.) de bancos de dados espaciais e os esquemas GML a partir do esquema conceitual.

---

<sup>8</sup>Web Feature Service: <http://www.opengeospatial.org/standards/wfs>

<sup>9</sup>*Simple Features Specification for SQL*: <http://www.opengeospatial.org/standards/sfs>

A transformação de esquemas conceituais, que apresentam estruturas em forma de grafo, para o esquema XML ou GML, que apresenta estruturas hierárquicas, não é um processo direto devido às diferenças existentes entre as representações dos respectivos modelos [41]. Assim, os esquemas devem ser convertidos sem perdas semânticas e estruturais. O mapeamento proposto de esquemas OMT-G em esquemas GML contém as seguintes propriedades: preservação da informação, ausência de redundância de informação e estrutura aninhada.

Uma metodologia preliminar para a transformação de esquemas OMT-G em esquemas de bancos de dados, juntamente com formalizações das restrições de integridade espaciais OMT-G, é apresentada por Borges et al. [5, 6]. O mapeamento proposto nesta dissertação automatiza a transformação de esquemas OMT-G em esquemas físicos de bancos de dados espaciais, estendendo os trabalhos citados, tratando todas as primitivas do OMT-G e suas restrições de integridade espaciais.

## 1.4 Organização da Dissertação

Esta dissertação está organizada como segue. O Capítulo 2 apresenta os principais conceitos relacionados ao modelo OMT-G, XML e GML e bancos de dados espaciais, juntamente com os trabalhos relacionados. No Capítulo 3 a abordagem proposta para a geração de esquemas GML é descrita, sendo apresentadas as regras de mapeamento e o algoritmo desenvolvido, juntamente com uma comparação com outros trabalhos. O Capítulo 4 descreve a abordagem proposta para a geração de esquemas físicos de banco de dados espaciais, sendo apresentada a transformação para os esquemas lógicos e esquemas físicos. O Capítulo 5 apresenta um estudo de caso demonstrando uma aplicação das abordagens propostas. Finalmente, o Capítulo 6 apresenta as conclusões e os trabalhos futuros.



# Capítulo 2

## Conceitos e Trabalhos Relacionados

### 2.1 O Modelo OMT-G

#### 2.1.1 Visão Geral

O modelo OMT-G utiliza as primitivas do diagrama de classes UML e introduz características geográficas para aumentar sua capacidade de representar a semântica dos dados espaciais. O OMT-G inclui primitivas para modelar a geometria e a topologia dos dados geográficos, suportando assim estruturas como agregações espaciais, relacionamentos em rede e relacionamentos topológicos.

Classes e relacionamentos são as primitivas básicas para a criação de esquemas OMT-G. Classes podem ser *convencionais* ou *espaciais*. Classes convencionais não têm propriedades geográficas e se comportam como classes UML. Classes espaciais incluem representação geográfica, que podem ser de dois tipos: individualizáveis, associados a elementos do mundo real (geo-objetos) ou distribuídos continuamente no espaço (geocampos). Geo-objetos podem ser representados utilizando pontos, linhas, polígonos ou elementos de rede (nós, arcos unidirecionais e arcos bidirecionais). Geocampos representam variáveis como tipo de solo, temperatura e relevo, geralmente vistos como uma superfície, e podem ser representados por isolinhas, tesselação, subdivisão planar, triangulação (TIN) ou amostragem. A tesselação pode corresponder a imagens digitais ou grades regulares.

Relacionamentos também podem ser *convencionais* (equivalente aos relacionamentos UML) ou *espaciais*. Relacionamentos espaciais incluem relacionamentos topológicos (ex: disjunto, contido, sobreposto, etc.), relacionamentos em rede (ex: rede viária) e agregações espaciais (i.e., agregações todo-parte, ex: país-estado). Generalizações e especializações podem ser totais/parciais ou disjuntas/sobrepostas e requerem

que as classes participantes tenham o mesmo tipo de representação. Outra primitiva, chamada generalização conceitual, permite a modelagem de objetos com múltiplas representações geográficas, que podem variar de acordo com a escala ou forma geométrica, e podem ser disjuntas ou sobrepostas. Nesse tipo de relacionamento, a superclasse não tem uma representação específica, já que poderá ser percebida de diferentes formas, conforme especificado nas subclasses, que são representadas por formas geométricas distintas. As Figuras 2.1a e 2.1b apresentam, respectivamente, classes do tipo geo-objeto e geocampo e a Figura 2.1c mostra relacionamentos espaciais e generalizações do OMT-G. Mais informações sobre o OMT-G podem ser encontradas em Borges et al. [4, 5, 6].

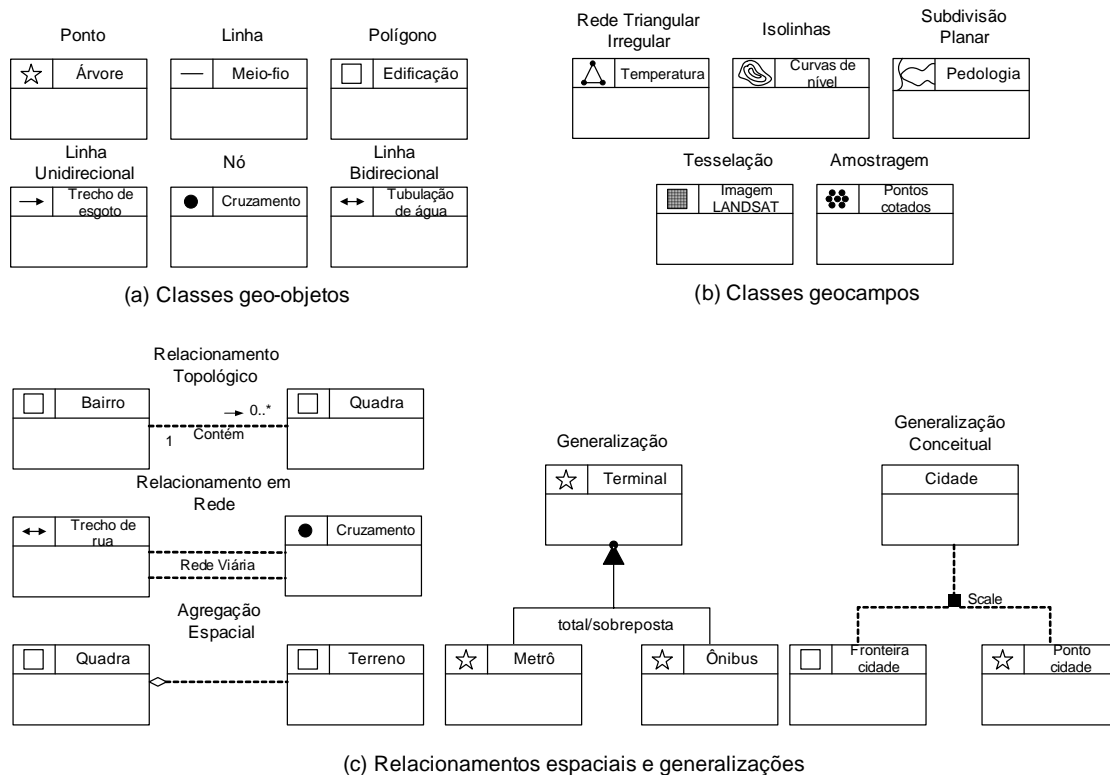


Figura 2.1. Primitivas do OMT-G

### 2.1.2 Restrições de Integridade Espaciais

No modelo OMT-G, algumas restrições de integridade espaciais são definidas implicitamente como parte da semântica das primitivas. Outras restrições são deduzidas pela análise dos esquemas. Restrições de integridade espaciais são definidas para relacionamentos topológicos, relacionamentos em rede, agregações espaciais e classes do tipo geocampo. Restrições de integridade espaciais definidas pelo usuário podem também

ser criadas através da especificação de regras de negócio e de restrições semânticas no esquema. Existem também restrições que se aplicam à representação geométrica das classes de geo-objetos (i.e., restrições relacionadas a consistência dos pontos, linhas, polígonos, etc.). Essas restrições não são detalhadas neste capítulo, visto que os bancos de dados espaciais geralmente definem funções que permitem a verificação direta de sua consistência.

As restrições de integridade do OMT-G [5] são definidas a seguir. As restrições *R1* a *R5* se referem aos geocampos. A restrição *R6* se refere aos relacionamentos em rede, enquanto que a restrição *R7* assegura a semântica das agregações espaciais. Além disso, restrições topológicas (*RT*), associadas aos relacionamentos topológicos tais como *contido*, *disjunto* ou *sobreposto*, correspondem às definições introduzidas por Egenhofer et al. para as matrizes de 4 e 9 interseções [9, 10]. Mais informações sobre as restrições de integridade espaciais do OMT-G podem ser encontradas em Borges et al. [5].

*R1: Restrição de Preenchimento do Plano.* Seja  $F$  um geocampo e seja  $P$  um ponto tal que  $P \in F$ . Então um valor  $V(P) = f(P, F)$ , i.e., o valor de  $F$  em  $P$  pode ser univocamente determinado.

*R2: Isolinhas.* Seja  $F$  um geocampo. Sejam  $\{v_0, v_1, \dots, v_n\}$ ,  $n + 1$  pontos no plano. Sejam  $a_0 = \overline{v_0 v_1}$ ,  $a_1 = \overline{v_1 v_2}$ ,  $\dots$ ,  $a_{n-1} = \overline{v_{n-1} v_n}$ ,  $n$  segmentos, conectando os pontos. Esses segmentos formam uma *isolinha*  $L$  se, e somente se, (1) a interseção de segmentos adjacentes em  $L$  ocorre apenas no ponto extremo compartilhado por eles (i.e.,  $a_i \cap a_{i+1} = v_{i+1}$ ), (2) segmentos não-adjacentes não se interceptam (i.e.,  $a_i \cap a_j = \emptyset$  para todo  $i, j$  tais que  $j \neq i + 1$ ), e (3) o valor de  $F$  em cada ponto  $P$  tal que  $P \in a_i$ ,  $0 \leq i \leq n - 1$ , é constante.

*R3: Tesselação.* Seja  $F$  um geocampo. Seja  $C = \{c_0, c_1, \dots, c_n\}$  um conjunto de células de formato regular e regularmente espaçadas que cobre  $F$ .  $C$  é uma *tesselação* de  $F$  se, e somente se, (1) para cada ponto  $P \in F$ , existir exatamente uma célula  $c_i \in C$ , e (2) para cada célula  $c_i$ , o valor de  $F$  é dado.

*R4: Subdivisão Planar.* Seja  $F$  um geocampo. Seja  $A = \{a_0, a_1, \dots, a_n\}$  um conjunto de polígonos tal que  $a_i \in F$  para todo  $i$  tal que  $0 \leq i \leq n - 1$ .  $A$  forma uma *subdivisão planar* representando  $F$  se, e somente se, para cada ponto  $P \in F$ , existe exatamente um polígono  $a_i \in A$ , para o qual o valor de  $F$  é dado.

*R5: Triangulação (TIN).* Seja  $F$  um geocampo. Seja  $T = \{t_0, t_1, \dots, t_n\}$  um conjunto de triângulos tal que  $t_i \in F$  para todo  $i$  tal que  $0 \leq i \leq n - 1$ .  $T$  forma uma *triangulação* representando  $F$  se, e somente se, para cada ponto  $P \in F$ , existe exatamente um

triângulo  $t_i \in T$ , para o qual o valor de  $F$  é dado.

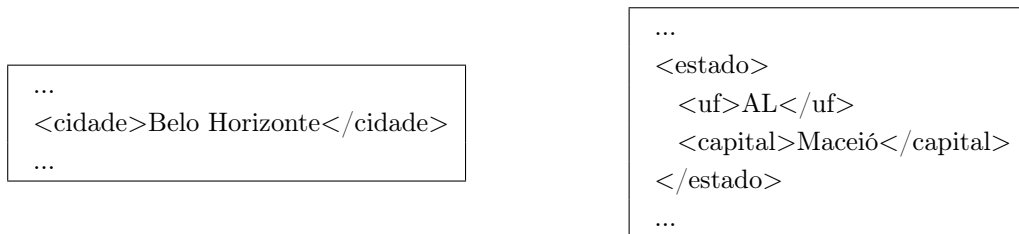
*R6: Rede Arco-Nó.* Seja  $G = \{N, A\}$  uma estrutura de rede, composta de um conjunto de nós  $N = \{n_0, n_1, \dots, n_p\}$  e um conjunto de arcos  $A = \{a_0, a_1, \dots, a_q\}$ . Membros de  $N$  e membros de  $A$  são relacionados de acordo com as seguintes restrições: (1) para cada nó  $n_i \in N$  deve existir pelo menos um arco  $a_k \in A$ ; (2) para cada arco  $a_k \in A$  devem existir exatamente dois nós  $n_i, n_j \in N$ .

*R7: Agregação Espacial.* Seja  $P = \{p_0, p_1, \dots, p_n\}$  um conjunto de geo-objetos. Então  $P$  forma outro objeto  $W$  por *agregação espacial* se, e somente se, (1)  $p_i \cap W = p_i$  para todo  $i$  tal que  $0 \leq i \leq n$ , (2)  $(W \cap \bigcup_{i=0}^n p_i) = W$ , e (3)  $((p_i \text{ toca } p_j) \vee (p_i \text{ disjunto } p_j)) = \text{true}$  para todo  $i, j$  tal que  $i \neq j$ .

Restrições de integridade espaciais podem ser materializadas em esquemas GML (através de estruturas aninhadas) ou fisicamente implementadas em banco de dados espaciais. Por exemplo, cláusulas *check* podem utilizar funções espaciais para garantir a consistência de objetos simples. A implementação de restrições de relacionamentos espaciais podem utilizar *triggers* e outras ferramentas dinâmicas de bancos de dados. Note que algumas aplicações para as quais existam limitações de desempenho podem optar por não implementar as restrições de integridade espaciais diretamente no banco de dados, mas verificar a existência de inconsistências de tempos em tempos.

## 2.2 XML e GML

XML é uma linguagem para a formatação de dados, sendo uma recomendação da W3C (*World Wide Web Consortium*). Os dados são armazenados em documentos XML, que são formados por elementos delimitados por *tags* de abertura e fechamento. A *tag* de abertura pode conter atributos e os elementos podem ser aninhados. A Figura 2.2 mostra trechos de um documento XML com o elemento *cidade* com conteúdo Belo Horizonte, e os elementos *uf* e *capital* aninhados com o elemento *estado*.



**Figura 2.2.** Exemplo de elemento (esquerda) e elemento aninhado (direita)

Documentos XML devem ser bem-formados<sup>1</sup> para seu correto processamento por sistemas, identificação e extração de informações. Documentos XML bem-formados podem ser representados como uma árvore, onde elementos e atributos correspondem aos nós da árvore. XML estabelece um formato rígido para a representação de informações, mas é flexível por possibilitar a criação de *tags* de acordo com as necessidades de cada aplicação e permitir a disposição de um mesmo conjunto de dados de maneiras diferentes. Consultas podem ser realizadas nos documentos XML através de linguagens como XQuery e XPath.

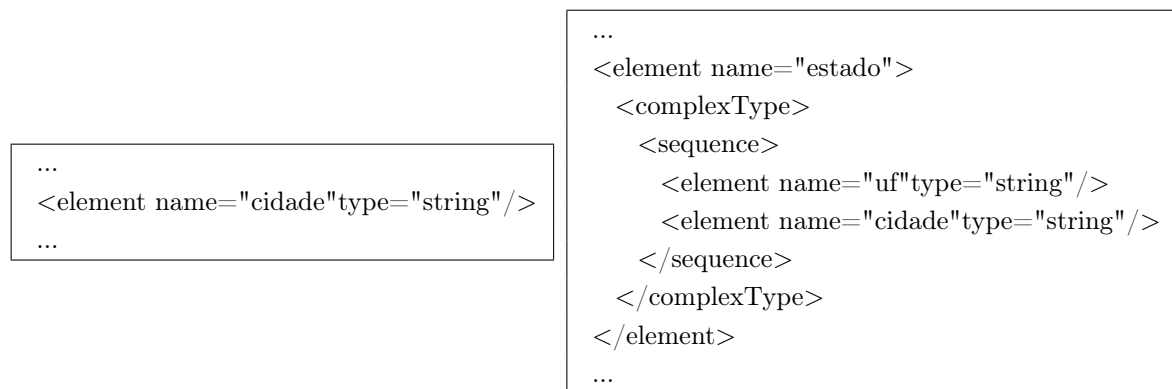
Um documento XML é geralmente descrito por um esquema XML. Seu uso não é obrigatório, contudo, é essencial quando se deseja conhecer o formato dos dados a serem manipulados. Um documento que obedece às restrições de um esquema é dito ser válido em relação a esse esquema. Existem várias linguagens de definição de esquemas XML, tais como DTD, Relax NG e XML Schema. Suas construções permitem a especificação das regras de formação dos documentos XML. Dentre essas linguagens, a mais utilizada e expressiva é XML Schema, que se tornou o padrão para a descrição de estruturas dos documentos XML. Uma característica do XML Schema é a utilização da sintaxe XML para sua definição, o que significa que também é um documento XML.

Diversos elementos, tais como `schema`, `element`, `attribute`, `complexType`, `sequence`, `choice`, `all`, entre outros, podem ser utilizados no XML Schema para a descrição do documento XML. O elemento raiz do XML Schema é `schema`, que contém as demais declarações. As construções `element` e `attribute` definem os elementos e os atributos dos documentos XML e podem estar associados a tipos. O elemento `complexType` define a criação de tipos complexos, ou seja, formado por elementos ou atributos. A base para a formação de um tipo complexo são os construtores `sequence`, `choice` e `all`. O construtor `sequence` estabelece ordem entre os elementos, o `choice` estabelece que apenas um dos elementos conste no documento e o `all` estabelece que qualquer ordem entre os elementos é permitida. A Figura 2.3 mostra trechos dos esquemas XML que descrevem os elementos da Figura 2.2.

GML é uma linguagem baseada em XML, criada para modelar e codificar informação geográfica, bem como para permitir seu transporte entre aplicações [8, 24]. Os mesmos conceitos de esquemas e documentos também se aplicam à GML. GML permite a criação de objetos concretos, como ruas ou construções, ou conceituais, como regiões e quadras. Os objetos são descritos em termos de suas propriedades geométricas como localização e forma. Uma propriedade geométrica é geralmente definida como uma instância de GML *Polygon*, GML *LineString*, GML *Point*, dentre outras, para re-

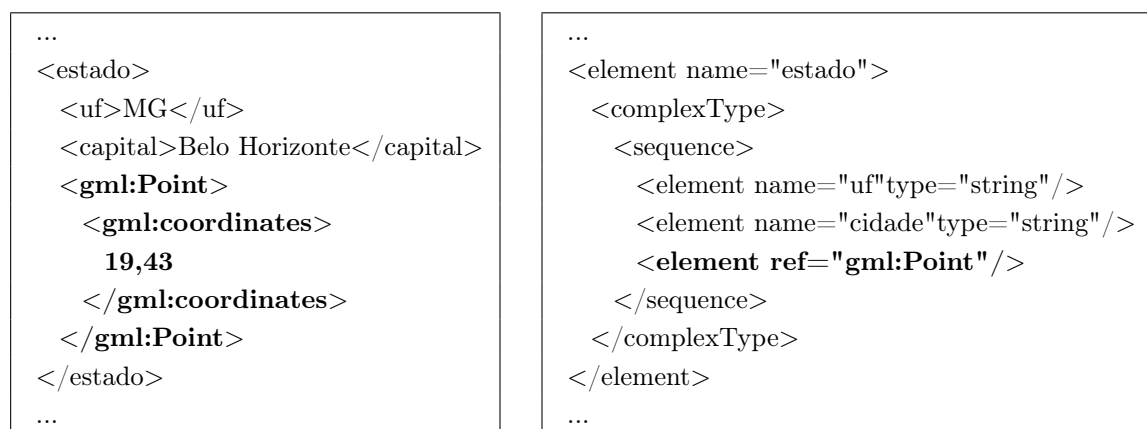
---

<sup>1</sup>Documentos XML bem-formados: <http://www.w3.org/TR/REC-xml>



**Figura 2.3.** Esquemas XML que descrevem os elementos da Figura 2.2

presentar polígonos, linhas e pontos, através de um sistema de coordenadas. Consultas nos documentos GML podem ser realizadas através extensões espaciais da linguagem XQuery, tais como GQuery [7], GQL [16] e GML-QL [42], que permitem a utilização de funções espaciais. A Figura 2.4 mostra o trecho de um documento GML (com uma instância do elemento GML *Point*) e seu esquema (com a descrição do elemento GML *Point*).



**Figura 2.4.** Elemento com representação geográfica (esquerda) e seu esquema GML (direita)

## 2.3 Sistemas de Gerência de Bancos de Dados Espaciais

Um sistema de banco de dados espacial é um sistema de banco de dados que oferece tipos de dados e consultas espaciais, provendo índices espaciais e algoritmos eficientes

para a manipulação da informação geográfica [15]. Em um banco de dados espacial podem existir dados convencionais e dados espaciais. Dados convencionais descrevem as características, e dados espaciais descrevem a localização e a forma geométrica dos objetos espaciais. Objetos espaciais geralmente estão associados a um domínio espacial, i.e., seus limites geográficos. Objetos espaciais são geralmente modelados para representar elementos do mundo real usando formas geométricas simples, tais como pontos, linhas ou polígonos. As principais extensões espaciais de bancos de dados são Oracle Spatial, PostGIS e MySQL Spatial Extensions. O Oracle Spatial, apesar de ser um software comercial, está disponível para fins de pesquisa.

O Oracle Spatial permite a definição de novos tipos de dados através da linguagem de definição de dados DDL (*Data Definition Language*), e a implementação de operações sobre esses novos tipos, através da linguagem PL/SQL, uma extensão da SQL. Ele é baseado nas especificações do OpenGIS e contém um conjunto de funcionalidades e procedimentos que permitem armazenar, acessar, modificar e consultar dados espaciais. É formado pelos seguintes componentes: o esquema MDSYS (que define a forma de armazenamento, a sintaxe e semântica dos tipos espaciais suportados, tais como *Polygon*, *LineString* e *Point*), mecanismos de indexação espacial e um conjunto de operadores e funções para representar consultas, junção espacial e outras operações de análise espacial [39, 23]. A Figura 2.5 mostra um esquema SQL para o Oracle Spatial com a criação da tabela *Estado* (com as colunas *uf* e *capital*, e a coluna espacial *geom*, do tipo SDO\_GEOMETRY, responsável por armazenar os dados espaciais) e a criação do índice espacial para a coluna espacial, indicando que *geom* deve ser um ponto com duas dimensões.

```
CREATE TABLE Estado (  
  uf CHAR(2),  
  capital VARCHAR2(20),  
  geom MDSYS.SDO_GEOMETRY);  
  
CREATE INDEX idx_Estado ON Estado(geom)  
  INDEXTYPE IS MDSYS.SPATIAL_INDEX  
  PARAMETERS ('SDO_INDX_DIMS=2,LAYER_GTYPE=POINT');
```

**Figura 2.5.** Esquema SQL com a criação de tabela e índice espacial

## 2.4 Trabalhos Relacionados

Esta seção aborda trabalhos sobre a geração de esquemas e documentos XML e GML assim como esquemas físicos de bancos de dados.

Existem diferentes técnicas para o mapeamento de esquemas conceituais, tais como ER (entidade-relacionamento), EER (*extended entity relationship* - entidade-relacionamento estendido), entre outros, para esquemas XML. Pigozzo & Quintarelli [38] apresentam um algoritmo para gerar esquemas XML, através da linguagem XML Schema, a partir de esquemas EER. O esquema XML gerado contém todas as primitivas do modelo original: entidades, relacionamentos, atributos, cardinalidades e generalizações. Para evitar redundâncias no esquema XML, são utilizadas chaves primárias e chaves estrangeiras. Algumas entidades do esquema EER são mapeadas como elementos da raiz no esquema XML, sendo assim denominadas *entidades de primeiro nível*, que são escolhidas de acordo com suas cardinalidades nos relacionamentos. O algoritmo também utiliza certas *condições de inclusão*, em que restrições são estabelecidas para a inserção de novos sub-elementos no esquema XML.

Franceschet et al. [13] estendem o algoritmo desenvolvido por Pigozzo & Quintarelli [38], utilizando o modelo espaço-temporal ChronoGeoGraph (CGG) no lugar do EER. São, assim, gerados esquemas XML e GML a partir de esquemas ChronoGeoGraph. Franceschet et al. [12] também propõem um mapeamento de esquemas EER para a linguagem XML Schema com as seguintes propriedades: preservação de informações, ausência de redundância, estrutura gerada altamente conectada e resultado reversível (ou seja, a partir do esquema XML é possível retornar ao esquema EER). Na realização do mapeamento, o esquema EER é primeiramente expresso em uma linguagem denominada XLS. A transformação entre o XLS e o XML Schema ocorre de forma direta e é definida através de uma sequência de passos que detalham o mapeamento de entidades, relacionamentos, entidades fracas e especializações. Entretanto, o trabalho não deixa claro quais entidades do esquema EER são selecionadas para serem entidades de primeiro nível e como os atributos são mapeados, além de não apresentar o algoritmo de mapeamento.

Liu & Li [28] propõem a criação de esquemas XML a partir de esquemas ER. Alguns critérios de qualidade para a modelagem de esquemas XML são discutidos e dentre esses se destacam: preservação da informação, ausência de redundância, estrutura fortemente aninhada e resultado reversível. Seguindo os critérios de qualidade apresentados, é proposto um algoritmo para a transformação de esquemas ER em esquemas XML. Esse algoritmo é baseado em algumas regras que mapeiam entidades, relacionamentos, generalizações e atributos para esquemas na linguagem XML Schema.



Um exemplo simples de mapeamento entre os esquemas é apresentado para mostrar o uso do algoritmo proposto. Entretanto, assim como no trabalho de Franceschet et al. [12, 13], não fica claro quais entidades do esquema ER são selecionadas para serem entidades de primeiro nível no esquema XML.

Schroeder & Mello [40] apresentam uma abordagem diferente das anteriores para o mapeamento de esquemas conceituais em esquemas XML. É proposta a geração de esquemas lógicos XML, ou seja, independentes da linguagem de esquema XML, a partir de esquemas ER, levando em consideração a quantidade de elementos existentes no documento XML e a carga de operações que serão realizadas nas consultas aos documentos XML. Assim, mostra-se necessário um conhecimento prévio sobre a quantidade estimada de instâncias de entidades e sobre quais operações serão realizadas com maior frequência. Essa informação é apresentada em conjunto com o esquema ER na etapa de modelagem. Assim, o algoritmo de conversão utiliza as informações pertencentes ao esquema ER, como entidades, relacionamentos e generalizações, juntamente com os dados de carga previamente fornecidos, para a geração do esquema XML lógico. Em Schroeder & Mello [41] é apresentada uma comparação da técnica apresentada com outras abordagens de mapeamento.

Alguns trabalhos utilizam outros modelos conceituais para a geração do esquema XML. Al-Kamha et al. [1] focam na representação de generalização/especialização na linguagem XML Schema, utilizando o modelo Conceptual XML. Locio et al. [29] utilizam o modelo X-Entity, uma extensão do ER, para a modelagem de esquemas XML. Mok & Embley [31] visam gerar estruturas XML a partir da análise de restrições em um modelo conceitual genérico. Algumas ferramentas comerciais providas pela IBM<sup>2</sup> e Sparx Systems<sup>3</sup> possuem softwares ou bibliotecas para converter determinados esquemas conceituais em esquemas XML. Diferentemente dos trabalhos anteriores, Kleiner & Lipeck [21] apresentam um algoritmo para a geração de esquemas DTD a partir de esquemas ER e Amano et al. [2] apresentam uma linguagem para o mapeamento entre esquemas XML. Uma comparação entre diversas abordagens de mapeamento de esquemas conceituais em esquemas XML é apresentada por Necasky [32].

Com relação à geração de documentos XML e GML, Park et al. [37] propõem um sistema que converte a informação geográfica armazenada nos bancos de dados espaciais para documentos GML, visando a interoperabilidade entre diferentes aplicações GIS. Entretanto, essa informação geográfica não é interpretada, sendo simplesmente exportada para documentos. Uma abordagem para a geração de documentos XML sintéticos é minuciosamente detalhada por Barbosa et al. [3].

---

<sup>2</sup>Model Transformation Framework e Generating XSD Schemas from UML Models.

<sup>3</sup>Sparx Systems UML to XML Schema Transformation.

Tendo em vista a geração de esquemas XML a partir de esquemas conceituais, a maioria dos trabalhos citados utiliza a linguagem XML Schema, devido à sua maior expressividade em relação às demais, característica que a tornou o padrão para descrição de estruturas de documentos XML. Foca-se na preservação da semântica e da estrutura existente no esquema conceitual para que não haja perda de informação durante a transformação. De modo geral, poucos detalhes sobre os algoritmos de mapeamento são apresentados. Comparações com outras abordagens, exemplos de aplicação, e implementações dos mapeamentos propostos também são escassos. Apenas um trabalho [13] está inserido no contexto geográfico, tratando da transição entre esquemas espaço-temporais em esquemas XML e GML.

Existem diversos modelos conceituais propostos para modelar aplicações geográficas [4, 17, 18, 22, 26, 33, 36, 43]. Gubiani [17] apresenta uma revisão sobre alguns dos principais modelos conceituais geográficos existentes. Tais modelos são usados na etapa de modelagem conceitual geográfica, mas, em geral, poucos detalhes são fornecidos sobre as etapas de modelagem lógica e física. Alguns modelos apresentam descrições de ferramentas associadas, que auxiliam o usuário na criação de esquemas conceituais e automatiza a geração de esquemas físicos para bancos de dados espaciais [25, 27, 30, 49]. Gubiani [17] apresenta o modelo espaço-temporal ChronoGeoGraph e descreve a etapa de modelagem lógica. Uma ferramenta<sup>4</sup> está sendo desenvolvida para a modelagem conceitual gráfica do ChronoGeoGraph e a geração de esquemas físicos para o Oracle Spatial, mas essa ainda apresenta alguns erros de mapeamento.

Este trabalho apresenta de forma detalhada os mapeamentos propostos. Na transformação para esquemas GML, são apresentados um conjunto de regras e um algoritmo de mapeamento assim como uma comparação com outras abordagens. Na transformação para esquemas de bancos de dados espaciais, são apresentadas as etapas lógica e física. Um estudo de caso exemplifica a utilização dos mapeamentos. Para ambos os mapeamentos, ferramentas foram desenvolvidas visando automatizar a transformação entre os esquemas. A Tabela 2.1 mostra as principais características de cada trabalho.

---

<sup>4</sup>ChronoGeoGraph Tool: <http://dbms.dimi.uniud.it/cgg>

**Tabela 2.1.** Principais características dos trabalhos

<b>Abordagem</b>	<b>Esquema fonte</b>	<b>Esquema alvo</b>	<b>Elementos de primeiro nível</b>	<b>Algoritmo Regras</b>	<b>Ferramenta</b>
Pigozzo & Quintarelli [38]	EER	XML Schema	Sim	Sim	Não
Franceschet et al. [12] [13]	EER e CGG	XML/GML Schema	Não	Parcial	Sim
Liu & Li [28]	ER	XML Schema	Não	Sim	Não
Schroeder & Mello [40]	ER	XML Schema	Sim	Sim	Não
Proposto	OMT-G	GML Schema	Sim	Sim	Sim

## Capítulo 3

# Mapeamento para Esquemas GML

Este capítulo define o mapeamento de esquemas conceituais OMT-G para esquemas GML [19]. O mapeamento ocorre sem perdas semânticas e estruturais ao passo que reduz o uso de restrições e aumenta o aninhamento dos elementos no esquema GML. Na Seção 3.1 são apresentadas as regras de mapeamento e a Seção 3.2 apresenta o algoritmo que realiza a transformação entre os esquemas. Na Seção 3.3 é apresentada uma comparação do mapeamento proposto com trabalhos relacionados, juntamente com uma análise dos esquemas GML gerados e dos tempo de processamento de consultas.

### 3.1 Regras de Mapeamento

A seguir são apresentadas as regras de mapeamento entre as primitivas do modelo OMT-G e as primitivas da linguagem GML. São definidas regras para o mapeamento do esquema OMT-G, domínio espacial, classes e relacionamentos convencionais e espaciais, generalizações, atributos e restrições. O algoritmo apresentado na Seção 3.2 utiliza essas regras para a geração dos esquemas GML. Para simplificar a descrição das regras, elementos como `complexType` e `sequence` são omitidos sempre que possível. As regras consideram um esquema OMT-G  $O$ , formado por classes  $c$  e relacionamentos  $r$  e um esquema GML  $G$  formado por elementos.

#### 3.1.1 Mapeamento do Esquema OMT-G e do Domínio Espacial

*Regra 1: Esquema OMT-G.* Criar um elemento *root* como sub-elemento do elemento XML `schema` no esquema GML  $G$  para representar o esquema OMT-G  $O$ . As classes

$c_i \in O$  e os relacionamentos  $r_i \in O$  mapeados para  $G$  são criados como sub-elementos  $ec_i$  e  $er_i$  de *root*.

*Regra 2: Domínio Espacial.* Criar um elemento opcional *spatialDomain* como sub-elemento de *root* com as propriedades geométricas<sup>1</sup> do GML Schema *boundedBy* (descreve os limites aproximados do domínio espacial, como um retângulo envolvente mínimo) e *extentOf* (descreve os limites reais do domínio espacial).

O domínio espacial representa os limites espaciais dos elementos  $ec_i \in G$  no documento, i.e., todos os elementos (que representam classes espaciais) de um documento GML baseado no esquema GML devem estar topologicamente contidos nesse limite. Esquemas OMT-G não incluem informações sobre o domínio espacial de forma explícita, assim estas podem ser fornecidas pelo usuário no momento da criação dos documentos GML. A Figura 3.1 ilustra o esquema GML com os elementos *root* e *spatialDomain*.

```
<schema>
  <element name="root">
    <complexType>
      <sequence>
        <element name="spatialDomain" minOccurs="0" maxOccurs="1"/>
        <complexType>
          <sequence>
            <element ref="gml:boundedBy" minOccurs="0" maxOccurs="1"/>
            <element ref="gml:extentOf" minOccurs="0" maxOccurs="1"/>
          </sequence>
        </complexType>
      </element>
      <!-- classes, relacionamentos e generalizações -->
    </sequence>
  </complexType>
</element>
</schema>
```

**Figura 3.1.** Esquema GML com os elementos *root* e *spatialDomain*

### 3.1.2 Mapeamento de Classes Convencionais e Espaciais

*Regra 3: Classes Convencionais e Espaciais.* Para cada classe  $c_i \in O$  criar um elemento  $ec_i$ . Caso  $c_i$  seja uma classe espacial, criar um sub-elemento em  $ec_i$  com uma geometria do GML Schema de acordo com o tipo da classe. Para as classes do tipo Ponto, Nó ou Amostragem criar um sub-elemento GML *Point*. Para as classes Linha, Arco Unidirecional ou Arco Bidirecional criar um sub-elemento GML *LineString*. Para

<sup>1</sup>Propriedades geométricas na GML impõe uma determinada interpretação para uma geometria [24]

Polígono ou Subdivisão Planar criar um sub-elemento GML *Polygon*. Para as classes Isolinhas criar um sub-elemento GML *LineString* e/ou GML *Polygon*. Para classes Triangulação criar um sub-elemento GML *Polygon* (triângulos) e GML *Point* (vértices). Para classes Tesselação criar um sub-elemento GML *Grid* (que representa grades regulares).

A Figura 3.2 ilustra o esquema GML para uma classe espacial do tipo Polígono.

```
<element name="ec_i">
  <complexType>
    <sequence>
      <!-- atributos -->
      <element ref="gml:Polygon"/>
      <!-- relacionamentos e generalizações -->
    </sequence>
  </complexType>
</element>
```

**Figura 3.2.** Esquema GML representando uma classe espacial do tipo Polígono

### 3.1.3 Mapeamento de Relacionamentos Convencionais e Espaciais

Nos sistemas de gerência de bancos de dados espaciais, relações topológicas entre objetos são verificadas *on-the-fly* (i.e., no momento da consulta) através de funções topológicas. Assim, os relacionamentos topológicos no esquema conceitual indicam restrições de integridade espaciais, ao invés de especificar a materialização das conexões entre os objetos. Nos esquemas GML, visto que estes não possuem funcionalidades nativas para representar restrições de integridade espaciais, os relacionamentos topológicos e as agregações são mapeadas para relacionamentos convencionais, o que exige a criação de elementos aninhados ou de pares chave primária-chave estrangeira, usando elementos *key* e *keyref*. É importante tentar reduzir o uso da restrição *keyref* e aumentar o aninhamento dos elementos no esquema GML. A redução do uso de restrições no esquema GML é importante para diminuir o tempo de validação de documentos GML. Já o aninhamento dos elementos no esquema GML é importante para aumentar a eficiência de consultas a documentos GML.

O mapeamento dos relacionamentos é realizado de acordo com regras introduzidas por Franceschet et al. [12] e resumidas na Tabela 3.1. Nas bordas da tabela encontram-se as cardinalidades da participação de classes *A* e *B* no relacionamento binário *R*. As notações do corpo da tabela indicam de forma simplificada o mapeamento

para esquemas GML. Por exemplo, a notação  $A(kA, R[\min, \max])$  indica a criação de um elemento representando  $A$  com dois sub-elementos. O primeiro sub-elemento<sup>2</sup>  $kA$  representa sua chave e o segundo sub-elemento  $R[\min, \max]$  representa o relacionamento  $R$  que  $A$  participa. A indentação indica o aninhamento entre os elementos, e *key* e *keyref* representam chaves primárias e estrangeiras.

*Regra 4: Relacionamentos convencionais, topológicos e agregações.* Para cada relacionamento convencional, topológico e agregação<sup>3</sup>  $r_i \in O$  entre uma classe  $c_i \in O$  e uma classe  $c_j \in O$ , criar um sub-elemento  $er_i$  no elemento representando a classe  $c_i$  ou  $c_j$  com as restrições *minOccurs* e *maxOccurs* de acordo com as cardinalidades de  $r_i$ , seguindo as regras da Tabela 3.1.

**Tabela 3.1.** Mapeamento dos relacionamentos convencionais, topológicos e agregações em estruturas GML

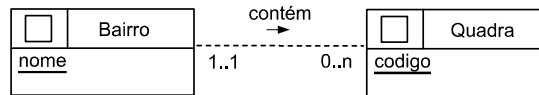
B/A	(0,1)	(0,n)	(1,1)	(1,n)
(0,1)	A(kA, R[0,1]) R(kB) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>key</i> (R.kB) <i>keyref</i> (R.kB→B.kB)	A(kA, R[0,1]) R(kB) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>keyref</i> (R.kB→B.kB)	A(kA, R[0,1]) R(B) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB)	A(kA) B(kB, R[1,n]) R(kA) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>key</i> (R.kA) <i>keyref</i> (R.kA→A.kA)
(0,n)	A(kA) B(kB, R[0,1]) R(kA) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>keyref</i> (R.kA→A.kA)	A(kA, R[0,n]) R(kB) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>keyref</i> (R.kB→B.kB)	A(kA, R[0,n]) R(B) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB)	A(kA) B(kB, R[1,n]) R(kA) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>keyref</i> (R.kA→A.kA)
(1,1)	B(kB, R[0,1]) R(A) A(kA) <i>key</i> (B.kB), <i>key</i> (A.kA)	B(kB, R[0,n]) R(A) A(kA) <i>key</i> (B.kB), <i>key</i> (A.kA)	A(kA, R[1,1]) R(B) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB)	B(kB, R[1,n]) R(A) A(kA) <i>key</i> (B.kB), <i>key</i> (A.kA)
(1,n)	A(kA, R[1,n]) R(kB) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>key</i> (R.kB) <i>keyref</i> (R.kB→B.kB)	A(kA, R[1,n]) R(kB) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>keyref</i> (R.kB→B.kB)	A(kA, R[1,n]) R(B) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB)	A(kA, R[1,n]) R(kB) B(kB) <i>key</i> (A.kA), <i>key</i> (B.kB) <i>keyref</i> (R.kB→B.kB)

Como exemplo, considere o mapeamento do esquema OMT-G da Figura 3.3, onde cada bairro contém zero ou mais quadras, e cada quadra está contida em exatamente um bairro. Existem duas possibilidades de mapeamento para esquemas GML (ambas preservando a semântica e a estrutura do esquema OMT-G): a *flat* (i.e., não considerado o aninhamento) e a *aninhada*. As Figuras 3.4 e 3.5 mostram esquemas GML para as abordagens *flat* e *aninhada*, respectivamente, representando o esquema OMT-G da Figura 3.3. Observe que o esquema *flat*, além de não utilizar o aninhamento, define duas

<sup>2</sup>Outros sub-elementos são omitidos por simplicidade.

<sup>3</sup>Agregações são tratadas como relacionamentos com cardinalidade (1,1) na classe *todo* e (1,n) na classe *parte*.

chaves primárias (*key*) e uma chave estrangeira (*keyref*) para indicar a correspondência entre quadras e bairros. Já o esquema *aninhado* define apenas duas chaves primárias (*key*). Logo, para esse exemplo, a Tabela 3.1 indica o mapeamento *aninhado* (i.e., opção que reduz o uso das restrições e utiliza o aninhamento) na segunda linha e terceira coluna (i.e., A(1,1) corresponde ao bairro e B(0,n) corresponde a quadra). Assim, no esquema *aninhado*, um elemento *bairro* é criado tendo seus atributos e o relacionamento *contém* como sub-elementos. Já que podem existir zero ou mais quadras contidas em cada bairro, o número mínimo e máximo de ocorrência de *contém* é (0,n). Um elemento *quadra* é criado, com seus atributos, como sub-elemento do elemento *contém*. Em seguida, os elementos *key* são definidos para as chaves primárias de *bairro* e *quadra*. Observe que, nesse caso, nenhum elemento *keyref* é necessário, devido ao uso do aninhamento.



**Figura 3.3.** Esquema OMT-G Bairro-Quadra

*Regra 5: Relacionamentos em Rede.* Para cada relacionamento em rede  $r_i \in O$  entre uma classe do tipo Arco Unidirecional ou Bidirecional  $c_i \in O$  e uma classe do tipo Nó  $c_j \in O$ , criar dois sub-elementos  $n1$  e  $n2$  no elemento  $arcoc_i$  representando a classe  $c_i$  e duas referências (uma para  $n1$  e uma para  $n2$ ) para o elemento  $noc_j$  representando a classe  $c_j$ .

A Figura 3.6 mostra esquemas GML genéricos representando as classes Nó e Arco de um relacionamento em rede.

### 3.1.4 Mapeamento de Generalizações e Generalizações Conceituais

Considere uma superclasse  $c_i \in O$  e suas subclasses  $s_i \in O$  relacionadas através de uma generalização  $r_i$  em um esquema OMT-G. Em um documento GML, a generalização total/disjunta permite que apenas um elemento representando uma das subclasses possa existir. A total/sobreposta permite que um ou mais elementos representando as subclasses possam existir. A parcial/disjunta permite que zero ou um elemento representando as subclasses possam existir. A parcial/sobreposta permite que zero ou mais elementos representando as subclasses possam existir. Assim, elementos de sequência (*sequence*) e escolha (*choice*) são utilizados no esquema GML para o mapeamento das generalizações.



<pre> &lt;element name="bairro"&gt;   &lt;complexType&gt;     &lt;sequence&gt;       &lt;element name="nome" type="xs:string"/&gt;       &lt;element ref="gml:Polygon"/&gt;     &lt;/sequence&gt;   &lt;/complexType&gt; &lt;/element&gt; </pre>
<pre> &lt;element name="quadra"&gt;   &lt;complexType&gt;     &lt;sequence&gt;       &lt;element name="codigo" type="xs:integer"/&gt;       &lt;element ref="gml:Polygon"/&gt;       &lt;element name="estaContido" minOccurs="1" maxOccurs="1"&gt;         &lt;complexType&gt;           &lt;attribute name="ref-bairro"/&gt;         &lt;/complexType&gt;       &lt;/element&gt;     &lt;/sequence&gt;   &lt;/complexType&gt; &lt;/element&gt; </pre>
<pre> &lt;key name="bairro-key"&gt;   &lt;selector xpath="//bairro"/&gt;   &lt;field xpath="nome"/&gt; &lt;/key&gt; </pre>
<pre> &lt;key name="quadra-key"&gt;   &lt;selector xpath="//quadra"/&gt;   &lt;field xpath="codigo"/&gt; &lt;/key&gt; </pre>
<pre> &lt;keyref name="quadra-keyref" refer="bairro-key"&gt;   &lt;selector xpath="//quadra"/&gt;   &lt;field xpath="ref-bairro"/&gt; &lt;/keyref&gt; </pre>

**Figura 3.4.** Esquema GML *flat* que representa o esquema da Figura 3.3 utilizando duas restrições *key* e uma *keyref*

*Regra 6: Generalização Total/Disjunta.* Para cada generalização  $r_i$  do tipo total/disjunta, criar um sub-elemento *choice* no elemento  $super_i$  representando a superclasse  $c_i$ . As subclasses  $s_i$  são criadas como sub-elementos  $sub_i$  de *choice*.

*Regra 7: Generalização Total/Sobreposta.* Para cada generalização  $r_i$  do tipo total/sobreposta, criar um sub-elemento *choice* com as restrições  $minOccurs=1$  e  $maxOccurs=número\ de\ subclasses$  no elemento  $super_i$  representando a superclasse  $c_i$ . As subclasses  $s_i$  são criadas como sub-elementos  $sub_i$  de *choice*.

*Regra 8: Generalização Parcial/Disjunta.* Para cada generalização  $r_i$  do tipo parcial/disjunta, criar um sub-elemento *choice* com as restrições  $minOccurs=0$  e  $maxOccurs=1$  no elemento  $super_i$  representando a superclasse  $c_i$ . As subclasses  $s_i$  são criadas como sub-elementos  $sub_i$  de *choice*.

*Regra 9: Generalização Parcial/Sobreposta.* Para cada generalização  $r_i$  do tipo par-

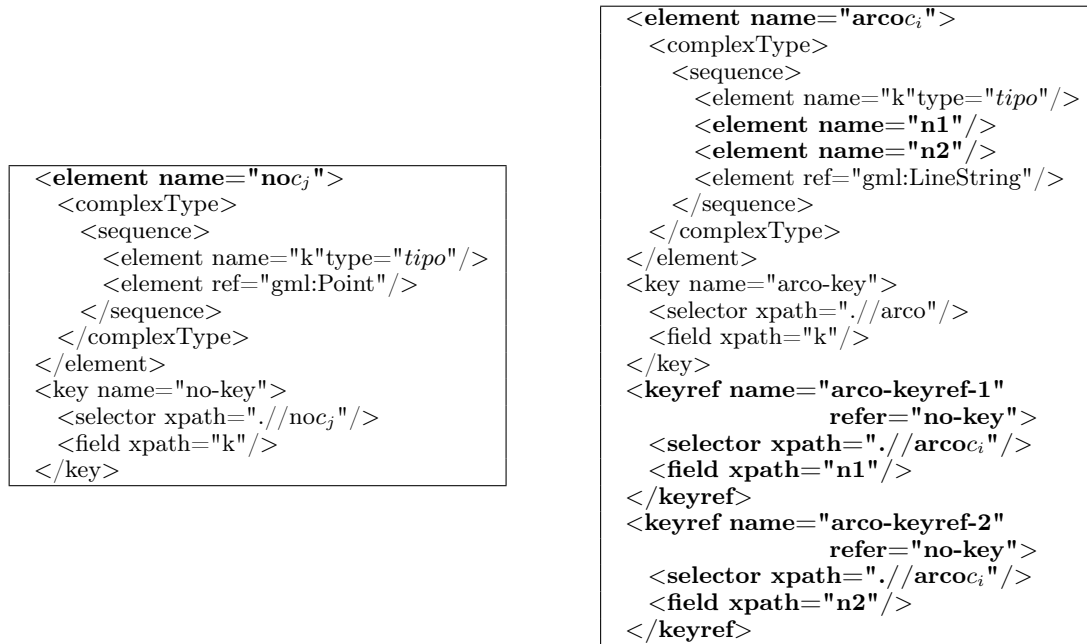
<pre> &lt;element name="bairro"&gt;   &lt;complexType&gt;     &lt;sequence&gt;       &lt;element name="nome" type="xs:string"/&gt;       &lt;element ref="gml:Polygon"/&gt;       &lt;element name="contem" minOccurs="0" maxOccurs="unbounded"&gt;         &lt;complexType&gt;           &lt;sequence&gt;             &lt;element name="quadra"&gt;               &lt;complexType&gt;                 &lt;sequence&gt;                   &lt;element name="codigo" type="xs:integer"/&gt;                   &lt;element ref="gml:Polygon"/&gt;                 &lt;/sequence&gt;               &lt;/complexType&gt;             &lt;/element&gt;           &lt;/sequence&gt;         &lt;/complexType&gt;       &lt;/element&gt;     &lt;/sequence&gt;   &lt;/complexType&gt; &lt;/element&gt; </pre>
<pre> &lt;key name="bairro-key"&gt;   &lt;selector xpath="."/&gt;   &lt;field xpath="nome"/&gt; &lt;/key&gt; </pre>
<pre> &lt;key name="quadra-key"&gt;   &lt;selector xpath="."/&gt;   &lt;field xpath="codigo"/&gt; &lt;/key&gt; </pre>

**Figura 3.5.** Esquema GML *aninhado* que representa o esquema da Figura 3.3 utilizando duas restrições *key* (descrito na segunda linha e terceira coluna da Tabela 3.1)

cial/sobreposta, criar um sub-elemento *sequence* no elemento *super<sub>i</sub>* representando a superclasse *c<sub>i</sub>*. As subclasses *s<sub>i</sub>* são criadas como sub-elementos *sub<sub>i</sub>* com as restrições *minOccurs=0* e *maxOccurs=1* de *sequence*.

*Regra 10. Generalização Conceitual. Disjunta:* O mapeamento é análogo ao da generalização total/disjunta. *Sobreposta:* O mapeamento é análogo ao da generalização total/sobreposta.

Note que o mapeamento dos tipos das classes não são especificados, uma vez que são detalhados na Regra 3. Assim, apesar de o mapeamento da generalização conceitual ser análogo ao mapeamento da generalização total/disjunta e total/sobreposta, na generalização conceitual os elementos representando as subclasses recebem tipos geométricos distintos, diferentemente da generalização. Como exemplo, a Tabela 3.2 mostra os documentos GML permitidos para cada tipo de generalização de um esquema OMT-G formado por uma superclasse *A* e duas subclasses *B* e *C*.

**Figura 3.6.** Esquemas GML representando um Nó e um Arco**Tabela 3.2.** Documentos permitidos para cada tipo de generalização dado um esquema OMT-G com uma superclasse  $A$  e duas subclasses  $B$  e  $C$ 

Tipo Generalização	Documento GML
Total/Disjunta	$\langle A \rangle \langle B \rangle \langle /A \rangle$
	$\langle A \rangle \langle C \rangle \langle /A \rangle$
Total/Sobreposta	$\langle A \rangle \langle B \rangle \langle /A \rangle$
	$\langle A \rangle \langle C \rangle \langle /A \rangle$
	$\langle A \rangle \langle B \rangle \langle C \rangle \langle /A \rangle$
Parcial/Disjunta	$\langle A \rangle \langle /A \rangle$
	$\langle A \rangle \langle B \rangle \langle /A \rangle$
	$\langle A \rangle \langle C \rangle \langle /A \rangle$
Parcial/Sobreposta	$\langle A \rangle \langle /A \rangle$
	$\langle A \rangle \langle B \rangle \langle /A \rangle$
	$\langle A \rangle \langle C \rangle \langle /A \rangle$
	$\langle A \rangle \langle B \rangle \langle C \rangle \langle /A \rangle$

### 3.1.5 Mapeamento de Atributos e Restrições

Atributos podem ser simples ou multivalorados com restrições de chave primária, chave estrangeira, tipo, tamanho e domínio.

*Regra 11: Atributos.* Para cada atributo simples ou multivalorado  $a_i$  de cada classe  $c_i \in O$ , criar um sub-elemento  $ea_i$  no elemento  $ec_i$  representando a classe  $c_i$  de acordo com seu tipo.

*Regra 12: Restrições de Atributos.* Caso  $a_i$  seja chave primária, incluir a restrição *key* em  $ea_i$ . Caso  $a_i$  seja mapeado como chave estrangeira, incluir a restrição *keyref* em  $ea_i$ . Caso  $a_i$  seja multivalorado, incluir as restrições *minOccurs* e *maxOccurs* em  $ea_i$  com a cardinalidade mínima e máxima. Caso  $a_i$  possua restrições de tamanho de *string*, incluir as restrições *minLength* e *maxLength* em  $ea_i$  com os tamanhos mínimos e máximos. Caso  $a_i$  possua domínio de valores, incluir a restrição *enumeration* em  $ea_i$  com os valores.

As primitivas do OMT-G e seus correspondentes na GML são resumidas na Tabela 3.3.

**Tabela 3.3.** Resumo do mapeamento entre as primitivas OMT-G e GML

Primitiva OMT-G (Regra)		Primitiva GML
Esquema OMT-G (1)		Elemento <i>root</i>
Domínio espacial (2)		Elemento <i>spatialDomain</i> com sub-elementos <i>boundedBy</i> e <i>extentOf</i>
Classe (3)	Convencional e Espacial	Elemento
	Ponto, Nó, Amostragem	<i>Point</i>
	Linha, Arco Unidirecional e Bidirecional	<i>LineString</i>
	Polígono, Subdivisão Planar	<i>Polygon</i>
	Isolinhas	<i>LineString</i> e/ou <i>Polygon</i>
	Triangulação	<i>Point</i> e <i>Polygon</i>
Relacionamento (4-5)	Tesselação	<i>Grid</i>
	Convencional, topológico, agregação	Elemento de acordo com as regras da Tabela 3.1
Generalização (6-10)	Em rede	Elementos $n1$ e $n2$
	Parcial/sobreposta	Elemento <i>sequence</i>
Atributo (11)	Outras	Elemento <i>choice</i>
	Simples	Elemento com tipo
	Multivalorado	Elemento com tipo e com restrições <i>minOccurs</i> e <i>maxOccurs</i>
Restrição (12)	Chave primária	<i>key</i>
	Chave estrangeira	<i>keyref</i>
	Tamanho de <i>string</i>	<i>minLength</i> e <i>maxLength</i>
	Domínio	<i>enumeration</i>

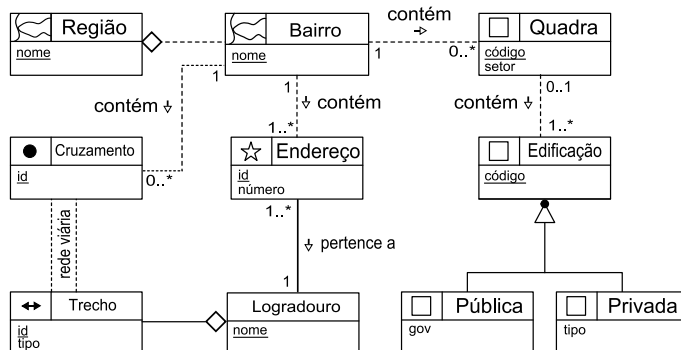
## 3.2 Algoritmo de Mapeamento

### 3.2.1 Descrição

O algoritmo de mapeamento proposto é baseado no trabalho de Pigozzo et al. [38]. O mapeamento consiste em duas etapas: a primeira determina os *elementos de primeiro nível* (*epn*), i.e., as classes que podem ser mapeadas como elementos-raiz no esquema GML, e a segunda gera o esquema GML. Existem dois tipos de elemento de primeiro nível: *epn1* e *epn2*.

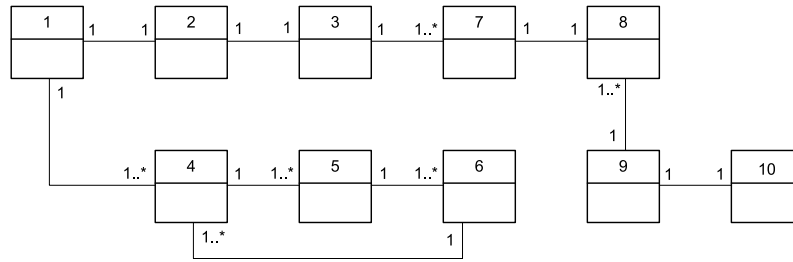
Os elementos *epn1* são determinados de acordo com a cardinalidade dos relacionamentos em que as classes participam. Assim para determinar se uma classe é *epn1* basta verificar os seus relacionamentos. Somente as seguintes condições (pelo menos uma) devem ser seguidas para uma classe ser mapeada como *epn1*: (1) a classe participa parcialmente em pelo menos um relacionamento, (2) a classe participa totalmente em pelo menos um relacionamento 1:N ou M:N, (3) a classe é uma superclasse em uma generalização e não participa de outro relacionamento, (4) a classe é a *todo* em uma agregação e não participa de outro relacionamento. Nas condições não são consideradas subclasses e classes *parte* de agregações nem os relacionamentos em rede. De fato, as condições (1) e (2) são apenas uma generalização para encontrar os elementos raiz na Tabela 3.1.

No esquema OMT-G da Figura 3.7 os elementos *epn1* são *Região*, *Logradouro* e *Edificação*. *Região* segue a condição (4), *Logradouro* segue a condição (2) e *Edificação* segue a condição (1). No entanto, dependendo da cardinalidade dos relacionamentos, existem casos em que elementos *epn1* não são encontrados. A Figura 3.8 mostra um exemplo de esquema OMT-G onde elementos *epn1* não são encontrados. Por isso, determina-se os elementos *epn2*. Esquemas OMT-G como o da Figura 3.8 podem ser vistos como um grafo direcionado  $G$ , onde a direção de uma aresta indica a cardinalidade (1,1), i.e., o aninhamento das estruturas (ex: (1,1)→(1,n)).



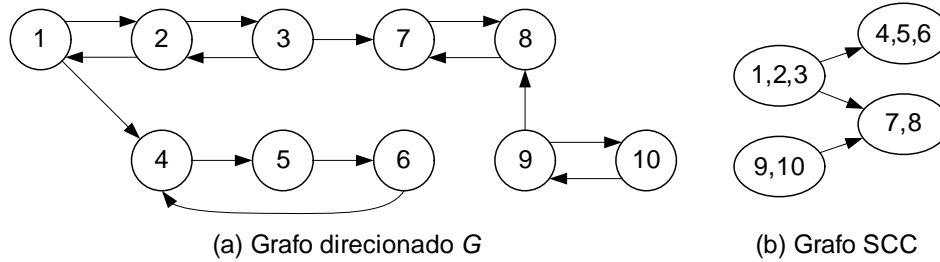
**Figura 3.7.** Exemplo de um esquema OMT-G

Os elementos *epn2* são determinados através do grafo direcionado  $G$ . Para uma classe ser mapeada como *epn2* ela deve ser um vértice pertencente ao grafo de *componentes fortemente conectado* (*strongly connected component* - SCC) de  $G$  que não possua arestas incidentes. Esse problema é denominado *problema de aninhamento da conectividade máxima* (*maximum connectivity nesting problem* - MCNP [12]) e garante a formação de florestas com conectividade máxima (i.e., um esquema com número máximo de elementos aninhados) quando elementos *epn2* são selecionados como elementos



**Figura 3.8.** Exemplo de um esquema OMT-G sem elementos *epn1*

raiz. A Figura 3.9a mostra o grafo direcionado  $G$  correspondente ao esquema da Figura 3.8, e a Figura 3.9b mostra seu grafo de componentes fortemente conectados, onde verifica-se que os elementos *epn2* são: 1, 2, 3, 9 e 10. O algoritmo para esse problema é descrito por Franceschet et al. [12].



**Figura 3.9.** Grafo direcionado  $G$  e seu grafo de componentes fortemente conectados com os elementos *epn2*: 1, 2, 3, 9 e 10

O Algoritmo 1 apresenta o pseudocódigo para o mapeamento entre os esquemas baseado nas regras apresentadas na Seção 3.1. As linhas 1 e 2 criam o elemento *root* e o *spatialDomain* (Regras 1 e 2). Na linha 3 os elementos de primeiro nível *epn1* e *epn2* são determinados. O laço entre as linhas 4 e 6 chama o procedimento *MapeiaClasse* para cada classe  $c \in O$  que ainda não foi mapeada para  $G$ , começando pelas classes em *epn1* e *epn2* (nessa ordem). Na linha 8 é criado o elemento *ec* representando a classe  $c$  como sub-elemento do elemento corrente (Regra 3). O elemento corrente é sempre o último elemento criado no esquema  $G$  representando uma classe ou um relacionamento. Na linha 9 são criados os atributos espaciais e não-espaciais de  $c$  como sub-elementos *ea* de *ec* (Regras 3, 11 e 12). O laço entre as linhas 10 e 27 mapeia cada relacionamento e generalização  $r \in O$  adjunto a  $c$  que ainda não foi mapeado para  $G$ . Para adequar a notação da Tabela 3.1, na linha 11 a classe  $c$  é chamada de  $a$  e a outra classe ou subclasse participante do relacionamento  $r$  é chamada de  $b$ ; os elementos representando  $a$  e  $b$  são chamados de *ea* e *eb*, respectivamente. Caso a classe  $b$  já esteja mapeada em  $G$ , um elemento *er* é criado representando  $r$  como

sub-elemento de *ea* apenas referenciando *eb* (linhas 12 e 13), o que garante a ausência de redundância de informação. Caso contrário, na linha 15 é verificado se *r* é um relacionamento convencional, topológico ou agregação, e pode ser criado como sub-elemento de *ea*, o que pode ser checado na Tabela 3.1<sup>4</sup>. As linhas 16 e 17 criam o elemento *er* representando *r* como sub-elemento de *ea* (Regra 4) e as restrições *key* e *keyref* dos relacionamentos de acordo com Tabela 3.1. Na linha 18 é verificado se *r* é um relacionamento em rede arco-nó, e na linha 19 são criados os elementos *n1* e *n2* como sub-elementos de *ea* referenciando *eb* (Regra 5). Na linha 20 é verificado se *r* é uma generalização e tem *a* como superclasse, e na linha 21 são criados os elementos *choice* ou *sequence* como sub-elementos de *ea* dependendo do tipo de generalização (Regras 6 a 10). A linha 25 chama recursivamente o procedimento *MapeiaClasse* se a classe *b* pode ser aninhada com a classe *a* e ainda não foi mapeada para *G*. Estruturas aninhadas podem ser criadas quando *r* é um relacionamento convencional ou topológico com cardinalidade (1,1), agregação ou generalização.

---

**Algoritmo 1** Mapeamento de esquemas OMT-G para esquemas GML

---

**Entrada:** esquema OMT-G *O*

**Saída:** esquema GML *G*

```

1: Criar o elemento root no esquema GML G
2: Criar o domínio espacial spatialDomain como sub-elemento de root
3: Determinar as classes elementos de primeiro nível epn1 e epn2
4: for cada classe c ∈ O que ainda não foi mapeada para G, começando pelas classes em epn1 e epn2 do
5:   MapeiaClasse(c)
6: end for

7: Procedimento MapeiaClasse(c: classe)
8: Criar um elemento ec representando c e adicionar como sub-elemento do elemento corrente
9: Criar elementos ea representando os atributos espaciais e não-espaciais de c com suas restrições e adicionar
   como sub-elemento de ec
10: for cada rel. ou generalização r ∈ O de c e que ainda não foi mapeado para G do
11:   Seja a a classe c e b a outra classe participante em r ou uma subclasse; ea e eb elementos representando
   a e b
12:   if (b já está mapeada) then
13:     Criar um elemento er representando r e adicionar como sub-elemento de ea referenciando eb
14:   else
15:     if (r é rel. convencional, topológico ou agregação, e pode ser criado como sub-elemento de ea) then
16:       Criar um elemento er representando r e adicionar como sub-elemento de ea
17:       Criar restrições key e keyref para er de acordo a cardinalidade de r
18:     else if (r é um rel. em rede e a e b são classes do tipo arco e nó) then
19:       Criar dois elementos n1 e n2 e adicionar com sub-elementos de ea referenciando eb
20:     else if (r é uma generalização e a é superclasse) then
21:       Criar um elemento choice ou sequence e adicionar como sub-elemento de ea
22:     end if
23:   end if
24:   if (b ainda não foi mapeada para G e pode ser aninhada com a) then
25:     MapeiaClasse(b)
26:   end if
27: end for

```

---

Observe que os esquemas GML gerados pelo Algoritmo 1 podem depender da

---

<sup>4</sup>Ex: Na linha 1 e coluna 2 (A(0,n) e B(0,1)) da Tabela 3.1, *r* é criado sub-elemento de *ea*. Na linha 2 e coluna 1 (A(0,1) e B(0,n)) *r* não é criado sub-elemento de *ea*

ordem em que as classes são examinadas (linha 4) e da ordem em que os relacionamentos e as generalizações de uma classe são alcançados (linha 10). Essas ordens de visitação diferentes não causam problemas (desde que seja mantida a ordem de iniciar o mapeamento pelos elementos *epn*, como pode ser visto na Seção 3.2.2), uma vez que os esquemas GML gerados são essencialmente equivalentes. Uma solução para esse problema seria gerar um esquema GML para cada combinação de elementos de primeiro nível, cabendo ao modelador selecionar o esquema GML a ser utilizado.

Com relação ao tempo de execução do Algoritmo 1, considere o esquema OMT-G um grafo  $G = (V, E)$ , onde  $V$  são as classes e  $E$  são os relacionamentos. O tempo para determinar elementos *epn1* e *epn2* é  $\Theta(V + E)$  [12]. O laço entre as linhas 4 e 6 demora o tempo  $\Theta(V)$ , fora o tempo para executar as chamadas de *MapeiaClasse*. O procedimento *MapeiaClasse* é executado exatamente uma vez para cada vértice  $v \in V$ , pois o procedimento *MapeiaClasse* é invocado somente sobre vértices que ainda não foram mapeados. Durante uma execução do procedimento *MapeiaClasse(v)*, o laço entre as linhas 10 e 27 é executado  $|Adjuntos(v)|$  vezes, onde  $Adjuntos(v)$  é o número de relacionamentos vizinhos de  $v$ . Tendo em vista que  $\sum_{v \in V} |Adjuntos(v)| = \Theta(E)$ , o custo total da execução das linhas 10 a 27 é  $\Theta(E)$ . Logo, o tempo de execução do Algoritmo 1 é  $\Theta(V + E)$ .

Uma ferramenta, chamada OMTG2GML<sup>5</sup>, foi desenvolvida para automatizar a geração de esquemas GML. A ferramenta implementa o Algoritmo 1 e recebe como entrada um esquema OMT-G sob o formato de um documento XML, retornando um esquema GML. No Apêndice A são encontradas mais informações sobre a ferramenta e o formato do documento XML.

### 3.2.2 Comparação de Mapeamentos

Conforme foi discutido, elementos raiz são determinados através dos elementos *epn* obtidos no esquema OMT-G. Considere o esquema OMT-G da Figura 3.8. Iniciando-se o mapeamento pelas classes que não estão em *epn* pode-se obter um esquema com 4 raízes, 4 restrições *keyref* e profundidade 6. Já iniciando-se o mapeamento pelas classes em *epn* obtém-se um esquema com 2 raízes, 2 restrições *keyref* e profundidade 10, resultando em um esquema com menos restrições e maior aninhamento.

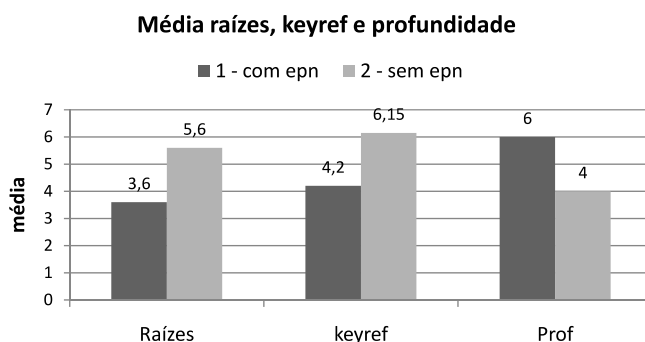
Tendo em vista a escolha dos elementos raiz para a geração de esquemas XML, foi realizado um estudo onde comparou-se os mapeamentos de esquemas OMT-G iniciados (1) pelas classes que estavam em *epn* e (2) pelas classes que não estavam em *epn*. Para isso, foram coletados vinte esquemas ER e EER das mais variadas aplica-

<sup>5</sup>A ferramenta OMTG2GML está disponível em <http://www.lbd.dcc.ufmg.br/lbd/tools>



ções de bancos de dados e transformados em esquemas OMT-G convencionais. Para cada esquema OMT-G, o Algoritmo 1 foi executado conforme (1) e (2), gerando dois esquemas XML. A Tabela 3.4 mostra os resultados obtidos. As três primeiras colunas mostram o identificador, a quantidade de classes e a quantidade de relacionamentos e generalizações, respectivamente, dos esquemas OMT-G. As demais colunas mostram a quantidade de classes mapeadas como elementos raiz, a quantidade de restrições *keyref* e a profundidade, respectivamente, dos esquemas XML gerados conforme (1) e (2). Pela análise da tabela, verifica-se que iniciando o mapeamento pelos elementos *epn* são gerados, de modo geral, esquemas XML com menos elementos raiz, com menos restrições e mais profundos. Isso resulta em documentos XML do tipo (1) com menor tempo de validação e menor tempo de processamento de consultas que documentos XML do tipo (2). Em alguns casos a diferença é bastante acentuada, como nos esquemas 4, 7 e 15. Mesmo nos casos em que as diferenças não são grandes (ex: 12, 13, 18), resultados significativos com relação ao tempo de validação e processamento de consultas podem ser obtidos [41]. A Figura 3.10 mostra um gráfico com as médias da quantidade de raízes, *keyref* e profundidade. Os esquemas do tipo (2) possuem em média 55,55% mais elementos raiz que os esquemas do tipo (1) e 46,42% mais restrições *keyref*. Os esquemas do tipo (1) são em média 50% mais profundos que os esquemas do tipo (2).

Esses dados confirmam a importância da escolha dos elementos *epn* na geração dos esquemas. Assim, a seleção correta dos elementos raiz impacta na geração de esquemas mais aninhados e com menos restrições *keyref*.



**Figura 3.10.** Média da quantidade de raízes, *keyref* e profundidade dos esquemas dos tipos (1) e (2)

**Tabela 3.4.** Quantidade de raízes, keyref e profundidade em cada esquema gerados através de (1) e (2)

Id	Classes	Rel e Gen	Raízes		keyref		Prof	
			(1)	(2)	(1)	(2)	(1)	(2)
1	11	11	5	6	2	3	5	5
2	15	11	5	6	5	6	5	3
3	13	14	4	6	5	6	9	4
4	6	7	2	6	3	7	5	2
5	4	5	1	3	2	4	5	4
6	8	9	2	4	2	5	10	6
7	7	9	1	5	3	7	8	4
8	5	7	3	4	5	6	4	4
9	6	5	3	5	2	4	7	3
10	8	10	3	4	5	6	7	5
11	7	8	4	6	3	5	4	3
12	10	8	3	4	4	5	5	4
13	9	7	4	5	3	4	5	4
14	10	11	7	8	8	9	6	4
15	10	11	1	7	2	8	7	4
16	11	8	3	5	2	4	7	3
17	29	32	10	13	19	21	5	7
18	10	9	5	6	5	6	4	3
19	11	7	3	4	2	3	5	3
20	8	9	3	5	2	4	7	5
Total	198	198	72	112	84	123	120	80

### 3.3 Comparação com Outras Abordagens

Visando comparar as técnicas de transformação, o mapeamento proposto e outros relacionados foram utilizados para a conversão de um esquema OMT-G em esquemas GML. Foram selecionados trabalhos que contém descrições detalhadas de suas transformações (i.e., regras e algoritmos) ou que disponibilizam ferramentas para a realização das transformações. Em seguida, documentos GML foram gerados com base nos esquemas GML e consultas foram executadas nesses documentos. Os resultados mostram que o mapeamento proposto gera resultados eficientes com relação ao tempo de processamento das consultas.

#### 3.3.1 Geração de Esquemas GML

A Figura 3.7 apresenta o esquema OMT-G utilizado nesta comparação. As classes *Região* e *Bairro* são representadas como geocampos. As classes *Quadra*, *Endereço*, *Edificação*, *Pública*, *Privada*, *Cruzamento* e *Trecho* são representadas como geo-objetos. *Logradouro* é uma classe convencional. São apresentados alguns relacionamentos topológicos *contém*; uma agregação espacial entre as classes *Região* e *Bairro*; uma agregação convencional entre as classes *Logradouro* e *Trecho*; um relacionamento em rede entre as classes *Trecho* e *Cruzamento*; e um relacionamento convencional entre as classes *Logradouro* e *Endereço*. Também verifica-se uma generalização do tipo total/disjunta

onde *Edificação* é a superclasse.

Esse esquema OMT-G foi mapeado em quatro esquemas GML: um *flat* e três *aninhados*. O esquema *flat* possui um mapeamento direto, ou seja, as classes são convertidas como elementos raiz. O primeiro esquema *aninhado*, chamado de *Relacionado-1*, foi gerado através da abordagem de Liu e Li [28], o segundo, *Relacionado-2*, utiliza a abordagem de Franceschet et al. [12, 13] e o terceiro, *O2G*, utiliza abordagem proposta. Como discutido na Seção 2.4, o algoritmo que gerou o esquema *Relacionado-1* propõe a criação de esquemas XML a partir de esquemas ER, assim foram introduzidas regras nessa abordagem para acomodar as primitivas geográficas do OMT-G. O esquema *Relacionado-2* foi gerado através da ferramenta ChronoGeoGraph<sup>6</sup>.

As Figuras 3.11, 3.12, 3.13 e 3.14 mostram a estrutura hierárquica<sup>7</sup> dos esquemas GML *O2G*, *Relacionado-2*, *Relacionado-1* e *flat*, respectivamente. Nessas estruturas é possível verificar através de uma árvore os principais elementos GML. Os retângulos indicam elementos XML `element`, os retângulos tracejados indicam referências para outros elementos e as conexões entre os elementos indicam a hierarquia, onde *root* é o elemento raiz. Também são mostradas as cardinalidades dos elementos; sua omissão indica a cardinalidade 1..1.

A seguir é descrita passo a passo a execução do Algoritmo 1. No esquema *O2G* (Figura 3.11), as classes *Região*, *Logradouro* e *Edificação* são mapeadas como elementos de primeiro nível. O elemento *spatialDomain* com suas propriedades é criado como um sub-elemento do elemento *root*. Começando o mapeamento pela classe *Região*, o elemento *Região*, com seus atributos, é criado como sub-elemento de *root*. A agregação espacial com *Bairro* é mapeada como sub-elemento de *Região*. Como a classe *Bairro* ainda não foi mapeada e pode ser aninhada com *Região*, o procedimento *MapeiaClasse* é chamado recursivamente com a classe *Bairro*. Assim, o elemento *Bairro*, com seus atributos, é criado como sub-elemento do elemento corrente, que nesse caso é o elemento representando a agregação espacial. A classe *Bairro* possui três relacionamentos adjuntos ainda não mapeados: *contém Quadra*, *contém Cruzamento* e *contém Endeço*. Começando por *contém Quadra*, *contém* é mapeado como sub-elemento de *Bairro* devido à sua cardinalidade, e novamente ocorre a chamada recursiva com a classe *Quadra*. O elemento *Quadra*, com seus atributos, é criado como sub-elemento do elemento corrente, que nesse caso é o *contém*. A classe *Quadra* possui apenas o relacionamento *contém Edificação* que ainda não foi mapeado; contudo, devido à sua cardinalidade, ele é mapeado como sub-elemento de *Quadra* com apenas uma referência para *Edificação*.

<sup>6</sup>ChronoGeoGraph tool: <http://dbms.dimi.uniud.it/cgg>

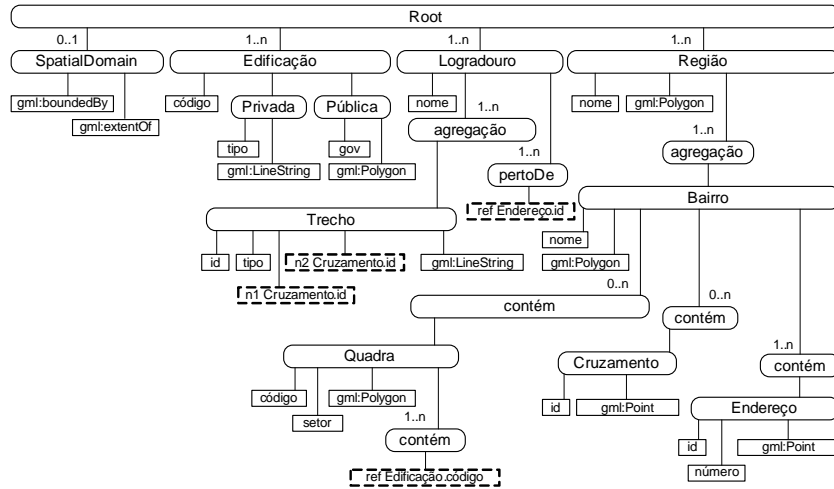
<sup>7</sup>Para simplificar a visualização dos esquemas GML, elementos XML como `complexType`, `sequence` e `choice` são omitidos, sendo mostrados apenas os principais elementos `element`

Como *Edificação* não pode ser aninhado em *Quadra*, essa chamada recursiva termina. Voltando aos relacionamentos adjuntos a *Bairro*, o próximo a ser mapeado é *contém Cruzamento*, que é mapeado como sub-elemento de *Bairro* de modo semelhante ao mapeamento de *contém Quadra*, ocorrendo a chamada recursiva para a classe *Cruzamento*. O elemento *Cruzamento*, com seus atributos, é criado como sub-elemento de *contém*. Como *Cruzamento* é uma classe do tipo Nó e só possui o relacionamento em rede adjunto ainda não mapeado, essa chamada recursiva termina. Assim, o próximo relacionamento a ser mapeado é *contém Endereço*, que também é mapeado como sub-elemento de *Bairro* de modo semelhante ao mapeamento de *contém Quadra*, ocorrendo a chamada recursiva para a classe *Endereço*. O elemento *Endereço*, com seus atributos, é criado como sub-elemento de *contém*. *Endereço* possui apenas o relacionamento *perto de Logradouro* adjunto ainda não mapeado, contudo, devido à sua cardinalidade, ele não é mapeado como sub-elemento de *Endereço*. Com isso, a chamada recursiva para *Bairro* termina.

A próxima classe a ser mapeada é *Logradouro*. O elemento *Logradouro*, com seus atributos, é criado como sub-elemento de *root*. A classe *Logradouro* possui dois relacionamentos adjuntos ainda não mapeados: a agregação convencional com *Trecho* e *perto de Endereço*. Começando pela agregação convencional com *Trecho*, esta é mapeada como sub-elemento de *Logradouro*. Como a classe *Trecho* ainda não foi mapeada e pode ser aninhada com *Logradouro*, ocorre a chamada recursiva com a classe *Trecho*. Assim, o elemento *Trecho*, com seus atributos, é criado como sub-elemento do elemento corrente, que nesse caso é o elemento representando a agregação convencional. A classe *Trecho* é do tipo Arco e possui apenas o relacionamento em rede ainda não mapeado. Assim, o relacionamento em rede é mapeado como sub-elemento de *Trecho* através da criação dos elementos *n1* e *n2* referenciando *Cruzamento*, e a chamada recursiva para *Trecho* termina. O próximo relacionamento a ser mapeado é *perto de Endereço*. Como a classe *Endereço* já foi mapeada, o relacionamento *perto de Endereço* é mapeado como sub-elemento de *Logradouro* com apenas uma referência para *Endereço*.

A próxima classe mapeada é *Edificação*. O elemento *Edificação*, com seus atributos, é criado como sub-elemento de *root*. Como *Edificação* é uma superclasse em uma generalização total/disjunta o elemento *choice* é criado como sub-elemento de *Edificação*. Assim, ocorre uma chamada recursiva para a subclasse *Público* e uma para *Privado*, onde os elementos representando as subclasses e seus atributos são criados como sub-elementos do elemento corrente, que nesse caso é *choice*. Com isso, as chamadas recursivas para as subclasses terminam. Como todas classes no esquema OMT-G foram mapeadas para o esquema GML, a execução do Algoritmo 1 termina.

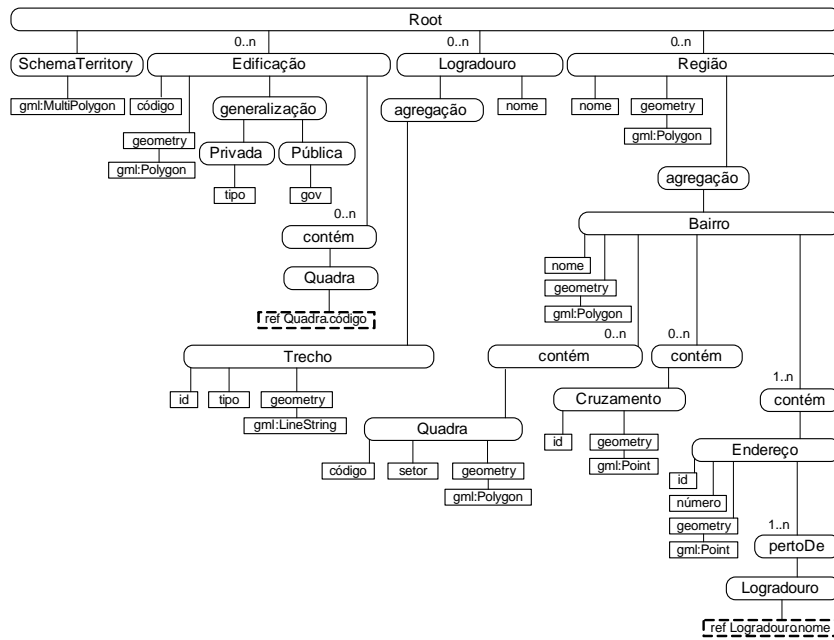
O esquema *Relacionado-2* é apresentado na Figura 3.12. Note que os esquemas



**Figura 3.11.** Estrutura hierárquica do esquema gerado pela abordagem proposta (*O2G*) com seus principais elementos

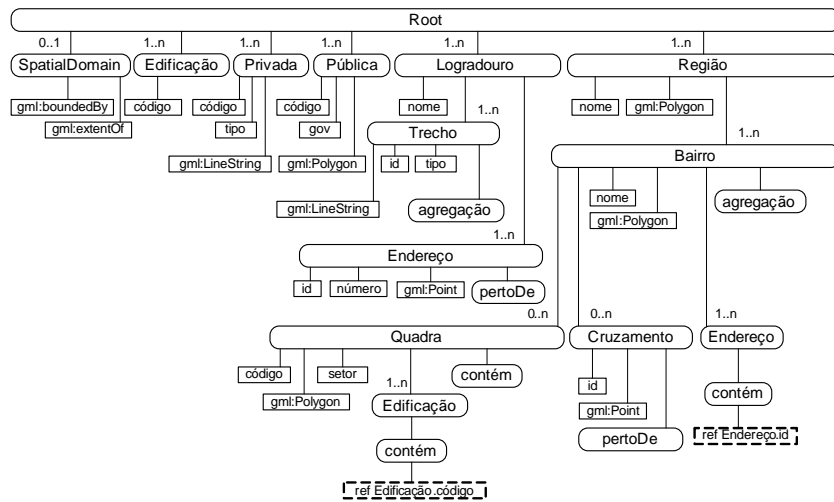
*Relacionado-2* e *O2G* apresentam algumas similaridades. O aninhamento da maioria das estruturas ocorre de forma semelhante. No entanto, algumas inconsistências foram geradas no esquema *Relacionado-2* em relação ao esquema OMT-G. A primeira inconsistência ocorre no mapeamento do relacionamento *perto de* entre *Logradouro* e *Endereço*. Esse relacionamento foi mapeado como sub-elemento de *Endereço* com cardinalidade 1..n quando deveria ser mapeado como sub-elemento de *Logradouro*, gerando assim um resultado inverso ao esperado. A segunda inconsistência ocorre no mapeamento do relacionamento *contém* entre *Quadra* e *Edificação*. Esse relacionamento foi mapeado como sub-elemento de *Edificação* com cardinalidade 0..n e deveria ser mapeado como sub-elemento de *Quadra*. Há, entretanto, algumas semelhanças entre as duas abordagens de mapeamento: a criação do domínio espacial denominado *SchemaTerritory* em *Relacionado-2* é equivalente à propriedade *boundedBy* em *O2G*; as generalizações são tratadas de forma similares através da utilização de elementos de sequência e escolha; e as geometrias são mapeadas através do elemento *geometry* em *Relacionado-2* e de forma direta em *O2G*.

O esquema *Relacionado-1* é apresentado na Figura 3.13. Existem três diferenças principais entre o mapeamento dos esquemas gerados por *Relacionado-1* e *O2G*. A primeira diferença é no mapeamento das generalizações, onde o relacionamento *IS-A* é utilizado (através de uma extensão do modelo ER equivalente à generalização total/-disjunta) através do elemento XML *extension*. Assim, é possível representar apenas um tipo de generalização. A segunda diferença é no posicionamento dos elementos representando os relacionamentos, que em alguns casos (dependendo da cardinalidade) são mapeados como sub-elementos dos elementos representando as classes participan-



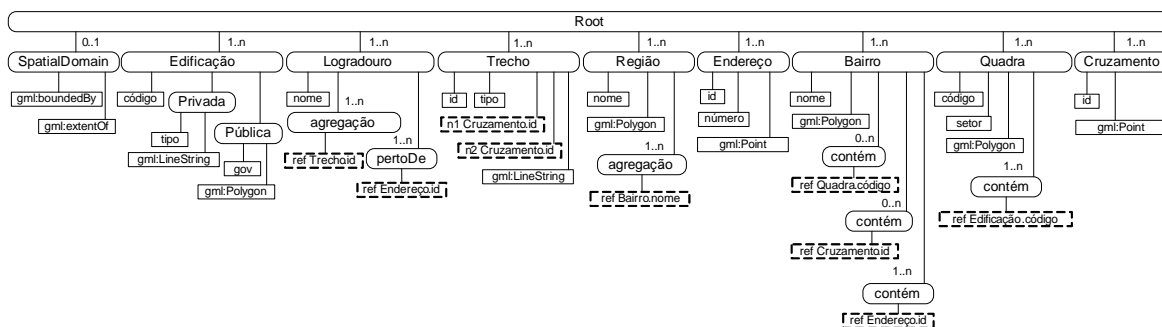
**Figura 3.12.** Estrutura hierárquica do esquema gerado por *Relacionado-2* com seus principais elementos

tes (por exemplo, a agregação entre *Região* e *Bairro* é mapeada como sub-elemento de *Bairro*, e não como sub-elemento de *Região*). A última diferença ocorre no mapeamento dos relacionamentos simétricos (i.e., relacionamentos com ambas as cardinalidades iguais). A abordagem *Relacionado-1* utiliza o conceito de *classe dominante* em um relacionamento, o que introduz uma etapa manual no algoritmo, uma vez que a classe dominante é selecionada pelo modelador.



**Figura 3.13.** Estrutura hierárquica do esquema gerado por *Relacionado-1* com seus principais elementos

O esquema *flat* é apresentado na Figura 3.14. Nesse esquema as classes são mapeadas como elementos raiz. Esses elementos utilizam referências *keyref* para representar os relacionamentos entre as classes. Note que, apesar do mapeamento ocorrer de forma direta, muitas referências são criadas para que estrutura do esquema OMT-G seja preservada.



**Figura 3.14.** Estrutura hierárquica do esquema gerado por *flat* com seus principais elementos

### 3.3.2 Consultas nos Documentos GML

Dados geográficos representando o esquema OMT-G da Figura 3.7 foram carregados no Oracle, utilizando informações de um banco de dados espacial da cidade de Belo Horizonte<sup>8</sup>. A partir dos dados geográficos carregados no banco de dados, foram criados documentos GML<sup>9</sup> para os esquemas *flat*, *Relacionado-1* e *O2G* (*Relacionado-2* não foi considerado devido às suas inconsistências). Os documentos GML foram criados através de scripts PL/SQL e funções do Oracle XML DB e Spatial, tais como *xmlement*, *xmlforest*, *xmltype* e *to\_gmlgeometry*. Os documentos contêm 3 *Regiões*, 124 *Bairros*, 3.230 *Quadras*, 14.017 *Endereços*, 2.188 *Logradouros*, 12.100 *Trechos*, 7.651 *Cruzamentos* e 17.355 *Edificações*.

Dez consultas espaciais e não-espaciais foram executadas em cada documento GML visando verificar os tempos de processamento. É esperado que a estratégia de aninhamento possa gerar melhores resultados devido à redução de acesso a referências e a redução de junções (substituída pela estrutura aninhada) durante a execução das consultas. As consultas não-espaciais exploram as estruturas do esquema GML, de modo que os principais elementos e os aninhamentos sejam cobertos. As consultas espaciais

<sup>8</sup>Dados obtidos junto a Prefeitura de Belo Horizonte

<sup>9</sup>Os documentos GML e as consultas estão disponíveis em <http://www.lbd.dcc.ufmg.br/lbd/tools>

exploram características geográficas que não podem ser deduzidas do documento, tais como área, distância e polígonos adjacentes.

Foram utilizadas as linguagens XQuery e GQL para a realização das consultas. GQL é uma extensão do XQuery que suporta funções para análise espacial (ex: *distance*, *convexHull* e *area*), funções para testar relações espaciais entre duas geometrias (ex: *intersects*, *touches*, *contains* e *overlaps*) e outras, diretamente em documentos GML. Essas funções realizam suas operações diretamente na geometria (coordenadas dos pontos, linhas e polígonos) contidas nos documentos GML.

As consultas em XQuery foram executadas no SGBD XML nativo BaseX<sup>10</sup> [14] e as consultas em GQL foram executadas no XQLPlus [16]. A Tabela 3.5 mostra a descrição das consultas. As consultas 1 a 5 são não-espaciais e especificadas em XQuery e as consultas 6 a 10 são espaciais e especificadas em GQL. As consultas foram criadas de acordo com a estrutura de cada esquema GML, assim, para cada descrição, três consultas foram implementadas. A Figura 3.15 mostra a implementação das consultas Q1 e Q6 para os esquemas *O2G*, *Relacionado-1* e *flat*.

**Tabela 3.5.** Descrição das consultas não-espaciais (Q1-Q5) e espaciais (Q6-Q10)

Id	Descrição	Id	Descrição
Q1	Recuperar os bairros relacionados a região Centro-Sul	Q6	Determinar <i>área</i> dos bairros contidos nas regiões Centro-Sul e Leste
Q2	Recuperar as quadras relacionadas as regiões Centro-Sul e Oeste	Q7	Recuperar as quadras localizadas <i>a menos de 80 metros</i> dos bairros Barroca, Alto Barroca e Savassi
Q3	Recuperar os endereços relacionados ao bairro Centro	Q8	Recuperar a <i>área</i> dos bairros <i>adjacentes</i> aos bairros Gutierrez, Estoril e Jonas Veiga
Q4	Recuperar os endereços e os logradouros relacionados ao bairro Funcionários	Q9	Determinar o <i>comprimento</i> dos trechos que <i>cruzam</i> os bairros contidos na região Leste
Q5	Determinar a quantidade de edificações relacionadas a cada quadra das regiões Centro-Sul e Leste considerando apenas quadras com setor < 10	Q10	Recuperar os cruzamentos dos bairros <i>adjacentes</i> aos bairros Esplanada, Santo Agostinho, Salgado Filho e Estoril

### 3.3.3 Resultados

As consultas foram executadas em um computador 2 GHz Intel Core2Duo com 3 GB de RAM. O tempo de processamento em segundos das consultas não-espaciais e espaciais é apresentado na Tabela 3.6. As colunas se referem ao tempo de resposta das consultas nos documentos GML *flat*, *Relacionado-1* e *O2G*. Percebe-se que as duas abordagens que utilizam o aninhamento (i.e., *Relacionado-1* e *O2G*) apresentam resultados mais eficientes do que a abordagem *flat*, visto que nessa é necessário o uso frequente de junções.

<sup>10</sup>BaseX: <http://www.inf.uni-konstanz.de/dbis/basex>



```

for $r in root/regiao[nome="CENTRO SUL"]/agregacao/bairro
return
  <nome>{$r/nome/text()}</nome>
(a) Consulta Q1 para O2G

for $r in root/regiao[nome="CENTRO SUL"]/bairro
where exists($r/agregacao)
return
  <nome>{$r/nome/text()}</nome>
(b) Consulta Q1 para Relacionado-1

for $r in root/regiao[nome="CENTRO SUL"],
  $n in root/bairro[nome=$r/agregacao/@ref]
return
  <nome>{$n/nome/text()}</nome>
(c) Consulta Q1 para flat

for $n in root/regiao[nome="CENTRO SUL" or nome="LESTE"]/agregacao/bairro
let $area := Area($n/gml:Polygon)
return
  <bairro>
    <nome>{$n/nome/text()}</nome>
    <area>{$area}</area>
  </bairro>
(d) Consulta Q6 para O2G

for $n in root/regiao[nome="CENTRO SUL" or nome="LESTE"]/bairro[agregacao]
let $area := Area($n/gml:Polygon)
return
  <bairro>
    <nome>{$n/nome/text()}</nome>
    <area>{$area}</area>
  </bairro>
(e) Consulta Q6 para Relacionado-1

for $r in root/regiao[nome="CENTRO SUL" or nome="LESTE"],
  $n in root/bairro
let $area := Area($n/gml:Polygon)
where $r/agregacao/@ref = $n/nome
return
  <bairro>
    <nome>{$n/nome/text()}</nome>
    <area>{$area}</area>
  </bairro>
(f) Consulta Q6 para flat

```

**Figura 3.15.** Consultas Q1 (a,b,c) em XQuery e Q6 (d,e,f) em GQL para cada esquema

Na Tabela 3.6 (Q1-Q5) verifica-se que *O2G* gerou resultados melhores do que *Relacionado-1* em todas as consultas. Isso se deve às diferenças entre suas estruturas aninhadas, conforme verifica-se nas Figuras 3.11 e 3.13. Em *Relacionado-1* é necessário testar a existência de um relacionamento entre dois elementos (como pode ser visto nas Figuras 3.15b e 3.15e, onde o relacionamento *agregação* é testado) enquanto que em *O2G* o relacionamento faz parte da própria estrutura aninhada.

Consultas espaciais têm um custo de tempo em geral superior ao das consultas não-espaciais, devido às suas operações de natureza topológica [48]. Diante disso, poderíamos concluir, de modo precipitado, que consultas espaciais em documentos GML *flat* e *aninhados* compartilhariam custo de tempo aproximados, uma vez que as operações espaciais consumiriam em ambos a maior parte do tempo de processamento. Entretanto, como pode-se observar na Tabela 3.6 (Q6-Q10), apesar de as operações espaciais consumirem muito tempo, consultas a documentos GML criados através de abordagens que visam o aninhamento das estruturas resultam em melhores resultados. Verifica-se também que *O2G* gerou melhores resultados que *Relacionado-1* em todas as consultas. Novamente, isso se deve às diferenças entre suas estruturas aninhadas.

**Tabela 3.6.** Tempo de processamento das consultas em segundos

<b>Id</b>	<b>Flat</b>	<b>Relacionado-1</b>	<b>O2G</b>	<b>Id</b>	<b>Flat</b>	<b>Relacionado-1</b>	<b>O2G</b>
Q1	0.09	0.03	0.02	Q6	2.85	2.79	2.61
Q2	29.99	0.11	0.09	Q7	10.41	3.93	3.77
Q3	14.83	61.19	0.22	Q8	150.90	29.17	27.78
Q4	13.33	29.84	8.48	Q9	259.52	162.60	162.07
Q5	29.72	0.35	0.26	Q10	24.14	4.20	3.91

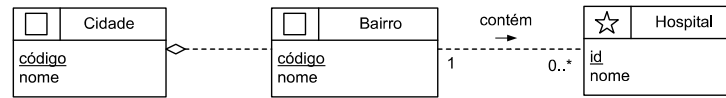
## Capítulo 4

# Mapeamento para Esquemas Físicos

O mapeamento de esquemas geográficos OMT-G em esquemas físicos ocorre de forma semelhante à transformação de esquemas conceituais convencionais (ex., ER e EER) quando as primitivas não-geográficas são consideradas. Quando as primitivas geográficas são consideradas, o processo deve acomodar alguns requisitos adicionais. Classes e relacionamentos espaciais possuem representações geográficas que devem ser preservadas no banco de dados. Além disso, esquemas geográficos possuem restrições de integridade espaciais.

A Figura 4.1 apresenta um esquema OMT-G. São definidas três classes: *Cidade*, *Bairro* e *Hospital*. Existem dois relacionamentos espaciais, uma agregação espacial entre as classes *Cidade* e *Bairro*, e um relacionamento topológico *contém* entre as classes *Bairro* e *Hospital*. Nesse esquema, três representações geográficas são indicadas através dos tipos geométricos das classes. Existem também cinco restrições de integridade espaciais, três relacionadas ao tipo geométrico adotado para cada classe, e dois relacionados os relacionamentos espaciais (ex: um hospital deve estar contido em um bairro). Esse exemplo simples mostra que, no mapeamento entre esquemas conceituais geográficos e físicos, detalhes referentes às primitivas espaciais devem ser considerados. Parte da transformação envolve o mapeamento de classes e relacionamentos convencionais para tabelas e restrições de integridade referenciais. As primitivas espaciais devem ser mapeadas com o acréscimo de restrições de integridade, implementadas usando *triggers* ou cláusulas *check*. Além disso, devem ser observados detalhes como inserções em tabelas de metadados e definições de índices espaciais para atributos espaciais.

Diante dessas características intrínsecas aos esquemas conceituais geográficos, este capítulo define o mapeamento de esquemas OMT-G para esquemas físicos de bancos



**Figura 4.1.** Exemplo de um esquema OMT-G

de dados espaciais. A conversão entre os esquemas ocorre sem perdas semânticas e estruturais, preservando as informações existentes no esquema conceitual geográfico, como as classes, os relacionamentos espaciais e não-espaciais e as generalizações. Para automatizar a transformação entre os esquemas, uma ferramenta foi desenvolvida baseada no mapeamento proposto. A Seção 4.1 apresenta o mapeamento para esquemas lógicos e a Seção 4.2 para esquemas físicos de bancos de dados espaciais.

## 4.1 Mapeamento para Esquemas Lógicos

Considerando a tendência atual dos sistemas de gerência de bancos de dados espaciais seguirem os padrões do *Open Geospatial Consortium* (OGC [35]), são consideradas apenas representações definidas no OGC *Simple Features Specification*, i.e., *point*, *linestring*, *polygon*, *multipoint*, *multilinestring*, *multipolygon* e *geometry collection*.

Primeiramente, são mostrados os mapeamentos das classes convencionais e espaciais. Em seguida, o mapeamento dos relacionamentos convencionais e espaciais e das generalizações. Observa-se que relacionamentos espaciais não precisam ser materializados no esquema de implementação, uma vez que a associação entre os objetos envolvidos é definida dinamicamente, no momento da consulta. Os relacionamentos espaciais podem, por outro lado, gerar restrições de integridade, para que o comportamento espacial esperado no esquema conceitual seja garantido pelo banco de dados. Detalha-se a seguir os quatro principais passos do mapeamento de esquemas conceituais OMT-G para esquemas lógicos, inspirados em Elmasri & Navathe [11].

*Passo 1: Mapeamento de classes convencionais e espaciais.* Para cada classe convencional  $C$  no esquema OMT-G, criar uma relação  $R$  contendo todos os atributos simples de  $C$ . Escolher um dos atributos-chave de  $C$  para ser a chave primária da relação  $R$ . O mesmo procedimento se aplica a classes espaciais, decidindo-se adicionalmente a alternativa de representação segundo os tipos geométricos disponíveis no banco de dados escolhido.

A Tabela 4.1 apresenta uma correspondência entre os tipos geométricos básicos do modelo OMT-G e os propostos pela OGC. As representações de geocampos exigem também o estabelecimento de restrições de integridade referentes à sua representação

(*R2* a *R5*). Observa-se que tesselações no OMT-G podem corresponder a dois tipos de representação física sutilmente diferentes: imagens digitais e grades regulares. Assim, caso a representação conceitual seja uma tesselação, pode-se optar entre uma representação matricial própria do SGBD (como a GeoRaster do Oracle Spatial) ou um campo binário longo, para conter dados de um determinado formato de imagem.

**Tabela 4.1.** Tipos geométricos: OMT-G e OGC

Representação OMT-G	Representação OpenGIS ( <i>Simple Features Specification</i> )
Ponto	Point
Linha	LineString
Polígono	Polygon
Arco Unidirecional	LineString
Arco Bidirecional	LineString
Amostragem	Point ou MultiPoint
Isolinhas	LineString
Subdivisão Planar	Polygon ou MultiPolygon
Triangulação	Point (vértice) e Polygon (triângulo)
Tesselação	GeoRaster ou BLOB

*Passo 2: Mapeamento de relacionamentos convencionais.* Para cada relacionamento convencional de cardinalidade 1:1, escolher uma das classes e incluir nela a chave primária da outra, no papel de chave estrangeira. Para relacionamentos convencionais de cardinalidade 1:N, incluir na relação correspondente à classe do lado N, como chave estrangeira, a chave primária da relação correspondente à classe do lado 1. No caso de relacionamentos convencionais de cardinalidade M:N, criar uma relação *R* intermediária, contendo as chaves primárias de ambas as relações envolvidas, no papel de chaves estrangeiras de suas respectivas relações, e formando, juntas, a chave primária da nova relação. Tratar os relacionamentos de agregação convencionais de forma semelhante aos relacionamentos com cardinalidade 1:N. Assim, o mapeamento de classes convencionais envolve apenas a criação restrições de integridade referenciais. O tratamento de relacionamentos convencionais independe da representação geográfica das classes envolvidas.

*Passo 3: Mapeamento de relacionamentos espaciais.* Relacionamentos espaciais explicitados em esquemas OMT-G não são necessariamente materializados nos esquemas lógicos e físicos, e sua existência pode ser verificada no banco de dados através de funções topológicas. Assim, eles indicam no projeto lógico o relacionamento esperado entre instâncias das classes envolvidas, e requerem a implementação de restrições de integridade espaciais na etapa física. Logo, o mapeamento ideal de relacionamentos espaciais não causa alterações diretas nas relações construídas, mas requer a implementação de controles dinâmicos (verificações *online* de consistência) ou estáticos (verificações

*offline* de consistência). Para bloquear violações de integridade espaciais *online*, controles dinâmicos são requeridos; controles estáticos são utilizados para verificação da consistência depois de uma carga de dados ou de tempos em tempos. O detalhamento da implementação dos controles dinâmicos e estáticos é apresentado na Seção 4.2.

*Passo 4: Mapeamento de generalizações.* O mapeamento de generalizações é o mesmo para classes convencionais e espaciais. É conveniente que as subclasses sejam relações distintas por motivos de gerenciamento da informação geográfica e de visualização. Dada uma superclasse  $C$  e suas subclasses  $\{S_1, S_2, \dots, S_n\}$ , converter cada generalização de acordo com as seguintes opções:

*Total/Disjunta.* Criar uma relação  $R_i$  para cada subclasse  $S_i$ ,  $1 \leq i \leq n$ , contendo todos os atributos de  $S_i$  e os atributos herdados da superclasse  $C$ , inclusive a chave primária. Criar restrição de integridade para não permitir repetição de valores de chaves primárias nas relações  $R_i$ .

*Total/Sobreposta.* Criar uma relação  $R_i$  para cada subclasse  $S_i$ ,  $1 \leq i \leq n$ , contendo todos os atributos de  $S_i$  e os atributos herdados da superclasse  $C$ , inclusive a chave primária.

*Parcial/Disjunta.* Criar uma relação  $R$  para a superclasse  $C$ , contendo todos os atributos e chave primária de  $C$ . Criar uma relação  $R_i$  para cada subclasse  $S_i$ ,  $1 \leq i \leq n$ , contendo todos os atributos de  $S_i$  e a chave primária da superclasse  $C$  no papel de chave estrangeira. Criar restrição de integridade para não permitir repetição de valores de chaves primárias nas relações  $R_i$ .

*Parcial/Sobreposta.* Criar uma relação  $R$  para a superclasse  $C$ , contendo todos os atributos e a chave primária de  $C$ . Criar uma relação  $R_i$  para cada subclasse  $S_i$ ,  $1 \leq i \leq n$ , contendo todos os atributos de  $S_i$  e a chave primária da superclasse  $C$  no papel de chave estrangeira.

*Generalização Conceitual.* *Disjunta:* O mapeamento é análogo a generalização total/disjunta. *Sobreposta:* O mapeamento é análogo a generalização total/sobreposta.

A Tabela 4.2 é uma adaptação da tabela de correspondência entre os modelos ER e relacional apresentada por Elmasri & Navathe [11], e resume o mapeamento dos construtores do modelo OMT-G.

## 4.2 Mapeamento para Esquemas Físicos

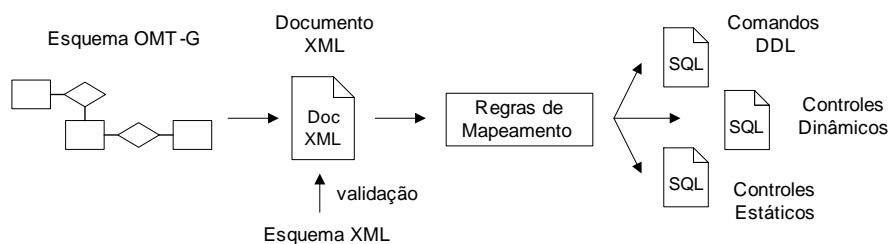
Esta seção discute a geração de esquemas físicos de banco de dados e detalha a implementação de algumas restrições de integridade espaciais.

**Tabela 4.2.** Mapeamento dos construtores do modelo OMT-G

OMT-G	Lógico
Classe convencional	Relação "Classe"
Classe espacial	Relação "Classe" com representação geográfica associada; se geocampo, restrição de integridade espacial ( $R2$ a $R5$ )
Relacionamento convencional 1:1 ou 1:N	Par chave primária-chave estrangeira
Relacionamento convencional M:N	Relação "Relacionamento" e dois pares chave primária-chave estrangeira
Relacionamento topológico	Restrição de integridade espacial relativa ao relacionamento topológico ( $RT$ )
Relacionamento em rede	Restrição de integridade espacial relativa ao relacionamento em rede ( $R6$ )
Agregação convencional	Par chave primária-chave estrangeira entre a classe <i>parte</i> e a classe <i>todo</i>
Agregação espacial	Restrição de integridade espacial relativa a agregação espacial ( $R7$ )
Generalização	Restrição de integridade entre subclasses e superclasse
Atributo simples	Atributo simples (coluna)
Atributo multivalorado	Relação "atributo multivalorado" e chave estrangeira (ver Elmasri & Navathe [11])
Atributo chave	Chave primária
Restrição espacial do usuário	Restrição de integridade espacial

### 4.2.1 Componentes Físicos

Uma ferramenta, chamada OMTG2SQL<sup>1</sup>, foi desenvolvida para automatizar a geração de esquemas físicos. A ferramenta implementa o mapeamento descrito na seção anterior e foi inicialmente preparada para gerar esquemas do Oracle Spatial. A Figura 4.2 mostra uma visão geral das etapas de mapeamento e os *scripts* gerados. A ferramenta recebe como entrada um esquema OMT-G sob o formato de um documento XML. O documento XML também pode conter descrições das restrições de integridade espaciais definidas pelo usuário. No Apêndice A são encontradas mais informações sobre o formato do documento XML.

**Figura 4.2.** Visão geral do mapeamento de esquemas OMT-G para esquemas físicos

O mapeamento de um esquema OMT-G para esquemas físicos gera três *scripts* PL/SQL: os comandos DDL, os controles dinâmicos e os controles estáticos. Os co-

<sup>1</sup>A ferramenta OMTG2SQL está disponível em <http://www.lbd.dcc.ufmg.br/lbd/tools>

*mandos DDL* contêm declarações de criação das tabelas (com suas colunas espaciais e não-espaciais), chaves primárias e estrangeiras. Também contêm declarações de *insert* na *view* de metadados (para o cadastro das colunas espaciais), de criação de índices espaciais e de criação de restrições de geometrias para classes espaciais (i.e., restrições que verificam o tipo de uma geometria). Os controles dinâmicos e estáticos são responsáveis pela implementação de restrições de integridade espaciais e não-espaciais. Os *controles dinâmicos* contêm *triggers* para verificação de consistência dos relacionamentos topológicos (i.e., restrições *RT*), das generalizações e das restrições de integridade espaciais definidas pelo usuário. Já os *controles estáticos* contêm funções para verificação *offline* da consistência dos relacionamentos em rede, das agregações espaciais e das classes do tipo geocampo (i.e., restrições *R2* a *R7*). A Tabela 4.3 resume os dados contidos em cada *script*.

Tabela 4.3. *Scripts* e seus dados

Comandos DDL	Controles Dinâmicos	Controles Estáticos
<ul style="list-style-type: none"> <li>• Criação de tabelas com colunas espaciais e não espaciais.</li> <li>• Criação de chaves primárias e estrangeiras.</li> <li>• Criação de <i>inserts</i> na <i>view</i> de metadados.</li> <li>• Criação de índices espaciais e restrições de geometrias.</li> </ul>	<ul style="list-style-type: none"> <li>• Criação de <i>triggers</i> para relacionamentos topológicos.</li> <li>• Criação de <i>triggers</i> para generalizações.</li> <li>• Criação de <i>triggers</i> para restrições de integridade espaciais definidas pelo usuário.</li> </ul>	<ul style="list-style-type: none"> <li>• Criação de funções para relacionamentos em rede.</li> <li>• Criação de funções para agregações espaciais.</li> <li>• Criação de funções para classes de geocampo.</li> </ul>

### 4.2.2 Criação das Tabelas e Índices Espaciais

Os comandos DDL podem conter declarações de criação de tabelas, de preenchimento da *view* de metadados, de criação de índices espaciais e de criação de restrições geométricas para classes espaciais.

Para cada classe espacial (com exceção de tesselação) mapeada para esquemas físicos de bancos de dados espaciais, são criados comandos SQL como apresentado na Figura 4.3. A Figura 4.3a mostra a declaração de criação de uma tabela com uma coluna espacial *geom* do tipo *SDO\_GEOMETRY*, que permite o armazenamento de geometrias como pontos, linhas e polígonos. Caso a classe espacial seja uma tesselação, a coluna espacial deve ser do tipo *SDO\_GEORASTER*. A Figura 4.3b mostra a declara-



ção para o preenchimento da *view* de metadados USER\_SDO\_GEOM\_METADATA para a coluna *geom*, onde também são apresentadas as dimensões e o sistema de referência da coluna espacial. A Figura 4.3c mostra a declaração de criação do índice espacial para a coluna *geom* com a restrição de geometria, indicando a dimensão e a geometria da coluna espacial. Nesse caso é utilizado o tipo POLYGON, indicando que *geom* deve armazenar um polígono em duas dimensões. Para pontos e linhas são utilizando os tipos POINT e LINESTRING. As classes convencionais são mapeadas de forma direta, sem a necessidade de preenchimento da *view* de metadados e de criação de índices espaciais.

<pre>CREATE TABLE Tabela_Espacial (   id NUMBER(5,0),   ...   geom MDSYS.SDO_GEOMETRY,   CONSTRAINT pk_chave PRIMARY KEY (id));</pre> <p>(a) Criação de tabela com coluna espacial</p>	<pre>INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID) VALUES ('Tabela_Espacial', 'geom',   MDSYS.SDO_DIM_ARRAY   (MDSYS.SDO_DIM_ELEMENT('X', -180, 180, 0.005),   MDSYS.SDO_DIM_ELEMENT('Y', -90, 90, 0.005)),   'SRID');</pre> <p>(b) Preenchimento da <i>view</i> de metadados</p>
<pre>CREATE INDEX SIDX_Tabela_Espacial ON Tabela_Espacial(geom) INDEXTYPE IS MDSYS.SPATIAL_INDEX PARAMETERS('SDO_IND_ DIMS=2, LAYER_GTYPE=POLYGON');</pre> <p>(c) Criação de índice espacial com restrição geométrica</p>	

**Figura 4.3.** Exemplo de comandos DDL para uma classe espacial

### 4.2.3 Implementação das Restrições de Integridade Espaciais

Os controles dinâmicos e estáticos podem conter diversas funções e *triggers* para a validação das restrições de integridades espaciais. Com o objetivo de apresentar uma visão geral da validação das restrições de integridade espaciais, esta seção discute a implementação das funções e *triggers* para a validação das agregações espaciais (*R7*), dos relacionamentos em rede (*R6*), das subdivisões planares (*R4*) e dos relacionamentos topológicos (*RT*), e mostra trechos em PL/SQL. Por questões de espaço, as demais implementações não são apresentadas, mas podem ser encontradas na página da ferramenta OMTG2SQL.

Nas Listagens 4.1 a 4.4 verificam-se a ocorrência de *tags*, que identificam os nomes das tabelas e suas respectivas chaves primárias (que são definidas nos comandos DDL), e de inserções de mensagens de erro na tabela *Spatial\_Error*, quando inconsistências nas validações são encontradas.

A Listagem 4.1 mostra a função para validação das agregações espaciais. Primeiramente cada elemento *todo* deve ser relacionado aos seus respectivos elementos *partes*, através das relações topológicas *contains*, *covers* ou *overlap*. Isso é realizado nas linhas 5-10, onde a tabela auxiliar *Sa\_Aux* recebe o *rowid* do *todo*, o *rowid* da *parte* e a geometria da parte. Após essa tabela ser carregada, ocorre a verificação de consistência da agregação espacial. O primeiro ponto da restrição de agregação espacial (i.e., (1)  $p_i \cap W = p_i$  para todo  $i$  tal que  $0 \leq i \leq n$ ) é verificado nas linhas 13-25. A idéia é verificar se a interseção de cada *parte* com o *todo* coincide com a respectiva *parte*. Note a importância da tabela *Sa\_Aux*, uma vez que cada *parte* só deve ser comparada ao seu respectivo *todo* e não com os demais. Assim, as *partes* que se sobrepõem ao *todo* são detectadas e inconsistências são capturadas na tabela de erros (linhas 21-22). O segundo ponto (i.e., (2)  $(W \cap \bigcup_{i=0}^n p_i) = W$ ) é verificado nas linhas 27-42. A idéia é verificar se a interseção do *todo* com a união das suas *partes* é igual ao respectivo *todo*. A união das *partes* é realizada através da função *join\_geometry* (29-33). Assim, espaços sem *partes* no *todo* são detectados e inconsistências são capturadas na tabela de erros (linhas 38-39). O terceiro ponto (i.e., (3)  $((p_i \text{ touch } p_j) \vee (p_i \text{ disjoint } p_j)) = \text{true}$  para todo  $i, j$  tal que  $i \neq j$ ) é verificado nas linhas 44-56, que identifica se apenas as relações topológicas *touch* e *disjoint* ocorrem entre diferentes *partes*. Caso essa condição não seja satisfeita, as inconsistências são capturadas na tabela de erros (linhas 52-53).

```

1 CREATE OR REPLACE FUNCTION val_spa_agr_<VAL_SPA_AGR_NAME>
2 RETURN VARCHAR IS
3   ...
4 BEGIN
5   FOR r IN (SELECT w.rowid as w_rowid, p.rowid as p_rowid, p.geom as p_geom
6             FROM <WHOLE_TABLE_NAME> w, <PART_TABLE_NAME> p
7             WHERE SDO_RELATE(w.geom, p.geom, 'MASK=CONTAINS+COVERS+
              OVERLAPBDYINTERSECT')='TRUE') LOOP
8     INSERT INTO sa_aux (w_rowid, p_rowid, p_geom)
9       VALUES (r.w_rowid, r.p_rowid, r.p_geom);
10  END LOOP;
11  ...
12  — Checking Point 1 of R7
13  FOR w IN (SELECT distinct w_rowid FROM sa_aux) LOOP
14    SELECT geom INTO geom_w
15      FROM <WHOLE_TABLE_NAME>
16      WHERE rowid=w.w_rowid;
17    FOR p IN (SELECT sa.p_rowid
18              FROM sa_aux sa
19              WHERE sa.w_rowid=w.w_rowid AND SDO_RELATE(sa.p_geom, SDO_GEOM
              .SDO_INTERSECTION(geom_w, sa.p_geom, 0.5), 'MASK=EQUAL') !=
              'TRUE') LOOP
20      ...
21      INSERT INTO spatial_error (type, error)
22        VALUES ('Spatial Aggregation Error', '<PART_TABLE_NAME>' || p_columns

```

```

        || ' intersection <WHOLE_TABLE_NAME> ' || w_columns || ' is not equal
        to part ');
23     p_contains_error := TRUE;
24     END LOOP;
25 END LOOP;
26 — Checking Point 2 of R7
27 FOR w IN (SELECT distinct w_rowid FROM sa_aux) LOOP
28     geom_join := NULL;
29     FOR p IN (SELECT sa.p_rowid, sa.p_geom
30                FROM sa_aux sa
31                WHERE sa.w_rowid=w.w_rowid) LOOP
32         geom_join := join_geometry(p.p_geom,geom_join);
33     END LOOP;
34     FOR w2 IN (SELECT *
35                FROM <WHOLE_TABLE_NAME>
36                WHERE rowid=w.w_rowid AND SDO_RELATE(geom, SDO_GEOM.
37                  SDO_INTERSECTION(geom,geom_join,0.5) , 'MASK=EQUAL') != '
38                  TRUE') LOOP
39         ...
40         INSERT INTO spatial_error (type, error)
41         VALUES ('Spatial Aggregation Error','<WHOLE_TABLE_NAME> ' || w_columns
42           || ' intersection all its parts is not equal to whole');
43         p_contains_error := TRUE;
44     END LOOP;
45 END LOOP;
46 — Checking Point 3 of R7
47 FOR w IN (SELECT distinct w_rowid FROM sa_aux) LOOP
48     SELECT geom INTO geom_w
49     FROM <WHOLE_TABLE_NAME>
50     WHERE rowid=w.w_rowid;
51     FOR p IN (SELECT sa1.p_rowid as p_rowid1, sa2.p_rowid as p_rowid2
52                FROM sa_aux sa1, sa_aux sa2
53                WHERE sa1.w_rowid=w.w_rowid AND sa2.w_rowid=w.w_rowid AND sa1
54                  .p_rowid!=sa2.p_rowid AND SDO_RELATE(sa1.p_geom,sa2.
55                    p_geom, 'MASK=TOUCH') != 'TRUE' AND SDO_RELATE(sa1.p_geom,
56                      sa2.p_geom, 'MASK=ANYINTERACT')='TRUE') LOOP
57         ...
58         INSERT INTO spatial_error (type, error)
59         VALUES ('Spatial Aggregation Error','Spatial relation between parts <
60           PART_TABLE_NAME> ' || p_columns || ' and ' || w_columns || ' is not touch
61           or disjoint ');
62         p_contains_error := TRUE;
63     END LOOP;
64 END LOOP;
65 DELETE FROM sa_aux; COMMIT;
66 IF (p_contains_error=TRUE) THEN
67     RETURN 'Not valid! See table Spatial_Error for more details.';
68 ELSE
69     RETURN 'Valid! No errors were found.';
70 END IF;
71 END;

```

**Listagem 4.1.** Função para validação das agregações espaciais

A Listagem 4.2 mostra a função para validação dos relacionamentos em rede arco-nó. De modo geral, é verificado se os vértices iniciais (linhas 9-24) e finais (linhas 25-40) de cada *arco* estão relacionados com exatamente um *nó*. Os erros são capturados na tabela de erros quando um *vértice* não está relacionado a nenhum *nó* (linhas 16-17 e 32-33) ou quando um *vértice* está relacionado a vários *nós* (linhas 21-22 e 37-38). Também é verificado se os *nós* estão relacionados com algum vértice inicial ou final (linhas 43-55) e, caso não estejam, os erros são capturados (linhas 51-52)

```

1 CREATE OR REPLACE FUNCTION val_network_ <VAL_NETWORK_NAME>
2 RETURN VARCHAR IS
3   ...
4 BEGIN
5   — Checking Point 2 of R8
6   FOR reg IN (SELECT rowid, geom FROM <ARC_TABLE_NAME>) LOOP
7     p_geom_initial_vertex := get_point(reg.geom);
8     p_geom_final_vertex := get_point(reg.geom, SDO_UTIL.GETNUMVERTICES(reg.
9       geom));
10    BEGIN
11      SELECT rowid INTO p_rowid_initial_vertex
12        FROM <NODE_TABLE_NAME>
13        WHERE MDSYS.SDO_EQUAL(geom, p_geom_initial_vertex)='TRUE';
14    EXCEPTION
15      WHEN NO_DATA_FOUND THEN
16      ...
17      INSERT INTO spatial_error (type, error)
18        VALUES ('Arc-Node Network Error', 'Initial vertex of arc <
19          ARC_TABLE_NAME>' || p_keys || ' is not related to any node');
20      p_contains_error := TRUE;
21    WHEN TOO_MANY_ROWS THEN
22    ...
23    INSERT INTO spatial_error (type, error)
24      VALUES ('Arc-Node Network Error', 'Initial vertex of arc <
25        ARC_TABLE_NAME>' || p_keys || ' is related to many nodes');
26    p_contains_error := TRUE;
27  END;
28  BEGIN
29    SELECT rowid INTO p_rowid_final_vertex
30      FROM <NODE_TABLE_NAME>
31      WHERE MDSYS.SDO_EQUAL(geom, p_geom_final_vertex)='TRUE';
32  EXCEPTION
33    WHEN NO_DATA_FOUND THEN
34    ...
35    INSERT INTO spatial_error (type, error)
36      VALUES ('Arc-Node Network Error', 'Final vertex of arc <
37        ARC_TABLE_NAME>' || p_keys || ' is not related to any node');
38    p_contains_error := TRUE;
39  WHEN TOO_MANY_ROWS THEN
40  ...
41  INSERT INTO spatial_error (type, error)
42    VALUES ('Arc-Node Network Error', 'Final vertex of arc <
43      ARC_TABLE_NAME>' || p_keys || ' is related to many nodes');

```

```

39     p_contains_error := TRUE;
40   END;
41 END LOOP;
42 — Checking point 1 of R8
43 FOR reg IN (SELECT rowid FROM <NODE_TABLE_NAME>) LOOP
44   BEGIN
45     SELECT a.rowid INTO p_rowid_point
46       FROM <ARC_TABLE_NAME> a, <NODE_TABLE_NAME> n
47       WHERE (MDSYS.SDO_EQUAL(n.geom, get_point(a.geom))='TRUE' OR MDSYS.
              SDO_EQUAL(n.geom, get_point(a.geom, SDO_UTIL.GETNUMVERTICES(a.geom)
              ))='TRUE') AND reg.rowid=n.rowid AND rownum <= 1;
48   EXCEPTION
49     WHEN NO_DATA_FOUND THEN
50       ...
51     INSERT INTO spatial_error (type, error)
52       VALUES ('Arc-Node Network Error', 'Node <NODE_TABLE_NAME> '||p_keys
              ||' is not related to any vertex');
53     p_contains_error := TRUE;
54   END;
55 END LOOP;
56 IF (p_contains_error=TRUE) THEN
57   RETURN 'Not valid! See table Spatial_Error for more details.';
58 ELSE
59   RETURN 'Valid! No errors were found.';
60 END IF;
61 END;

```

**Listagem 4.2.** Função para validação dos relacionamentos em rede arco-nó

A Listagem 4.3 mostra a função para validação das classes espaciais do tipo subdivisão planar. É verificado se a relação topológica entre os objetos é apenas *toca* ou *disjunto* (linhas 4-12) e, caso contrário, os erros são capturados (linhas 9-10). As funções das Listagens 4.1, 4.2 e 4.3 podem ser também implementadas como *controles dinâmicos*, utilizando *triggers*. Entretanto, a opção por implementar algumas restrições como *controles estáticos* é justificada quando as verificações consomem recursos excessivos para ser executadas durante uma transação. Selecionar entre dinâmico e estático deve ser uma decisão do modelador, dependendo da característica dos dados.

```

1 CREATE OR REPLACE FUNCTION val_pla_sub_<VAL_PLA_SUB_NAME>
2 RETURN VARCHAR IS
3   ...
4 BEGIN
5   FOR p IN (SELECT ps1.rowid as rowid1, ps2.rowid as rowid2
6             FROM <PLANAR_SUB_TABLE_NAME> ps1, <PLANAR_SUB_TABLE_NAME> ps2
7             WHERE ps1.rowid!=ps2.rowid AND SDO_RELATE(ps1.geom, ps2.geom, '
              MASK=TOUCH')!='TRUE' AND SDO_RELATE(ps1.geom, ps2.geom, 'MASK
              =ANYINTERACT')='TRUE') LOOP
8     ...
9     INSERT INTO spatial_error (type, error)
10      VALUES ('Planar Subdivision Error', 'Spatial relation between <

```

```

        PLANAR_SUB_TABLE_NAME> '||p1_columns||' and '||p2_columns||' is not
        touch or disjoint');
11     p_contains_error := TRUE;
12 END LOOP;
13 IF (p_contains_error=TRUE) THEN
14     RETURN 'Not valid! See table Spatial_Error for more details.';
15 ELSE
16     RETURN 'Valid! No errors were found.';
17 END IF;
18 ...
19 END;

```

**Listagem 4.3.** Função para validação das subdivisão planares

A Listagem 4.4 mostra a *trigger* para validação dos relacionamentos topológicos. As relações topológicas entre as duas geometrias (a armazenada e a que vai ser inserida ou atualizada) são verificadas através da função SDO\_RELATE (linhas 8-10) e, caso sejam inválidas, os erros são capturados (linha 13).

```

1 CREATE OR REPLACE TRIGGER val_top_rel_ <VAL_TOP_REL_NAME>
2   BEFORE INSERT OR UPDATE ON <B_TABLE_NAME>
3   REFERENCING NEW AS NEW OLD AS OLD
4   FOR EACH ROW
5 DECLARE
6   w_rowid rowid;
7 BEGIN
8   SELECT rowid INTO w_rowid
9     FROM <A_TABLE_NAME> w
10    WHERE SDO_RELATE(w.geom, :NEW.geom, 'MASK=<SPATIAL_RELATION_MASK>')='TRUE'
        AND rownum <= 1;
11 EXCEPTION
12   WHEN NO_DATA_FOUND THEN
13     RAISE_APPLICATION_ERROR(-20001, 'Topological relationship between <
        A_TABLE_NAME> and <B_TABLE_NAME> <B_TABLE_KEYS> is not <
        SPATIAL_RELATION>');
14 END;

```

**Listagem 4.4.** *Trigger* para validação dos relacionamentos topológicos

# Capítulo 5

## Estudo de Caso

Este capítulo apresenta um estudo de caso desenvolvido para demonstrar o mapeamento de esquemas OMT-G para esquemas GML e físicos de bancos de dados.

### 5.1 Esquema OMT-G

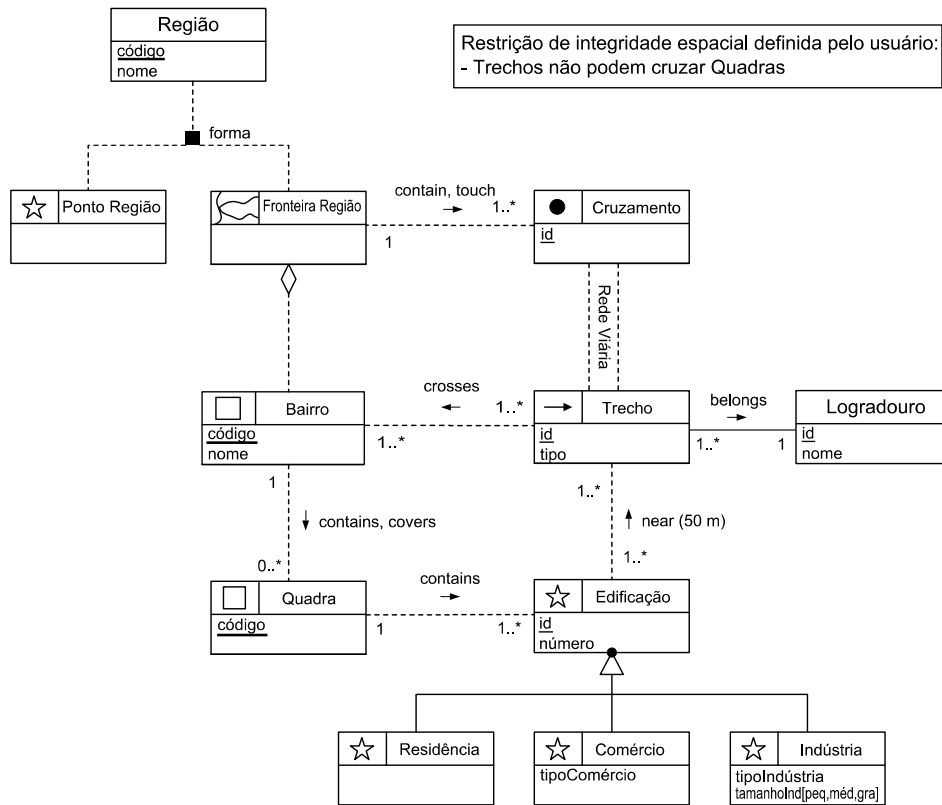
O estudo de caso considera a aplicação geográfica urbana descrita no esquema OMT-G da Figura 5.1. As classes *Ponto Região*, *Bairro*, *Quadra*, *Cruzamento*, *Trecho*, *Edificação*, *Residência*, *Comércio* e *Indústria* são representados como geo-objetos. A classe *Fronteira Região* é representada como geocampo. As classes *Região* e *Logradouro* são convencionais, i.e., não possuem representação geométrica ou localização associada. Existem diversos relacionamentos espaciais e não-espaciais. O relacionamento entre *Trecho* e *Cruzamento* forma uma rede arco-nó. Também verifica-se uma generalização total/disjunta e uma generalização conceitual sobreposta onde as classes *Edificação* e *Região* são as superclasses, respectivamente. Existe também uma restrição de integridade espacial definida pelo usuário, onde é especificado que *Trechos* não podem cruzar *Quadradas*. O esquema OMT-G foi formatado como um documento XML e fornecido como entrada para as ferramentas OMTG2GML e OMTG2SQL<sup>1</sup>.

### 5.2 Esquemas GML e Físicos

No mapeamento para esquemas GML, verifica-se que as classes *Logradouro* e *Região* são mapeadas como elementos de primeiro nível. Iniciando o mapeamento por *Logradouro*,

---

<sup>1</sup>O documento XML representando o esquema OMT-G, os esquemas GML, os esquemas físicos e as instâncias desse estudo de caso estão disponíveis em <http://www.lbd.dcc.ufmg.br/lbd/tools>



**Figura 5.1.** Esquema OMT-G para uma aplicação geográfica urbana

tem-se a estrutura do esquema GML<sup>2</sup> mostrada na Figura 5.2. *Logradouro* possui como sub-elementos os seus atributos e *Trecho* através do elemento *belongs*. *Trecho*, por sua vez, contém os seus atributos, o elemento *gml:LineString*, o elemento *crosses* referenciando *Bairro*, o elemento *near* referenciando *Edificação* e os elementos *n1* e *n2* referenciando *Cruzamento*. As referências de *crosses* e *near* ocorrem devido às suas cardinalidades M:N (ver linha 4 e coluna 4 da Tabela 3.1).

*Região* possui como sub-elementos os seus atributos, *PontoRegião* e *FronteiraRegião*. A estrutura que contém as subclasses permite a representação de *PontoRegião* ou *FronteiraRegião* ou ambos no documento GML, visto que se trata de uma generalização conceitual do tipo sobreposta. *PontoRegião* contém o sub-elemento *gml:Point*. Já *FronteiraRegião* contém os sub-elementos *gml:Polygon*, *Cruzamento* (através do elemento *contains-touch*) e *Bairro* (através do elemento *spatial-aggregation*). *Cruzamento* possui o seu atributo e o elemento *gml:Point*. *Bairro*, por sua vez, contém os seus atributos, o elemento *gml:Polygon* e *Quadra*, através do elemento *contains-covers*. *Quadra* contém os seus atributos, o elemento *gml:Polygon* e *Edificação*, através do elemento *contains*. Finalmente, *Edificação* contém os seus atributos, *Residência*, *Comércio* e *Indústria*,

<sup>2</sup>Estrutura gerada com a ferramenta Altova XMLSpy: <http://www.altova.com/xml-editor>



com seus respectivos atributos e geometrias. A estrutura que contém as subclasses permite a representação de *Residência* ou *Comércio* ou *Indústria* no documento GML, visto que se trata de uma generalização do tipo total/disjunta.

Iniciando o mapeamento por *Região*, tem-se a estrutura do esquema GML mostrada na Figura 5.3. A principal diferença entre os esquemas GML é no mapeamento dos relacionamentos *crosses* e *near*, que são codificados como sub-elementos de *Bairro* e *Edificação* ao invés de serem mapeados como sub-elementos de *Trecho*. Verifica-se que o número de restrições permanece o mesmo (duas restrições *keyref*) e os aninhamentos das estruturas permanecem similares (ambos com profundidade 9).

No mapeamento para esquema físicos, os comandos DDL contém doze declarações *create table* com suas colunas espaciais e não-espaciais, isto é, um *create table* para cada classe (exceto para *Região* e *Edificação* devido à generalização), um para a tabela *Spatial\_Error* e um para a tabela *Sa\_Aux*. Também contém dez declarações *insert into USER\_SDO\_GEOM\_METADATA* e dez *create spatial index*. As tabelas *Logradouro* e *Spatial\_Error* não precisam dos comandos *insert into* e *create spatial index*, uma vez que não possuem colunas espaciais. Também contém dois comandos *alter table* para adicionar a chave primária de *Logradouro* como chave estrangeira em *Trecho*, devido ao relacionamento convencional 1:N *belongs*.

Os controles dinâmico contém nove declarações *create trigger* para os relacionamentos topológicos, isto é, um *create trigger* para cada relacionamento topológico (note que os relacionamentos topológicos *contains* e *near* devem ser considerados como uma ligação direta para as subclasses *Residência*, *Comércio* e *Indústria*, uma vez que a superclasse *Edificação* não é materializada no esquema físico). Também contém três *create trigger* para a validação da consistência da generalização e um *create trigger* para a restrição de integridade espacial definida pelo usuário. Os controles estáticos contém três declarações *create function* para a validação *offline* da agregação espacial, da subdivisão planar e do relacionamento em rede, e dois *create function* para funções auxiliares.

## 5.3 Consultas

Como forma de comparar e validar a correspondência entre instâncias dos esquemas, os mesmos dados geográficos foram utilizados para criar um documento GML baseado no esquema GML da Figura 5.2 e para povoar um banco de dados baseado nos esquemas físicos gerados<sup>3</sup>. Assim, consultas correspondentes em SQL e XQuery foram executa-

---

<sup>3</sup>Uma estratégia para utilizar as informações do processo de modelagem para promover a geração automática de documentos GML foi concebida, e proposta para trabalho futuro. A estratégia usa a

das no SGBD Oracle Spatial e no documento GML através da ferramenta BaseX. As consultas e seus respectivos resultados são mostrados nas Figuras 5.4 a 5.7.

A Figura 5.4 mostra uma consulta para recuperar os bairros contidos em uma determinada região. Na consulta SQL, é necessário o uso do operador espacial `SDO_RELATE` para determinar o relacionamento topológico entre os objetos. Já na consulta XQuery, o relacionamento é materializado na própria estrutura do documento através do elemento *spatial-aggregation*. Note que no resultado da consulta SQL, a geometria dos bairros é mostrada na coluna *geom* através do tipo `SDO_GEOMETRY`, que nesse caso representa um polígono. Já no resultado da consulta XQuery, a geometria dos bairros é explicitada através do elemento *gml:Polygon* e seus sub-elementos.

A Figura 5.5 mostra uma consulta para recuperar os trechos de um determinado logradouro. Nesse caso, não é necessário o uso de operadores espaciais na consulta SQL, uma vez que as referências entre trechos e logradouros são realizadas através de chaves primárias e estrangeiras. Já na consulta XQuery, o relacionamento é materializado através do elemento *belongs*. No resultado da consulta XQuery, a geometria dos trechos é explicitada através do elemento *gml:LineString* e seus sub-elementos.

A Figura 5.6 mostra uma consulta para recuperar os bairros que são cruzados por um determinado trecho. Na consulta SQL é necessário o uso do operador espacial `SDO_RELATE` para determinar o relacionamento topológico entre os objetos. Já a consulta XQuery utiliza a referência *@ref* do elemento *crosses*, uma vez que o relacionamento *crosses* foi criado como referência para bairro (ver Figura 5.2).

Finalmente, a Figura 5.7 mostra uma consulta para recuperar os cruzamentos contidos em uma determinada região. Novamente, na consulta SQL é necessário o uso do operador espacial `SDO_RELATE` para determinar o relacionamento topológico entre os objetos. Já na consulta XQuery, o relacionamento é materializado através do elemento *contains-touch*. No resultado da consulta XQuery, a geometria dos cruzamentos é explicitada através do elemento *gml:Point* e seus sub-elementos.

Os resultados gerados pelas consultas em SQL e XQuery são sempre equivalentes, uma vez que os esquemas GML e físicos, nos quais foram baseadas as instâncias, refletem o mesmo esquema OMT-G. Nos documentos GML, consultas espaciais podem ainda ser realizadas através da linguagem GQL, de modo semelhante as consultas nos bancos de dados espaciais. Com isso, é possível obter dados que não são materializados nos documentos GML, tais como outras relações e análises espaciais.

---

correspondência entre as primitivas do esquema conceitual, os elementos GML e o esquema físico para criar documentos que obedeçam ao esquema GML e que contenham os dados presentes nas tabelas do banco de dados espacial, previamente povoados.

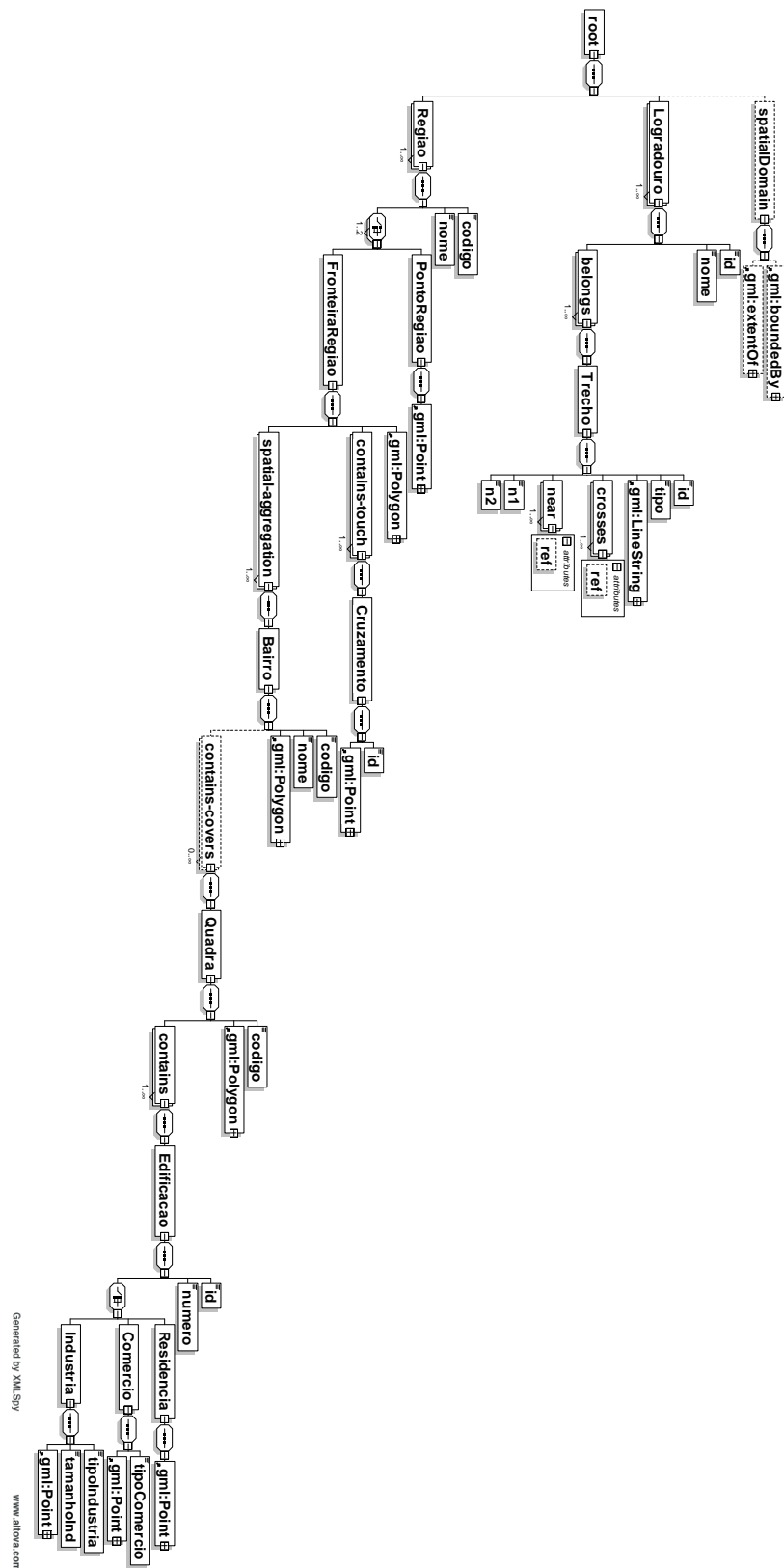


Figura 5.2. Esquema GML iniciando o mapeamento por *Logradouro*

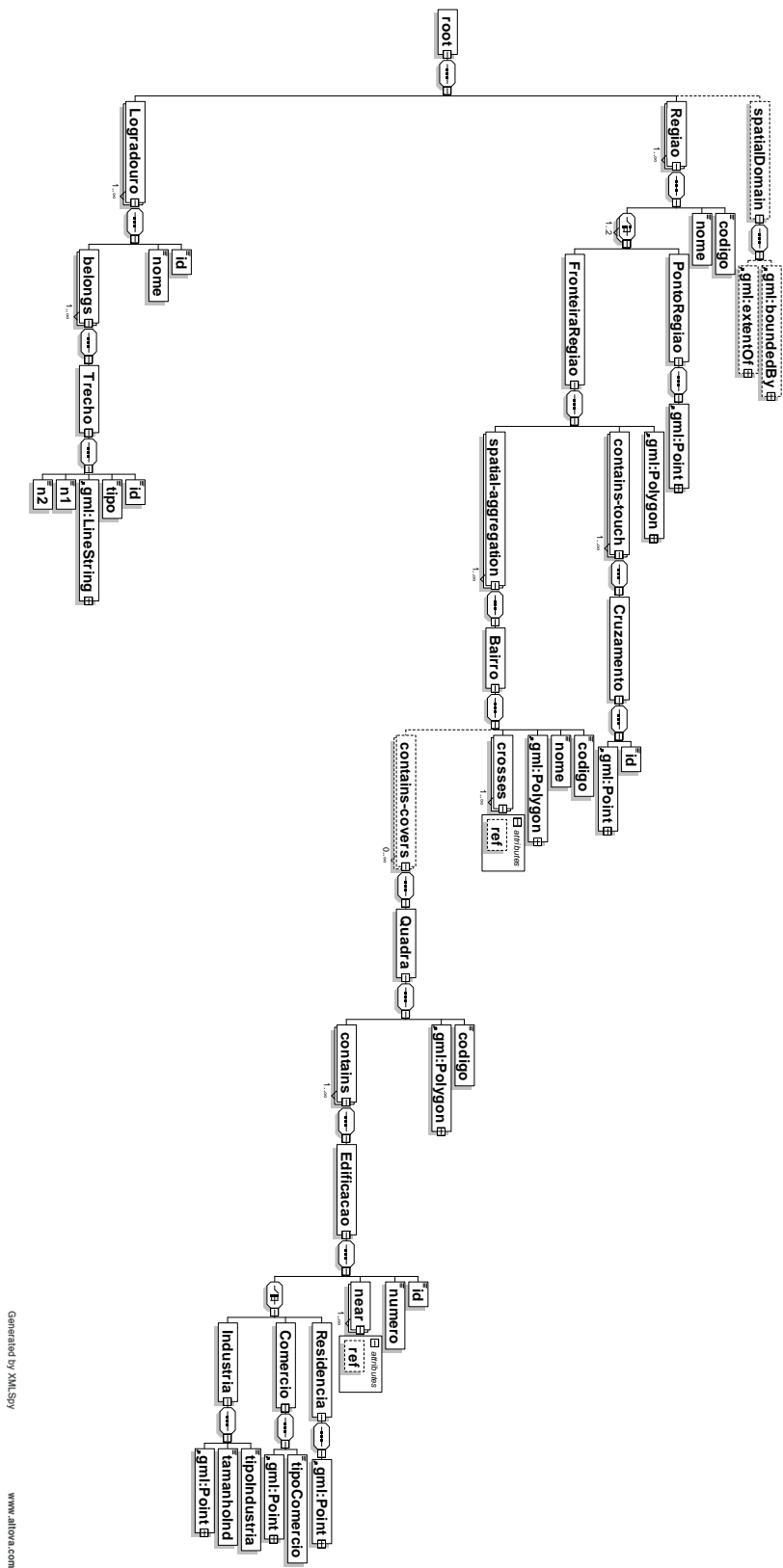


Figura 5.3. Esquema GML iniciando o mapeamento por *Região*

```
SELECT b.*
FROM FronteiraRegiao a, Bairro b
WHERE SDO_RELATE(a.geom, b.geom, 'mask=contains+covers') = 'TRUE'
and a.nome = 'G'
```

(a) consulta SQL

```
declare namespace gml="http://www.opengisnet/gml";
<bairros-regiao>{
for $r in root/Regiao[ nome="G"]/FronteiraRegiao/spatial-aggregation/Bairro
return
  <bairro>
    { $r/codigo }
    { $r/nome }
    { $r/gml:Polygon }
  </bairro>
}</bairros-regiao>
```

(b) consulta XQuery

codigo	nome	geom
15	G1	(2003,29100,Null,(1,1003,1),(11,8,11,0,16,0,16,8,11,8))
16	G2	(2003,29100,Null,(1,1003,1),(16,8,16,0,20,0,20,8,16,8))

(c) resultado consulta SQL

```
<bairros-regiao>
<bairro>
  <codigo>15</codigo>
  <nome>G1</nome>
  <gml:Polygon srsName="SDO:29100">
    <gml:outerBoundaryIs>
      <gml:LinearRing>
        <gml:coordinates decimal="." cs="," ts=" ">
          11,8 11,0 16,0 16,8 11,8
        </gml:coordinates>
      </gml:LinearRing>
    </gml:outerBoundaryIs>
  </gml:Polygon>
</bairro>
<bairro>
  <codigo>16</codigo>
  <nome>G2</nome>
  <gml:Polygon srsName="SDO:29100">
    <gml:outerBoundaryIs>
      <gml:LinearRing>
        <gml:coordinates decimal="." cs="," ts=" ">
          16,8 16,0 20,0 20,8 16,8
        </gml:coordinates>
      </gml:LinearRing>
    </gml:outerBoundaryIs>
  </gml:Polygon>
</bairro>
</bairros-regiao>
```

(d) resultado consulta XQuery

**Figura 5.4.** Consultas/resultados SQL e XQuery: recuperar os bairros contidos em uma determinada região

```
SELECT b.*
FROM logradouro a, trecho b
WHERE a.nome = 'Norte' and a.id = b.id;
```

(a) consulta SQL

```
declare namespace gml="http://www.opengisnet/gml";
<trechos-logradouro>{
for $r in root/Logradouro[ nome="Norte"]/belongsTrecho
return
  <trecho>
    { $r/id }
    { $r/tipo }
    { $r/gml:LineString }
  </trecho>
}</trechos-logradouro>
```

(b) consulta XQuery

id	tipo	geom
5	rua	(2002,29100,Null,(1,2,1),(9,5,18,5,12,18,5))
3	rua	(2002,29100,Null,(1,2,1),(3,5,18,5,9,5,18,5))
2	rua	(2002,29100,Null,(1,2,1),(0,18,5,3,5,18,5))

(c) resultado consulta SQL

```
<trechos-logradouro>
<trecho>
  <id>2</id>
  <tipo>rua</tipo>
  <gml:LineString srsName="SDO:29100">
    <gml:coordinates decimal="." cs="," ts=" ">
      0,18.5 3.5,18.5
    </gml:coordinates>
  </gml:LineString>
</trecho>
<trecho>
  <id>3</id>
  <tipo>rua</tipo>
  <gml:LineString srsName="SDO:29100">
    <gml:coordinates decimal="." cs="," ts=" ">
      3.5,18.5 9.5,18.5
    </gml:coordinates>
  </gml:LineString>
</trecho>
<trecho>
  <id>5</id>
  <tipo>rua</tipo>
  <gml:LineString srsName="SDO:29100">
    <gml:coordinates decimal="." cs="," ts=" ">
      9.5,18.5 12,18.5
    </gml:coordinates>
  </gml:LineString>
</trecho>
</trechos-logradouro>
```

(d) resultado consulta XQuery

**Figura 5.5.** Consultas/resultados SQL e XQuery: recuperar os trechos de um determinado logradouro

```
SELECT b.*
FROM Trecho a, Bairro b
WHERE SDO_RELATE(a.geom, b.geom,
'mask=overlapbyintersect+overlapbydisjoint+inside+coveredby') = 'TRUE'
and a.id = 3
```

(a) consulta SQL

```
declare namespace gml="http://www.opengis.net/gml";
<bairros-trecho>{
for $t in root/Logradouro/belongs/Trecho[id=3]/crosses,
  $r in root/Bairro[codigo=$t/@ref]
return
  <bairro>
    {$r/codigo}
    {$r/nome}
    {$r/gml:Polygon}
  </bairro>
}</bairros-trecho>
```

(b) consulta XQuery

codigo	nome	geom
1	A1	(2003,29100,Null,(1,1003,1),(0,20,0,17,7,17,7,20,0,20))
5	C1	(2003,29100,Null,(1,1003,1),(7,20,7,14,11,14,11,20,7,20))

(c) resultado consulta SQL

```
<bairros-trecho>
<bairro>
  <codigo>1</codigo>
  <nome>A1</nome>
  <gml:Polygon srsName="SDO:29100">
    <gml:outerBoundaryIs>
      <gml:LinearRing>
        <gml:coordinates decimal="." cs="," ts=" ">
          0,20 0,17 7,17 7,20 0,20
        </gml:coordinates>
      </gml:LinearRing>
    </gml:outerBoundaryIs>
  </gml:Polygon>
</bairro>
<bairro>
  <codigo>5</codigo>
  <nome>C1</nome>
  <gml:Polygon srsName="SDO:29100">
    <gml:outerBoundaryIs>
      <gml:LinearRing>
        <gml:coordinates decimal="." cs="," ts=" ">
          7,20 7,14 11,14 11,20 7,20
        </gml:coordinates>
      </gml:LinearRing>
    </gml:outerBoundaryIs>
  </gml:Polygon>
</bairro>
</bairros-trecho>
```

(d) resultado consulta XQuery

**Figura 5.6.** Consultas/resultados SQL e XQuery: recuperar os bairros que são cruzados por um determinado trecho

```
SELECT b.*
FROM FronteiraRegiao a, Cruzamento b
WHERE SDO_RELATE(a.geom, b.geom,
'mask=contains+touch') = 'TRUE' and a.nome = 'A'
```

(a) consulta SQL

```
declare namespace gml="http://www.opengis.net/gml";
<cruzamentos-regiao>{
for $c in root/Regiao[nome="A"]/FronteiraRegiao/contains-touch/Cruzamento
return
  $c
}</cruzamentos-regiao>
```

(b) consulta XQuery

id	geom
1	(2001,29100,(3,5,20,Null),Null,Null)
2	(2001,29100,(0,18,5,Null),Null,Null)
3	(2001,29100,(3,5,18,5,Null),Null,Null)

(c) resultado consulta SQL

```
<cruzamentos-regiao>
<cruzamento>
  <id>1</id>
  <gml:Point srsName="SDO:29100">
    <gml:coordinates decimal="." cs="," ts=" ">
      3,5,20
    </gml:coordinates>
  </gml:Point>
</cruzamento>
<cruzamento>
  <id>2</id>
  <gml:Point srsName="SDO:29100">
    <gml:coordinates decimal="." cs="," ts=" ">
      0,18,5
    </gml:coordinates>
  </gml:Point>
</cruzamento>
<cruzamento>
  <id>3</id>
  <gml:Point srsName="SDO:29100">
    <gml:coordinates decimal="." cs="," ts=" ">
      3,5,18,5
    </gml:coordinates>
  </gml:Point>
</cruzamento>
</cruzamentos-regiao>
```

(d) resultado consulta XQuery

**Figura 5.7.** Consultas/resultados SQL e XQuery: recuperar os cruzamentos contidos em uma determinada região

## Capítulo 6

# Conclusões e Trabalhos Futuros

Foram definidos mapeamentos de esquemas OMT-G em esquemas GML e esquemas físicos de bancos de dados espaciais. Tais mapeamentos ainda não tinham sido desenvolvidos nem constavam na literatura, uma vez que o modelo OMT-G foi apenas formalizado. Nesse contexto, o trabalho em questão colabora para uma maior utilização do modelo OMT-G na prática. Considerando o mapeamento de esquemas conceituais geográficos para esquemas GML, apenas um trabalho foi encontrado na literatura, mas com poucas informações sobre suas transformações.

Para a transformação em esquemas GML foram definidas regras e um algoritmo de mapeamento. As regras de mapeamento detalham a transformação das primitivas do OMT-G, tais como classes e relacionamentos, em elementos na GML. O mapeamento dos relacionamentos do OMT-G é realizado através de uma estratégia para reduzir o uso de restrições e aumentar o aninhamento dos elementos no esquema GML. A redução do uso de restrições é importante para diminuir o tempo de validação de documentos GML e o aninhamento dos elementos é importante para aumentar a eficiência de consultas a documentos GML, quando linguagens como XQuery e XPath são utilizadas. O algoritmo de mapeamento baseia-se nas regras propostas e consiste de duas etapas: a determinação dos elementos de primeiro nível e a geração do esquema GML. Seu tempo de execução é  $\Theta(V + E)$ , onde  $V$  são as classes e  $E$  são os relacionamentos no esquema OMT-G.

Nos exemplos estudados, foi verificado que a seleção correta dos elementos raiz impacta na geração de esquemas mais aninhados e com menos restrições *keyref*, confirmando a importância dos elementos de primeiro nível para a geração dos esquemas. Esquemas OMT-G mapeados começando por classes que não são elementos de primeiro nível possuem em média 55,55% mais elementos raiz e 46,42% mais restrições *keyref*. Esquemas mapeados começando por classes que são elementos de primeiro nível são

em média 50% mais profundos.

O mapeamento proposto e outros relacionados foram utilizados para a conversão de um esquema OMT-G em esquemas GML, visando uma comparação das técnicas de transformação e dos tempos de processamento de consultas espaciais e não-espaciais em documentos GML. A abordagem que gerou o esquema *Relacionado-1* não trata originalmente primitivas geográficas e a abordagem que gerou o esquema *Relacionado-2*, a única encontrada na literatura que trata da geração de esquemas GML, apresentou algumas inconsistências. Os esquemas aninhados (*O2G* e *Relacionado-1*) apresentaram resultados mais eficientes de tempo de processamento de consultas que o esquema *flat*, visto que nessa é necessário o uso frequente de junções. O esquema *O2G* obteve melhores resultados que o esquema *Relacionado-1*, uma vez que nessa é necessário testar a existência dos relacionamentos, enquanto que em *O2G* o relacionamento faz parte da própria estrutura aninhada.

Para a transformação em esquemas de bancos de dados espaciais foram definidas as etapas de mapeamento lógico e físico, de modo a tratar as primitivas do OMT-G e suas restrições de integridade espaciais. O mapeamento lógico estende regras de Elmasri & Navathe [11] para o tratamento das primitivas geográficas. O mapeamento físico gera três *scripts* PL/SQL. O primeiro script, os *comandos DDL*, contém declarações de criação das tabelas, chaves primárias e estrangeiras, declarações de *insert* na *view* de metadados, de criação de índices espaciais e de criação de restrições geométricas para classes espaciais. O segundo, os *controles dinâmicos*, contém *triggers* para verificação de consistência dos relacionamentos topológicos, das generalizações e das restrições de integridade espaciais definidas pelo usuário. O terceiro, os *controles estáticos*, contém funções para verificação da consistência dos relacionamentos em rede, das agregações espaciais e das classes do tipo geocampo.

Duas ferramentas foram desenvolvidas, OMTG2GML e OMTG2SQL, visando automatizar a geração dos esquemas GML e físicos a partir do esquema OMT-G.

Como um primeiro trabalho futuro, está a implementação de uma ferramenta gráfica para a modelagem do OMT-G. A ferramenta poderá ser utilizada em conjunto com as aplicações desenvolvidas neste trabalho, automatizando a geração de esquemas GML e físicos de bancos de dados espaciais. Como as ferramentas OMTG2GML e OMTG2SQL recebem como entrada um esquema OMT-G no formato de um documento XML, basta que a ferramenta gere os esquemas OMT-G nesse formato para que a integração ocorra de forma trivial.

Um outro trabalho é uma abordagem diferente para a geração dos esquemas GML. Baseado no trabalho de Schroeder & Mello [40, 41] (que tratam da geração de esquemas XML), pode-se gerar esquemas GML levando em consideração a quantidade



de elementos geográficos e não-geográficos existentes no documento GML e a carga de operações que serão realizadas nas consultas aos documentos GML. Isso é possível caso exista um conhecimento prévio sobre a quantidade estimada de instâncias de classes e sobre quais operações serão realizadas com maior frequência. Documentos GML criados com essa abordagem terão tempo de processamento de consultas mais eficientes para as consultas que são realizadas com mais frequência.

Outro trabalho é a geração de esquemas físicos para os bancos de dados espaciais PostgreSQL (PostGIS) e MySQL (MySQL Spatial Extensions). Assim como neste trabalho foi utilizada a linguagem PL/SQL da Oracle, os outros esquemas deverão utilizar suas linguagens específicas, PL/pgSQL para o PostgreSQL, e SQL/PSM para o MySQL. Como PL/SQL, PL/pgSQL e SQL/PSM são linguagens procedimentais com a finalidade de realizar consultas em bancos de dados, suas estruturas básicas variam pouco. A linguagem PL/pgSQL é semelhante à linguagem PL/SQL em muitos aspectos<sup>1</sup>, facilitando assim geração de esquemas para o PostgreSQL.

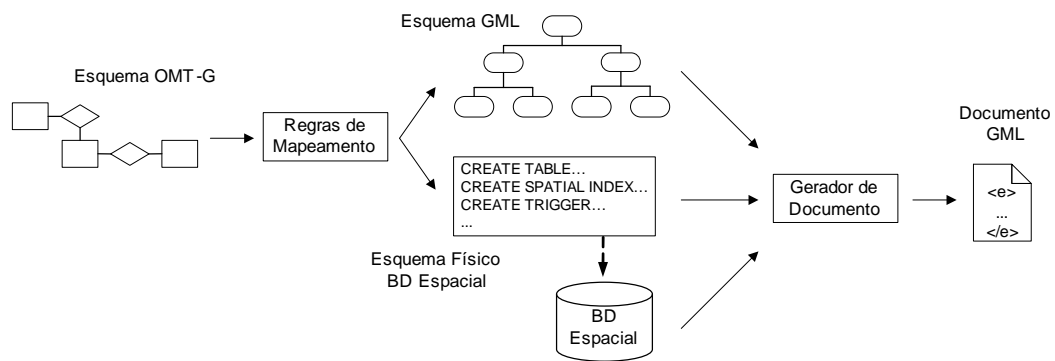
Finalmente, tem-se a implementação de uma abordagem para a geração de documentos GML. Grande parte das informações geográficas após modeladas são armazenadas em bancos de dados espaciais [39]. Assim, converter o conteúdo dos bancos de dados espaciais para documentos GML é uma atividade importante para a troca de informação geográfica. SGBDs como o Oracle Spatial e PostGIS suportam o mapeamento de suas informações para documentos XML e GML. No entanto, geralmente, essas informações são simplesmente exportadas [37], não levando em conta os relacionamentos entre os objetos espaciais, resultando em documentos GML *flats* e que não são baseados em esquemas GML.

Com os mapeamentos propostos, documentos GML podem ser criados baseados nos esquemas GML, nos esquemas físicos e na informação armazenada no banco de dados espacial [20], caso não forem criados diretamente a partir do esquema GML. A idéia principal é a materialização dos relacionamentos espaciais nos documentos GML, utilizando o esquema GML para a interpretação da informação geográfica armazenada. A Figura 6.1 ilustra as etapas envolvidas nesse processo. Primeiramente, o esquema OMT-G é convertido nos esquemas GML e físico. Após, os dados geográficos são carregados no banco de dados espacial, baseado no esquema físico gerado. Finalmente, o documento GML é gerado baseado nos esquemas GML e físico e na informação armazenada no banco de dados espacial.

Conforme citado no Capítulo 3, nos bancos de dados espaciais as relações topológicas entre os objetos são verificadas através de consultas com funções topológicas.

---

<sup>1</sup><http://www.pgdocptbr.sourceforge.net/pg80/plpgsql-porting.html>



**Figura 6.1.** Etapas para a geração de documentos GML

Linguagens padrão de consulta em XML não suportam o uso de tais funções topológicas. No entanto, uma vez que o relacionamentos espaciais são materializados no momento da criação do documento GML, linguagens de consulta como XPath e XQuery podem ser utilizadas para consultar não só relacionamentos topológicos, mas também os demais relacionamentos espaciais presentes no esquema OMT-G. A geração de documentos GML baseados em dados reais armazenados nos bancos de dados espaciais é algo ainda pouco explorado e encontrado na literatura e os mapeamentos propostos nesta dissertação fornece as bases que isso seja possível.

# Referências Bibliográficas

- [1] Al-Kamha, R.; Embley, D. W. & Liddle, S. W. (2005). Representing Generalization/Specialization in XML Schema. In *Proceedings of Enterprise Modelling and Information Systems Architectures*, volume 75, pp. 93–104.
- [2] Amano, S.; Libkin, L. & Murlak, F. (2009). XML Schema Mappings. In *Proceedings of Symposium on Principles of Database Systems*, pp. 33–42.
- [3] Barbosa, D.; Mendelzon, A. O.; Keenleyside, J. & Lyons, K. A. (2002). ToXgene: a template-based data generator for XML. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 49–54.
- [4] Borges, K. A. V.; Davis Jr., C. A. & Laender, A. H. F. (2001). OMT-G: An Object-Oriented Data Model for Geographic Applications. *Geoinformatica*, 5(3):221–260.
- [5] Borges, K. A. V.; Davis Jr., C. A. & Laender, A. H. F. (2002). Integrity Constraints in Spatial Databases. In *Proceeding of Database Integrity: Challenges and Solutions*, pp. 144–171.
- [6] Borges, K. A. V.; Davis Jr., C. A. & Laender, A. H. F. (2005). Modelagem Conceitual de Dados Geográficos. In Casanova, M.; Câmara, G.; Davis Jr., C. A.; Vinhas, L. & Ribeiro, G., editores, *Bancos de Dados Geográficos*, pp. 83–136. MundoGeo Editora, Curitiba.
- [7] Boucelma, O. & Colonna, F.-M. (2004). Querying GML data with GQuery. Technical report, University of Provence.
- [8] Davis Jr., C. A. & Hora, A. C. (2010). XML em Aplicações Geoespaciais: GML. In Mirella M. Moro, Vanessa Braganholo, C. H., editor, *Gestão de Dados na Web e XML*, pp. –. (no prelo), –.
- [9] Egenhofer, M. (1993). Point-Set Topological Spatial Relations. *Journal of Geographic Information Systems*, 47(3&4):261–273.

- [10] Egenhofer, M. & Franzosa, R. (1991). Point-Set Topological Spatial Relations. *Geographical Information Science*, 5(2):161–174.
- [11] Elmasri, R. & Navathe, S. B. (2006). *Fundamentals of Database Systems (5th Edition)*. Addison Wesley.
- [12] Franceschet, M.; Gubiani, D.; Montanari, A. & Piazza, C. (2009). From Entity Relationship to XML Schema: A Graph-Theoretic Approach. In *Proceedings of XML Database Symposium*, pp. 165–179.
- [13] Franceschet, M.; Montanari, A. & Gubiani, D. (2007). Modeling and Validating Spatio-Temporal Conceptual Schemas in XML Schema. In *Proceedings of Conference on Database and Expert Systems Applications*, pp. 25–29.
- [14] Grun, C.; Gath, S.; Holupirek, A. & Scholl, M. H. (2009). XQuery Full Text Implementation in BaseX. In *Proceedings of XML Database Symposium*, pp. 114–128.
- [15] Güting, R. H. (1994). An Introduction to Spatial Database Systems. *VLDB Journal*, 3:357–399.
- [16] Guan, J.; Zhu, F.; Zhou, J. & Niu, L. (2006). GQL: Extending XQuery to Query GML Documents. In *Proceedings of Geo-Spatial Information Science*, volume 9, pp. 118–126.
- [17] Gubiani, D. (2007). *ChronoGeoGraph: a Development Framework for Spatio-Temporal Databases*. PhD thesis, Università degli Studi G. d'Annunzio di Chieti-Pescara.
- [18] Hadzilacos, T. & Tryfona, N. (1997). An Extended Entity-relationship Model for Geographic Applications. *SIGMOD Record*, 26(3):24–29.
- [19] Hora, A. C.; Davis Jr., C. A. & Moro, M. M. (2010a). Generating XML/GML Schemas from Geographic Conceptual Schemas. In *Proceedings of Alberto Mendelzon International Workshop on Foundations of Data Management*.
- [20] Hora, A. C.; Davis Jr., C. A. & Moro, M. M. (2010b). Mapeamento de Relacionamentos em Rede Armazenados em Bancos de Dados Espaciais para Documentos GML. In *Proceedings of Brazilian Symposium on Databases (Short Papers)*.
- [21] Kleiner, C. & Lipeck, U. W. (2001). Automatic Generation of XML DTDs from Conceptual Database Schemas. In *Proceedings of Wirtschaft und Wissenschaft in*

- der Network Economy Tagungsband der GI/OCG Jahrestagung - Informatik*, pp. 396–405.
- [22] Kusters, G.; Pagel, B.-U. & Six, H.-W. (1997). GIS-application development with GEOOO. *Geographical Information Science*, 11(4):307–335.
- [23] Kothuri, R. V.; Godfrind, A. & Beinat, E. (2007). *Pro Oracle Spatial for Oracle Database 11g*. Apress.
- [24] Lake, R.; Burggraf, D.; Trninic, M. & Rae, L. (2004). *Geography Mark-Up Language: Foundation for the Geo-Web*. John Wiley & Sons.
- [25] Lbath, A. & Pinet, F. (2000). The Development and Customization of GIS-Based Applications and Web-Based GIS Applications with the CASE Tool AIGLE. In *Proceedings of Symposium on Advances in Geographic Information Systems*, pp. 194–196.
- [26] Lisboa, J. & Iochpe, C. (1999). Specifying analysis patterns for geographic databases on the basis of a conceptual framework. In *Proceedings of ACM Symposium on Advances in Geographic Information Systems*, pp. 7–13.
- [27] Lisboa, J.; Júnior, M. F. R. & Daltio, J. (2004). ArgoCASEGEO - Uma Ferramenta CASE de Código-aberto para o Modelo UML-GeoFrame. In *Proceedings of Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software*, pp. 103–113.
- [28] Liu, C. & Li, J. (2006). Designing Quality XML Schemas from E-R Diagrams. In *Proceedings of Conference on Web-Age Information*, pp. 508–519.
- [29] Lósio, B. F.; Salgado, A. C. & do Rêgo Galvão, L. (2003). Conceptual Modeling of XML Schemas. In *Proceedings of Workshop on Web Information and Data Management*, pp. 102–105.
- [30] Minout, M.; Parent, C. & Zimányi, E. (2004). A Tool for Transforming Conceptual Schemas of Spatio-Temporal Databases with Multiple Representation. In *Proceedings of Conference on Databases and Applications*, pp. 1–6.
- [31] Mok, W. Y. & Embley, D. W. (2006). Generating Compact Redundancy-Free XML Documents from Conceptual-Model Hypergraphs. *IEEE Trans. on Knowl. and Data Eng.*, 18(8):1082–1096.

- [32] Necasky, M. (2006). Conceptual Modeling for XML: A Survey. In *Proceedings of Workshop on Databases, Texts, Specifications and Objects*, volume 176, pp. 40–53.
- [33] Oliveira, J. L. D.; Pires, F. & Medeiros, C. B. (1997). An Environment for Modeling and Design of Geographic Applications. *Geoinformatica*, 1(1):29–58.
- [34] Open Geospatial Consortium (2010). Geography Markup Language (GML). <http://www.opengeospatial.org/standards/gml>. Último acesso em Novembro de 2010.
- [35] ORM (2010). OGC Reference Model. <http://www.opengeospatial.org/standards/orm>. Último acesso em Novembro de 2010.
- [36] Parent, C.; Spaccapietra, S. & Zimányi, E. (2006). *Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach*. Springer.
- [37] Park, S.-Y.; Lee, J.-D. & Bae, H.-Y. (2003). Easily Accessible GML-based Geographic Information System for Multiple Data Server over the Web. In *Proceedings of Information Systems Technology and its Applications*.
- [38] Pigozzo, P. & Quintarelli, E. (2005). An Algorithm for Generating XML Schemas from ER Schemas. In *Proceedings of Italian Symposium on Database Systems*, pp. 192–199.
- [39] Queiroz, G. R. & Ferreira, K. R. (2005). SGBD com extensões espaciais. In Casanova, M.; Câmara, G.; Davis Jr., C. A.; Vinhas, L. & Ribeiro, G., editores, *Bancos de Dados Geográficos*, pp. 267–303. MundoGeo Editora, Curitiba.
- [40] Schroeder, R. & Mello, R. d. S. (2008). Improving query performance on XML documents: a workload-driven design approach. In *Proceeding of Symposium on Document Engineering*, pp. 177–186.
- [41] Schroeder, R. & Mello, R. D. S. (2009). Designing XML Documents from Conceptual Schemas and Workload Information. *Multimedia Tools Appl*, 43(3):303–326.
- [42] Vatsavai, R. R. (2002). GML-QL: A Spatial Query Language Specification for GML. Technical report, University of Minnesota.
- [43] Worboys, M. F.; Hearnshaw, H. M. & Maguire, D. J. (1990). Object-oriented Data Modelling for Spatial Databases. *Geographical Information Science*, 4(4):369–383.
- [44] World Wide Web Consortium (2010a). Extensible Markup Language (XML). <http://www.w3.org/TR/xml/>. Último acesso em Novembro de 2010.

- [45] World Wide Web Consortium (2010b). XML Path Language (XPath). <http://www.w3.org/TR/xpath>. Último acesso em Novembro de 2010.
- [46] World Wide Web Consortium (2010c). Xml schema part 1: Structures second edition. <http://www.w3.org/TR/xmlschema-1>. Último acesso em Novembro de 2010.
- [47] World Wide Web Consortium (2010d). XQuery 1.0: An XML Query Language (XQuery). <http://www.w3.org/TR/xquery/>. Último acesso em Novembro de 2010.
- [48] Zhu, F.; Guan, J.; Zhou, J. & Zhou, S. (2006). Storing and Querying GML in Object-Relational Databases. In *Proceedings of Symposium on Advances in Geographic Information Systems*, pp. 107–114. ACM.
- [49] Zimányi, E. & Minout, M. (2006). Implementing Conceptual Spatio-temporal Schemas in Object-Relational DBMSs. In *Proceedings of Workshop on Semantic-Based Geographical Information Systems*, pp. 1648–1657.

## Apêndice A

# Ferramentas e Representação de Esquemas OMT-G

As ferramentas OMTG2GML e OMTG2SQL foram desenvolvidas em Java e recebem como entrada um esquema OMT-G sob o formato de um documento XML. O formato XML de entrada possibilita que outras aplicações ou ferramentas gráficas, mesmo desenvolvidas em outras linguagens, possam se comunicar com OMTG2GML e OMTG2SQL. O documento XML é descrito por um esquema XML especificado na linguagem XML Schema. As ferramentas verificam se os documento XML são bem-formados e se são válidos em relação ao esquema XML. A Figura A.1 mostra a estrutura do esquema XML. A versão completa na linguagem XML Schema está disponível em <http://www.lbd.dcc.ufmg.br/lbd/tools>.

Verifica-se que o esquema XML permite a representação todas as primitivas do OMT-G, tais como classes, atributos, relacionamentos e generalizações. As classes possuem um *nome*, um *tipo* e um conjunto de *atributos*. Os atributos possuem um *nome*, um *tipo* e diversas restrições opcionais relacionadas a *chave primária*, valores *default*, *tamanho*, *domínio*, entre outros. Para cada tipo de relacionamento e generalização uma estrutura deve ser seguida, conforme pode visto no esquema XML. Para a geração de esquemas físicos pode-se também representar as restrições de integridade espaciais definidas pelo usuário. Alguns elementos do esquema XML possuem restrição de valores, tais como o *tipo* das classes, o *tipo* das generalizações e as *relações espaciais* dos relacionamento topológicos.



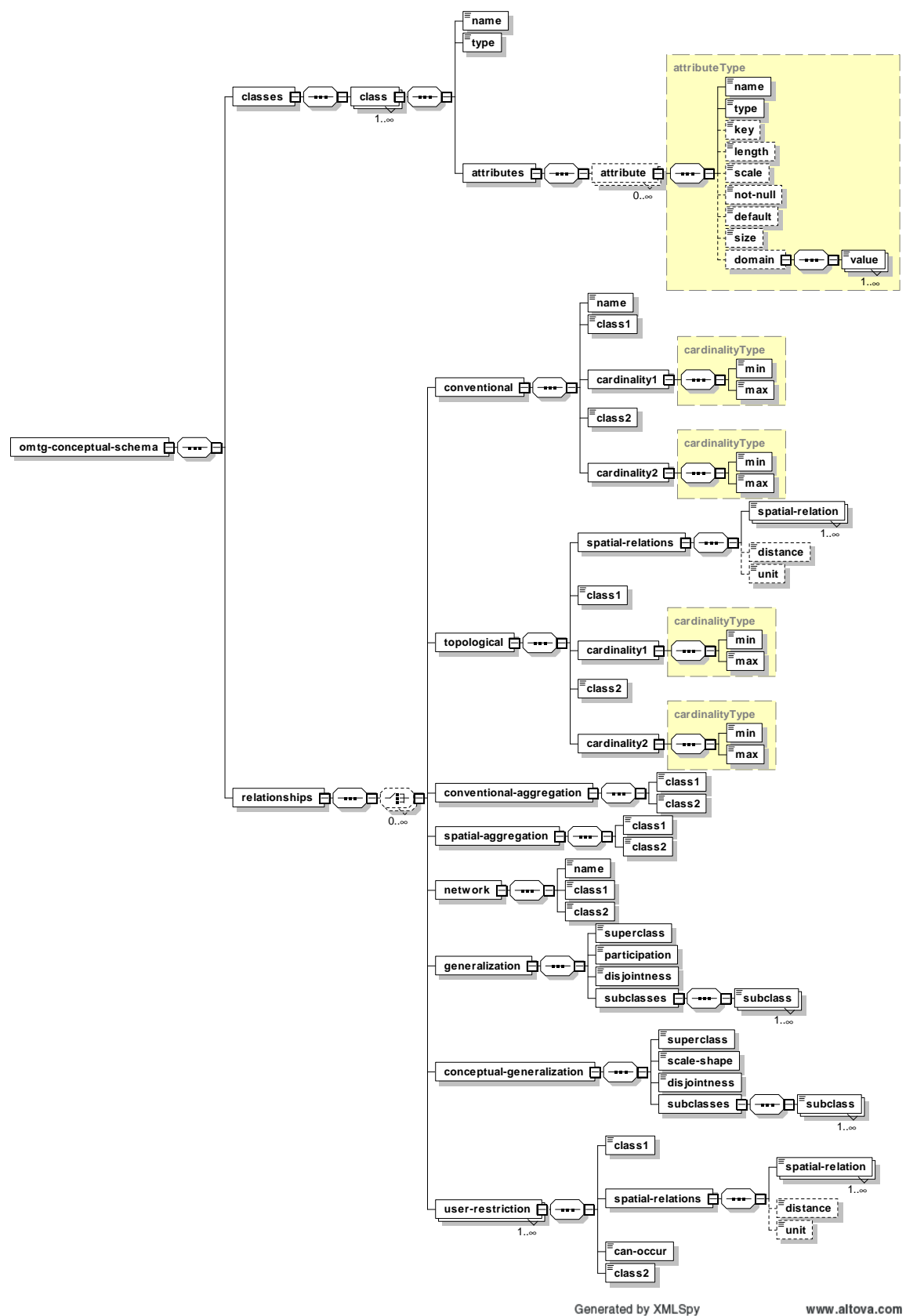


Figura A.1. Esquema XML para representação de esquemas OMT-G