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

Estendendo GeoDWCASE para Oracle Spatial e MySQL

TRABALHO DE GRADUAÇÃO

Paulo Roberto de Melo Rodrigues

RECIFE, JANEIRO DE 2008

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

Paulo Roberto de Melo Rodrigues

Estendendo GeoDWCASE para Oracle Spatial e MySQL

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Robson do Nascimento Fidalgo

RECIFE, JANEIRO DE 2008

À minha família e amigos com os quais sempre pude contar em todos os momentos

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por todas as oportunidades oferecidas ao longo da minha vida.

Ao meu orientador Robson Fidalgo, por toda força e disponibilidade dispensadas, tornando possível a concretização deste trabalho.

À Rafael Fonseca, pois esse trabalho nada mais é do que uma extensão de sua dissertação de mestrado.

À minha família, por todo o apoio incondicional oferecido durante todas as etapas da minha vida.

Ao meus amigos, por todas as palavras de apoio e os momentos de descontração durante todo o curso de graduação.

RESUMO

O grande desafio das corporações é gerenciar um grande volume de dados convencionais e geográficos, de forma que, estas informações tornem-se um diferencial competitivo. Sistemas de suporte à decisão dão apoio aos líderes de uma organização fornecendo dados de mais alto nível para decisões complexas e importantes. Data Warehouse (DW) é uma das tecnologias que melhor provê suporte ao processo de tomada de decisão. O acréscimo de dados geográficos nas base de dados das organizações modernas, fez com que surgisse a idéia de construção de um DWG (Data Warehouse Geográfico), que tem como objetivo gerenciar, eficientemente, grandes quantidades de dados históricos que também incluem dados espaciais. Não existe consenso sobre como integrar os dados convencionais com os geográficos de forma mais adequada. Neste contexto, Fonseca et. al. [Fonseca et. al. 2007a] proporam o metamodelo GeoDWM (Geographical DW Metamodel) que define como organizar e relacionar medidas, dimensões e tipos geográficos para se obter um esquema DWG sem inconsistências sintáticas. Por sua vez, Fonseca et. al. [Fonseca et. al. 2007b] proporam a ferramenta case GeoDWCASE (Geographical DW CASE) a qual oferece uma IDE gráfica, auxilia na construção e validação do esquema conceitual DWG e também gera automaticamente o esquema lógico do banco de dados. O objetivo deste trabalho é estender GeoDWCASE para que gere automaticamente o modelo lógico para o Oracle Spatial e MySQL.

Palavras-chave: Banco de Dados Geográfico, Data Warehouse Geográfico, Ferramentas CASE, Plugins.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 Introdução

Este capítulo descreve, de forma sucinta, o contexto, os objetivos e motivações deste trabalho de graduação. No final, é apresentada a organização dos capítulos.

1.1 Apresentação

A constante diminuição dos custos de armazenamento de dados e a popularização de outros meios de obtenção desses dados (por exemplo, imagens de satélite), que antes eram acessíveis a poucos, criaram um novo ambiente com novos desafios para as corporações modernas. O grande desafio das corporações é gerenciar um grande volume de dados convencionais e geográficos, de forma que, estas informações tornem-se um diferencial competitivo.

Sistemas de suporte à decisão dão apoio aos líderes de uma organização fornecendo dados de mais alto nível para decisões complexas e importantes. Data Warehouse (DW) é uma das tecnologias que melhor provê suporte ao processo de tomada de decisão. A construção de Data Warehouses (depósitos de dados) só foi possível graças ao crescente poder de processamento e sofisticação das ferramentas e técnicas analíticas. DW tem como característica principal o processamento multidimensional.

Os dados geográficos estão cada vez mais presentes nas organizações modernas. Os DW não possuíam a capacidade de armazenar ou manipular dados espaciais. Surge então a idéia de DWG (Data Warehouse Geográfico), que tem como objetivo gerenciar, eficientemente, grandes quantidades de dados históricos que também incluem dados espaciais. Não existe consenso sobre como integrar os dados convencionais com os geográficos de forma mais adequada. No entanto, é comum a construção de um DWG (Data Warehouse Geográfico) onde os dados convencionais e geográficos são integrados em uma única base de dados.

Neste contexto, Fonseca et. al. [Fonseca et. al. 2007a] proporam o metamodelo GeoDWM (Geographical DW Metamodel) que define como organizar e relacionar medidas, dimensões e tipos geográficos para se obter um esquema DWG sem inconsistências sintáticas. Por sua vez, Fonseca et. al. [Fonseca et. al. 2007b] proporam a ferramenta case GeoDWCASE (Geographical DW CASE) a qual oferece uma IDE gráfica, auxilia na construção e validação do esquema conceitual DWG e também gera automaticamente o esquema lógico do banco de dados.

1.2 Objetivos

O objetivo deste trabalho é estender a ferramenta GeoDWCASE para que gere automaticamente o modelo lógico para o Oracle Spatial [Oracle Spatial 2008] e MySQL [MySQL 2008]. GeoDWCASE, atualmente, dá suporte a geração automática do modelo lógico para o SGBD PostgresSQL [PostgresSQL 2008] com a extensão PostGIS [PostGIS 2008]. A partir da experiência adquirida com a implementação dos plugins, que darão as novas funcionadades a ferramenta GeoDWCASE, este trabalho tem também como objetivo indicar melhorias para a ferramenta CASE.

Com isso, este trabalho visa fazer uma contribuição para o projeto GOLAPWare (Geographical On-line Analytical Processing Software) [GOLAPWare 2008] que propõe uma solução para integrar processamento analítico e geográfico em uma arquitetura baseada no uso de DWG.

1.3 Estrutura do Trabalho de Graduação

Os demais capítulos deste trabalho estão organizados da seguinte forma: o capítulo 2 aborda os principais conceitos de banco de dados geograficos, apresenta também as extensões espaciais dos bancos de dados Oracle, PostgreSQL e MySQL. O capítulo 3 apresenta uma visão geral sobre Data Warehouse Geográfico e os principais componentes do projeto GOLAPWare: a arquitetura GOLAPA, GeoDWFrame, o metamodelo GeoDWM e a ferramenta case GeoDWCASE. O capítulo 4 mostra como foi feita e as principais dificuldades encontradas na implementação dos plugins que farão com que GeoDWCASE gere código automaticamente para o Oracle Spatial e MySQL. O capítulo 5 mostra os estudos de casos realizados que exemplificam o trabalho realizado. Por fim, o capítulo 6 aborda as conclusões deste trabalho, juntamente com suas principais contribuições e indicações sobre trabalhos futuros.

2 Banco de Dados Geográficos

Este capítulo tem como objetivo mostrar as principais características dos bancos de dados geográficos, que são de fundamental importância para a construção de um Data Warehouse Geográfico. Além disso, são apresentados os principais sistemas de banco de dados que suportam dados geográficos e suas características convencionais e geográficas.

2.1 Introdução

Um BDG (Banco de Dados Geográficos) estende as funcionalidades de um SGBD convencional para fornecer o armazenamento, a manipulação e a consulta eficiente de dados geográficos [Rigaux et al. 2002]. Desta forma, o BDG gerencia os dados geográficos, incluindo a utilização de tipos de dados espaciais, métodos de acesso espacial eficiente e linguagens de consulta espacial. Estes dados são representados normalmente a partir de três componentes básicos [Câmara et al. 1996]: 1) não-espacial (ou convencional), descreve o fenômeno geográfico; 2) espacial, informa sua localização, propriedades geométricas e topológicas; e 3) temporal, armazena a data de coleta.

2.2 Modelos para BDG

Existem dois modelos formais para representar as entidades geográficas do mundo real [Câmara et al. 1996]: geo-campos e geo-objetos. O modelo de geo-campos visualiza o espaço geográfico como uma superfície contínua, que não possui uma limitação espacial perfeitamente definida (e.g., clima, vegetação, precipitação). Geo-campo representa um atributo que possui valores em todos os pontos pertencentes a uma região geográfica, onde para cada ponto do espaço um campo terá um valor diferente. O modelo de geo-objetos representa o espaço geográfico como uma coleção de entidades distintas e identificáveis, onde cada entidade é definida por uma fronteira fechada. Um geo-objeto é uma entidade geográfica singular e indivisível, caracterizada por sua identidade, suas fronteiras e seus atributos. Este modelo representa uma superfície ocupada por feições geográficas identificáveis e possuidoras de características e geometrias próprias, definindo seus limites no espaço (e.g., limites municipais, bairros e praças).

Muito se questiona se existem realmente diferenças fundamentais entre geo-campos e geo-objetos, ou se eles não seriam apenas duas maneiras de ver o mesmo tipo de dado. No entanto, existe uma diferença essencial entre eles, que é o papel da fronteira [Câmara et. al. 2006]. A fronteira de um geo-campo é apenas uma divisão relacionada com nossa capacidade de medida, conseqüentemente, o geo-campo pode ser dividido em parte e ainda assim manter sua propriedade essencial (que é a sua função de atributo). Já dentro da fronteira de um geo-objeto, todas as propriedades são constantes, logo, o mesmo não manterá suas propriedades essenciais caso seja dividido.

A *Figura 2.1* mostra um exemplo de geo-campo (a variável associada à imagem é a refletância do solo) e um conjunto de geo-objetos associados (distritos de São Paulo mostrados em tons de cinza, que mostram o índice de exclusão social proporcional a intensidade dos tons de cinza).

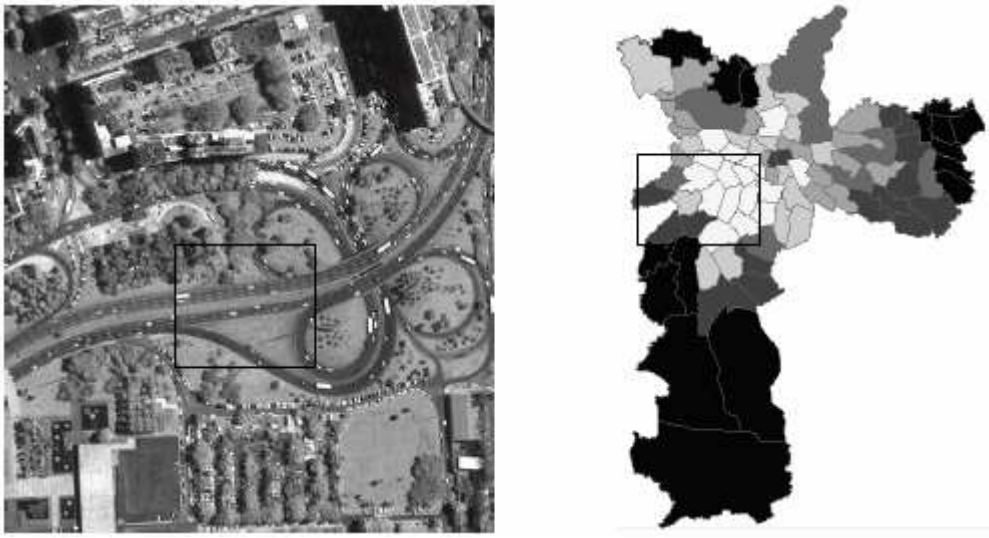

Figura 2.1 – Exemplo de geo-campo (imagem IKONOS do Rio de Janeiro) e de um conjunto de geo-objetos (distritos da cidade de São Paulo). [Câmara et. al. 2006]

2.3 Estruturas de Dados

As estruturas de dados utilizadas em bancos de dados geográficos podem ser divididas em: estruturas vetoriais e estruturas matriciais.

Estruturas vetoriais (vector) são utilizadas para representar objetos geográficos (coordenadas de suas fronteiras). Este formato de estrutura se utiliza de três formas básicas: *pontos, linhas e áreas (polígonos).*

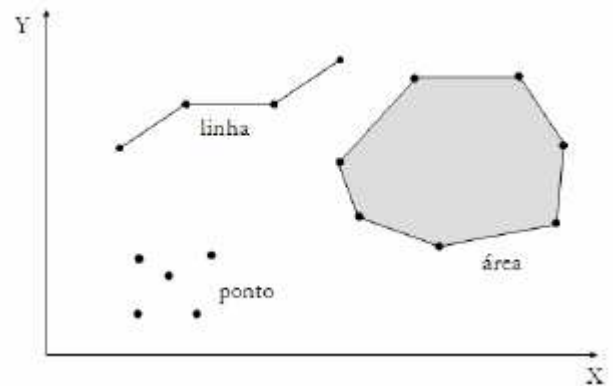

Figura 2.2 – Representações vetoriais em duas dimensões. [Câmara et. al. 2006]

Um ponto identifica localizações ou ocorrências no espaço (e.g., localização de crimes, ocorrências de doenças, e localização de espécies vegetais) e consiste, basicamente, em um par ordenado (x, y) de coordenadas espaciais. Uma linha representa feições unidimensionais e pode ser definida com um conjunto de pontos conectados. Uma área (ou polígono) é a região do plano limitada por uma ou mais linhas poligonais conectadas, de forma que, a fronteira do polígono divide o plano em duas regiões: interior e exterior.

Na estrutura matricial (*raster*), o espaço é representado por uma matriz *P(m, n)* composta por *m* colunas e *n* linhas. Para a representação matricial, é necessário que o espaço possa ser tratado com uma superfície plana, cada célula está associada a uma porção do terreno [Câmara et. al. 2006].

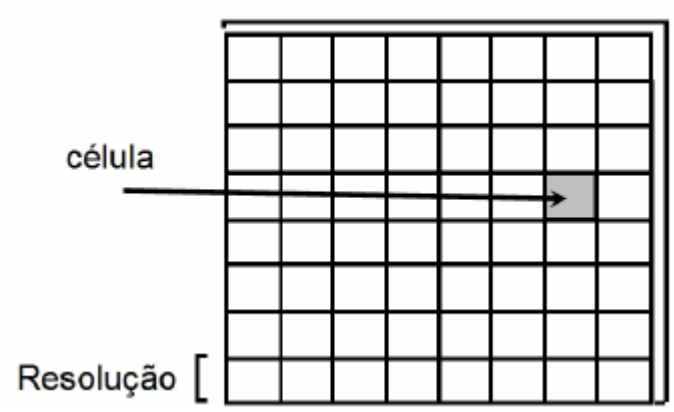

Figura 2.3 – Estrutura matricial. [Câmara et. al. 2006]

Na estrutura matricial, as coordenadas geográficas são obtidas a partir da posição da célula na matriz. A sobreposição de matrizes e a abstração de conjuntos de várias células adjacente em uma única célula são as operações típicas de estruturas matriciais.

2.4 Métodos de Acesso para Dados Espaciais

Os bancos de dados geográficos devem fornecer suporte eficiente para consultas e indexações com base nas informações espaciais armazenadas no banco. As estruturas de índices convencionais, como árvore-B e índices hash, não podem ser

utilizados, pois não fornecem métodos de indexação e consultas de forma eficiente. Estruturas de índices específicas devem ser utilizadas, como árvores-R e quad-tree.

Geralmente uma consulta espacial envolve apenas uma pequena parte do banco de dados. Logo, é necessário a criação de mecanismos de indexação para que as consultas não se tornem ineficientes. Métodos de acesso espacial, ou índices espaciais, são estruturas de dados auxiliares que tornam mais eficiente o processamento de consultas espaciais. Portanto, é importante que a consulta espacial comece com uma fase de filtragem, que identifica previamente os possíveis objetos relacionados com a consulta.

R-tree, é uma estrutura de dados hierárquica derivada da árvore-B. A diferença para a árvore-B está na natureza das chaves: valores numéricos ou alfanuméricos simples, no caso das árvores-B, e pontos extremos de retângulos, no caso das árvores-R (Guttman, 1984).

O que R-Tree busca organizar não é exatamente a forma gráfica do objeto, e sim o seu mínimo retângulo envolvente (minimum bounding rectangle, MBR). Este retângulo é formado a partir da observação dos limites geométrico mínimo e máximo do contorno do objeto, e é expresso pelas coordenadas dos seus pontos inferior esquerdo e superior direito. No caso de estarmos trabalhando com pontos, estes extremos vão coincidir com as coordenadas do próprio objeto, mas isto não compromete o método.

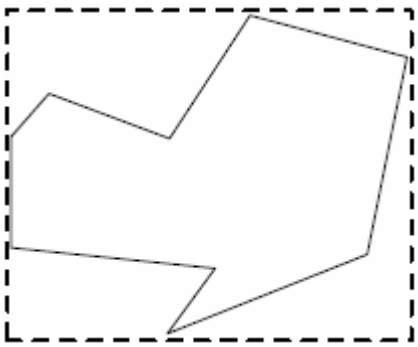

Figura 2.4 – Objeto poligonal e seu retângulo envolvente mínimo. [Câmara et. al. 2006]

Um problema com R-Tree é que a ordem de inserção dos objetos interfere na forma final da árvore, portanto, vai interferir também com o resultado das operações de subdivisão dos nós para manter o balanceamento. Existem algumas técnicas para

tentar aperfeiçoar este comportamento da árvore-R, mas sempre com algum custo adicional em termos de processamento.

2.5 Oracle Spatial

Oracle Spatial [Murray 2003] é uma extensão espacial desenvolvida sobre o modelo objeto-relacional do SGDB Oracle. Este modelo permite definir novos tipos de dados através da linguagem de definição de dados SQL DDL, e implementar operações sobre esses novos tipos. Esta extensão é baseada nas especificações do OpenGIS [OGC 2007] e contém um conjunto de funcionalidades e procedimentos que permitem armazenar, acessar, modificar e consultar dados espaciais de representação vetorial.

O Oracle Spatial é formado pelos seguintes componentes:

- Um modelo próprio de dados chamado MDSYS que define a forma de armazenamento, a sintaxe e semântica dos tipos espaciais suportados.
- Mecanismo de indexação espacial.
- Um conjunto de operadores e funções para representar consultas, junção espacial e outras operações de análise espacial.

O modelo de dados do Oracle Spatial consiste em uma estrutura hierárquica de elementos, geometrias e camadas (layers). Cada camada é formada por um conjunto de geometrias, que por sua vez são formadas por um conjunto de elementos. Os elementos podem ser do tipo Point, LineString e Polygon. Uma geometria pode ser formada por um único elemento ou um conjunto homogêneo (MultiPoint, MultiLineString ou MultiPolygon) ou heterogêneo (Collection). Por sua vez, uma camada é formada por um conjunto de geometrias que possuem os mesmos atributos.

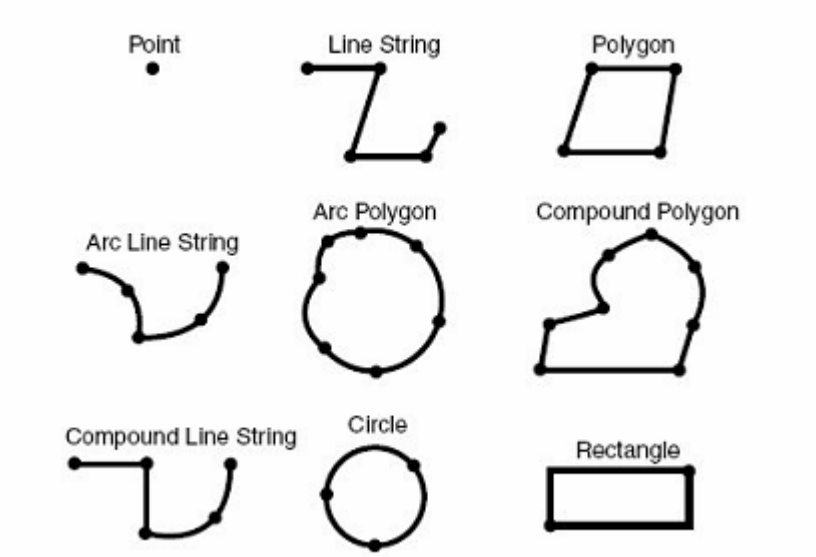

Figura 2.5 – Tipos espaciais primitivos do Oracle Spatial [Murray, C. 2003]

A extensão Spatial suporta o armazenamento e indexação de tipos bidimensionais, tridimensionais e tetradimensionais, no entanto, as funções e operadores só funcionam para os tipos bidimensionais. Os tipos bidimensionais são compostos por pontos formados por duas coordenadas X e Y, que geralmente representam longitude e latitude.

Devido ao Oracle Spatial utilizar um modelo objeto-relacional, cada geometria é armazenada em um objeto SDO_GEOMETRY. Em uma tabela espacial, a geometria é definida como uma coluna do tipo SDO_GEOMETRY que contém informações sobre as coordenadas, tipo, projeção e a geometria do objeto. O objeto SDO_GEOMETRY é definido da seguinte forma:

```
CREATE TYPE sdo_geometry AS OBJECT (
      SDO_GTYPE             NUMBER,
      SDO_SRID              NUMBER,
      SDO_POINT             SDO_POINT_TYPE,
      SDO_ELEM_INFO         SDO_ELEM_INFO_ARRAY
      SDO_ORDINATES         SDO_ORDINATE_ARRAY
);
```

O objeto SDO_GEOMETRY é composto pelos seguintes atributos:

- SDO_GTYPE: formado por quatro números, onde os dois primeiros indicam a dimensão da geometria e os outros dois o seu tipo. Os tipos podem ser: 00 (não conhecido), 01 (ponto), 02 (linha ou curva), 03 (polígono), 04 (coleção), 05 (multipontos), 06 (multilinhas) e 07 (multipolígonos);
- SDO_SRID: utilizado para identificar o sistema de coordenadas, ou sistema de referência espacial, associado à geometria;
- SDO_POINT: é definido utilizando um objeto do tipo SDO_POINT_TYPE, que contém os atributos X, Y e Z para representar as coordenadas de um ponto. Somente é preenchido se a geometria for do tipo ponto, ou seja, se os dois últimos números do SDO_GTYPE forem iguais a “01”;
- SDO_ELEM_INFO: é um vetor de tamanho variável que armazena as características dos elementos que compõem a geometria. As coordenadas de cada elemento são armazenadas em um vetor variável chamado SDO_ORDINATES e são interpretadas através de três números armazenados no SDO_ELEM_INFO:
    - SDO_STARTING_OFFSET: indica qual a posição da primeira coordenada do elemento no SDO_ORDINATES;
    - SDO_ETYPE: indica o tipo do elemento;
    - SDO_INTERPRETATION: indica como o elemento deve ser interpretado juntamente com o SDO_ETYPE.
- SDO_ORDINATES: é um vetor de tamanho variável que armazena os valores das coordenadas da geometria.

O Oracle Spatial utiliza um modelo de consulta baseado em duas etapas, chamadas de primeiro e segundo filtro, como mostrado na Figura 2.6. O primeiro filtro considera as aproximações das geometrias, pelo critério do mínimo retângulo envolvente (MBR), para reduzir a complexidade computacional. Este filtro é de baixo custo computacional e seleciona um subconjunto menor de geometrias candidatas,

que será passado para o segundo filtro. O segundo filtro trabalha com as geometrias exatas, por isso é computacionalmente mais caro e só é aplicado ao subconjunto resultante do primeiro filtro.

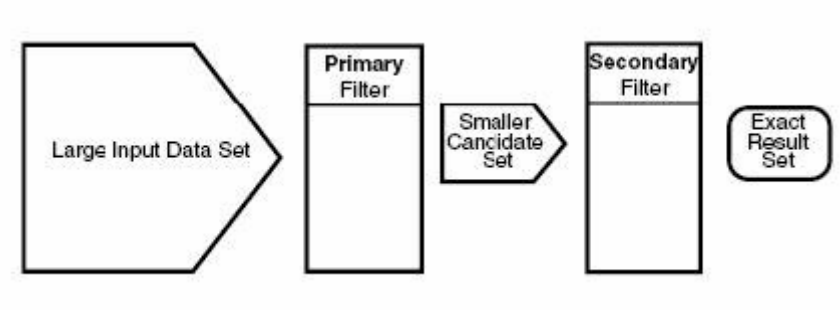

Figura 2.6 – Modelo de consulta [Murray, C. 2003]

2.6 PostGIS

O PostGIS é uma extensão espacial construída sobre o PostgreSQL, desenvolvida pela comunidade de software livre, e que segue as especificações da SFSSQL. O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional, gratuito e open-source.

Na *tabela 2.1* são apresentados os principais tipos de dados convencionais (não geográficos) suportados por PostgreSQL e utilizados em GeoDWCASE.

Tipo de Dado	**Alias**	**Descrição**
SMALLINT	INT2	Inteiro de dois bytes com sinal
INTEGER	INT, INT4	Inteiro de quatro bytes com sinal
BIGINT	INT8	Inteiro de oito bytes com sinal
REAL	FLOAT4	Número de ponto flutuante de precisão simples
DOUBLE PRECISION	FLOAT8	Número de ponto flutuante de precisão dupla
TEXT		Cadeia de caracteres de comprimento variável não limitado
CHARACTER VARYING (n)	VARCHAR (n)	Cadeia de caracteres de comprimento variável com limite.

TIMESTAMP [(p)] WITHOUT TIME ZONE	TIMESTAMP	Data e hora
BOOLEAN	BOOL	Booleano lógico (true/false)
SERIAL	SERIAL4	Inteiro de quatro bytes com auto-incremento

Tabela 2.1 – Tipos de dados espaciais do PostGIS [Câmara et. al. 2006].

Para armazenar números inteiros (números sem a parte fracionária), o PostgreSQL fornece três tipos de dados: *smallint (int2), integer (int, int4) e bigint (int8)*. Cada um desses tipos possuem uma faixa de valores permitidos, a tentativa de armazenar um valor fora dessa faixa resultará em erro. O tipo *smallint* só é utilizado quando o espaço em disco é escasso. O tipo *integer* é o mais utilizado, porque oferece o melhor equilíbrio entre faixa de valores, tamanho de armazenamento e desempenho. O tipo *bigint* é utilizado quando a faixa de valores cobertos pelo *integer* não é suficiente.

Os tipos de dados *real (float4)* e *double precision (float8)* são tipos númericos não exatos de precisão variável. São não exatos pois nem todos os valores podem ser convertidos exatamente para o formato interno, sendo armazenados as suas aproximações. O tipo *real* possui uma faixa de pelo menos 1E-37 a 1E+37, com precisão de pelo menos 6 dígitos decimais. O tipo *double precision* normalmente possui uma faixa em torno de 1E-307 a 1E+308 com precisão de pelo menos 15 dígitos.

O tipo *character varying (n),* ou *varchar (n),* armazena cadeias de caracteres com comprimento de até *n* caracteres. Resultará em erro a tentativa de armazenar uma cadeia de caracteres mais longa em uma coluna deste tipo de dado. É possível utilizar o *character varying* sem o especificador de comprimento, este tipo passa então a aceitar cadeias de caracteres de qualquer tamanho. PostgreSQL também disponibiliza o tipo *text*, que armazena cadeias de caracteres de qualquer comprimento. O tipo *text* não está especificado no padrão SQL.

Timestamp [(p)] wihout zone time (timestamp) é um tipo de dado que armazena tanto data quanto hora. “Without zone time” significa que não é possível armazenar a zona horária. O atributo *p* especifica o número de dígitos fracionários presentos no campo de segundos. O intervalo permitido para *p* é de 0 a 6.

O PostgreSQL disponibiliza o tipo *boolean* (ou simplesmente *bool*) padrão do SQL. O tipo *bool* possue dois estados possíveis: “verdade” (true) ou “falso” (false). O estado “desconhecido” é representado pelo valor *null.* Os valores literais válidos para o estado “verdade” são: *TRUE, ‘t’, ‘true’, ‘y’, ‘yes’, ‘1’.* Já os valores literais válidos para o estado “falso” são: *FALSE, ‘f’, ‘false’, ‘n’, ‘no’, ‘0’.*

O tipo *serial* é meramente uma notação conveniente para definir colunas identificadoras únicas, ou seja, não é um tipo verdadeiro. Tem como função auto-incrementar o atributo escolhido como identificar único (id) da tabela.

Os tipos geométricos espaciais suportados por PostGIS são: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON e GEOMETRYCOLLECTION.

Na *Figura 2.7* são ilustrados os tipos espaciais suportados pelo PostGIS.

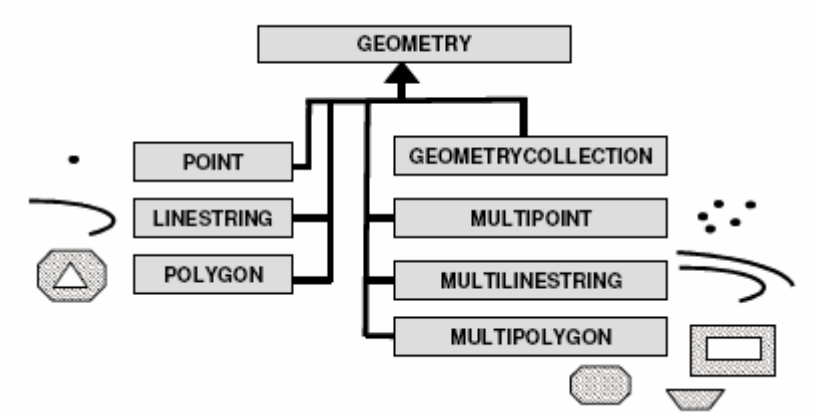

Figura 2.7 – Tipos de dados espaciais do PostGIS [Câmara et. al. 2006].

Esses tipos de dados podem ser representados da seguinte forma:

• POINT: (0 0 0)

• LINESTRING: (0 0, 1 1, 2 2)

• POLYGON: ((0 0 0, 4 0 0, 4 4 0, 0 4 0, 0 0 0), (1 0 0, ...), ...)

• MULTIPOINT: (0 0 0, 4 4 0)

• MULTILINESTRING: ((0 0 0, 1 1 0, 2 2 0), (4 4 0, 5 5 0, 6 6 0))

• MULTIPOLYGON: (((0 0 0, 4 0 0, 4 4 0, 0 4 0, 0 0 0), (...), ...), ...)

• GEOMETRYCOLLECTION: (POINT(2 2 0), LINESTRING((4 4 0, 9 9 0))

A criação de uma tabela com tipo espacial é construída em duas etapas. Na primeira, os atributos básicos (alfanuméricos) são definidos e na segunda, a função AddGeometryColumn é utilizada para adicionar a coluna com o tipo espacial.

Primeira Etapa (definição dos atributos básicos)

```
CREATE TABLE nome_tabela
( atributo01 SERIAL,
  atributo02 VARCHAR(10),
  atributo03 VARCHAR(50),
  PRIMARY KEY (atributo01)
);
```

Segunda Etapa (adição da coluna espacial)

```
SELECT AddGeometryColumn(
     Parâmetro 1,
     Parâmetro 2,
     Parâmetro 3,
     Parâmetro 4,
     Parâmetro 5,
     Parâmetro 6);
```

A função AddGeometryColumn implementada no PostGIS e especificada no OpenGIS, realiza todo o trabalho de preenchimento da tabela de metadados “geometry_columns”. Os parâmetros dessa função são:

• nome do banco de dados [Parâmetro 1];

• nome da tabela que irá conter a coluna espacial [Parâmetro 2];

• nome da coluna espacial [Parâmetro 3];

• sistema de coordenadas em que se encontram as geometrias da tabela [Parâmetro 4];

• tipo da coluna espacial, que serve para criar uma restrição que verifica o tipo do objeto sendo inserido na tabela [Parâmetro 5];

• dimensão em que se encontram as coordenadas dos dados [Parâmetro 6]

2.7 MySQL

MySQL suporta uma extensão espacial que permite gerar, armazenar e analisar características geográficas. MySQL implementa a extensão espacial seguindo a especificação OpenGIS (Open Geospatial Consortium) [OGC 2007]. É implementado um subconjunto do ambiente de SQL com tipo geométricos *(SQL With Geometry Types*) proposto por OGC.

Na tabela 2.2 são apresentados os principais tipos de dados convencionais fornecidos por MySQL.

Tipo de Dado	**Descrição**
TINYINT	Inteiro de um byte com ou sem sinal
SMALLINT	Inteiro de dois bytes com ou sem sinal
MEDIUMINT	Inteiro de três bytes com ou sem sinal
INTEGER	Inteiro de quatro bytes com ou sem sinal
BIGINT	Inteiro de oito bytes com ou sem sinal
FLOAT	Número de ponto flutuante de precisão simples
DOUBLE	Número de ponto flutuante de precisão dupla
TEXT	Cadeia de caracteres de comprimento variável não limitado
VARCHAR (n)	Cadeia de caracteres de comprimento variável com limite.
TIMESTAMP	Combinação de data e hora
BOOLEAN	Booleano lógico (true/false)
SERIAL	Sinônimo de *BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE*

Tabela 2.2 – Tipos de dados do MySQL

MySQL fornece diversos tipos de dados para armazenar números inteiros (números sem a parte fracionária): *TINYINT, SMALLINT, MEDIUMINT, INTEGER,*

BIGINT. A principal diferença entre esses tipos de dados é a faixa de valores que cada um suporta. Todos esses tipos numéricos podem ser especificados com sinal (signed) ou sem sinal (unsigned).

Os tipos de dados *FLOAT* e *DOUBLE* são utilizados para representar valores de dados numéricos aproximados. Para *FLOAT*, o MySQL suporta um atributo opcional (colocado entre parênteses após a palavra chave FLOAT) que especifica a precisão em bits. MySQL trata *DOUBLE* e *REAL* como sinônimos para *DOUBLE PRECISION*. Para maior portabilidade, o código de tipos de dados de números de ponto flutuante dve ser usado sem a especificação da precisão ou o número de dígitos.

Os tipos *varchar(n)* e *text* armazenam cadeia de caracteres de tamanho variável. No caso do tipo *varchar(n)*, o campo que especifica o tamanho máximo é obrigatório. *TimeStamp* é uma combinação de data e hora. A margem vai desde o 1 de Janeiro de 1970 ao ano 2037. O formato de armazenamento depende do tamanho do campo. O tipo *bool (boolean)* é considerado sinônimo de *TINYINT(1).* O valor zero é considerado *falso*. Valores diferentes de zero são considerados *verdadeiros*.

Para facilitar o uso de código SQL de outros sistemas de banco de dados, MySQL mapeia tipos de dados como mostrado na tabela 2.3. Esse mapeamento facilita a importação de definições de tabelas de outros sistemas de banco de dados para MySQL. O mapeamento do tipo de dado ocorre no momento de criação da tabela, após esse processo, a especificação do tipo de dado original é descartada, ficando na tabela criada o tipo de dado equivalente de MySQL.

Tipos de outros SGBD's	**Tipos de Dados MySQL**
BOOL	TINYINT
BOOLEAN	TINYINT
CHARACTER VARYING	VARCHAR (m)
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT

Tabela 2.3 – Mapeamento de tipos no MySQL

A implementação atual das funções geométricas em MySQL não estão tão bem implementadas quanto em outros produtos comerciais (por exemplo, Oracle Spatial). Além disso, a implementação de MySQL é limitada a vetores de dados bidimensionais. No entanto, as funções de MySQL são suficientes para muitas tarefas.

MySQL possui tipos de dados que correspondem as classes do OpenGIS.

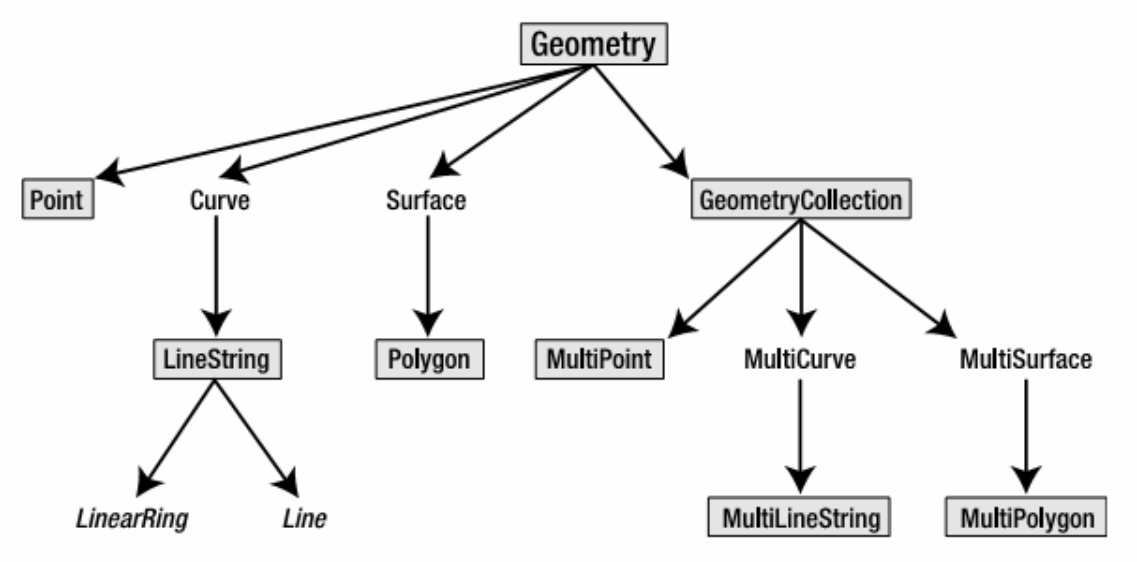

Figura 2.8 – Tipos de dados espaciais do MySQL

- **GEOMETRY**: Tipo geométrico que pode armazenar pontos, linhas e polígonos.
- **POINT:** Um ponto em um sistema de coordenadas; sem dimensão.
- **LINESTRING**: Um ou mais segmentos lineares unindo dois pontos; unidimensional.
- **POLYGON**: Um **LINESTRING** fechado (o fim coincide com o início); bidimensional.
- **MULTIPOINT:** Uma ou mais geometrias do tipo **POINT.**
- **MULTILINESTRING:** Uma ou mais geometrias do tipo **LINESTRING.**
- **MULTIPOLYGON:** Uma ou mais geometrias do tipo **POLYGON.**
- **GEOMETRYCOLLECTION:** Pode armazenar qualquer coleção de objetos de qualquer tipo. No entanto, os outros tipos de coleções

(**MULTIPOINT**, **MULTILINESTRING**, **MULTIPOLYGON),** só podem armazenar coleções de seus tipos específicos.

Exemplo de tabela no MySQL:

```
CREATE TABLE exemplo (
     id       INT NOT NULL PRIMARY KEY,
     border   POLYGON NOT NULL,
     ela      LINESTRING,
     ref      POINT )
```

2.8 Considerações Finais

Neste capítulo, foram apresentadas as principais características dos banco de dados geográficos: modelos para representar as entidades geográficas do mundo real, as estruturas de dados para representar os dados geográficos e os métodos de acesso espaciais. Bancos de dados geográficos são de fundamental importância para a construção de um Data Warehouse Geográfico.

Além disso, foram apresentados os três sistemas de banco de dados (Oracle, PostgreSQL e MySQL), analisando suas características convencionais e espaciais. O Oracle Spatial se diferencia dos outros dois por utilizar um modelo objeto-relacional, onde cada geometria é armazenada em um objeto SDO_GEOMETRY. Postgis e MySQL possuem o mesmo conjunto de objetos geométricos, no entanto, Postgis só adiciona o componente geográfico depois da criação da tabela, enquanto o MySQL especifica o atributo geométrico dentro da tabela.

3. Projeto GOLAPWare

Este capítulo apresenta uma visão geral sobre Data Warehouse Geográfico e os principais componentes do projeto GOLAPWare: a arquitetura GOLAPA, GeoDWFrame, o metamodelo GeoDWM e a ferramenta case GeoDWCASE.

3.1 Apresentação

O grande desafio das corporações é gerenciar um grande volume de dados convencionais e geográficos, de forma que, estas informações tornem-se um diferencial competitivo.

Sistemas de suporte à decisão dão apoio aos líderes de uma organização fornecendo dados de mais alto nível para decisões complexas e importantes. As tecnologias que provêm suporte à tomada de decisão são: Data Warehouse (DW), Online Analytical Processing (OLAP) e Sistemas de Informações Geográficas (SIG). DW e OLAP têm como característica principal o processamento multidimensional, enquanto que SIG são sistemas utilizados para coletar, modelar, armazenar e analisar informações que descrevem propriedades físicas do mundo geográfico. A integração destas tecnologias é de extrema importância, pois, possibilita a análise geográfica dos dados das ferramentas DW e OLAP, assim como, permite que estas cruzem e representem os dados geográficos sob diferentes níveis de detalhes.

O projeto GOLAPWare [GOLAPWare 2007], visa integrar essas tecnologias tendo como GOLAPA (Geographical Online Analytical Processing Architecture) [Fidalgo et al. 2001, Fidalgo et al. 2004a, Fidalgo 2005] como arquitetura de referência.

3.2 Data Warehouse Geográfico

Organizações modernas, cada vez mais, adicionam dados geográficos às suas aplicações (i.e., localização de um fornecedor, localização de um cliente). Os DW (Data Warehouse) [Inmon 1997, Kimball 1996, Kimball et al. 1998] tradicionais sempre possuíram dimensões que armazenavam dados de natureza geográfica. No entanto, uma vez que esses sistemas não possuíam a capacidade de armazenar ou manipular dados espaciais [Malinowski e Zimanyi 2004], eles apenas se limitavam a representar esses dados de um forma alfanumérica.

Data Warehouse é uma importante tecnologia na área de sistemas de suporte à decisão. DW tem como objetivo principal habilitar o usuário especializado (executivo, gerente, analista) a tomar decisões melhores e mais rápidas. O uso inicial do termo DW é creditado a Inmon [Inmon 1997], ele define: "*Data Warehouse é uma coleção de dados orientada a assunto, integrada, não-volátil, variante no tempo, que dá apoio às decisões gerenciais*". Detalhando as principais características de um DW, temos que:

- *Orientado por assunto*: os dados são armazenados de acordo com os assuntos de interesse da organização (por exemplo: produto, cliente, loja), diferentemente dos sistemas transacionais, que armazenam as informações das transações que geraram os fatos.
- *Integrado*: dados oriundos de diversos sistemas operacionais da organização (podendo possuir bases heterogêneas) são integrados. Todo o dado é, anteriormente, tratado, de forma a não possuir inconsistências.
- *Não-volátil*: os dados no DW raramente são modificados. Depois de carregados, geralmente, a única operação realizada sobre eles é a consulta.
- *Variante no tempo*: a cada mudança ocorrida no dado, uma nova entrada é criada e não atualizada, como acontece nos sistemas operacionais. Isso permite ao DW dar apoio a análises de série temporal e de tendências.

O processo de construção de um DW envolve as seguintes atividades:

- Extração de dados de múltiplas fontes heterogêneas (i.e., banco de dados, arquivos texto).
- Formatação dos dados visando à consistência dentro do DW.
- Limpeza dos dados para assegurar a validade. O reconhecimento de dados errôneos e incompletos é de alta complexidade, e a automatização desse processo pode se revelar mais complexo ainda.
- Carga dos dados no DW. Esta tarefa é significativa devido ao alto volume de dados a serem carregados. São necessárias ferramentas de monitoração, bem como métodos para recuperação de cargas incompletas ou incorretas.

O modelo mais utilizado para representação dos dados do DW é o modelo estrela (Star Join) [Inmon 1997, Kimball 1996, Kimball et al. 1998]. Este modelo possui dois tipos de tabelas: 1) Tabela de Fato: armazena os valores numéricos do negocio e também as chaves primárias das dimensões; 2) Tabela de Dimensões: armazenas descrições textuais sobre os fatos. A figura 3.1 ilustra um esquema estrela.

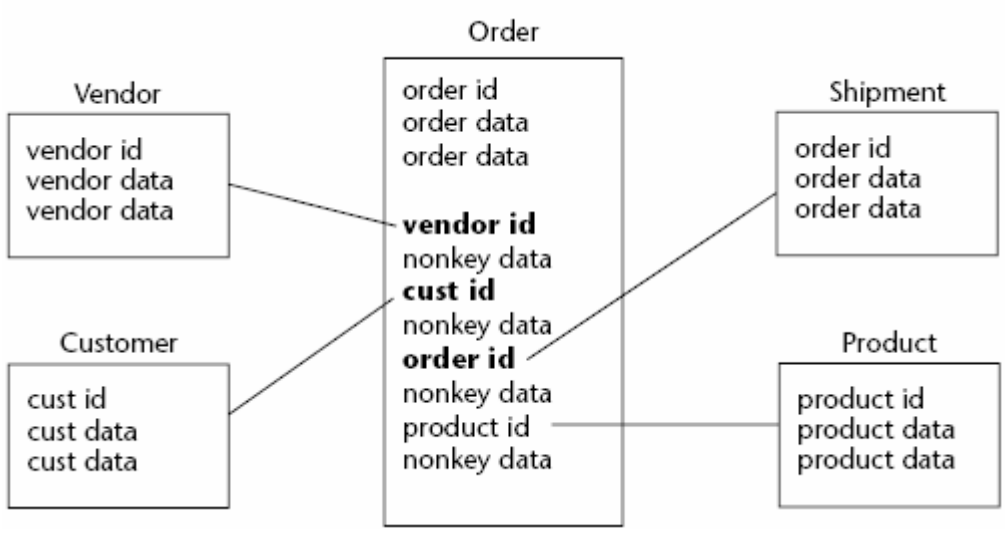

Figura 3.1 – Esquema estrela [Inmon 1997]

O modelo floco de neve (Snowflake) [Inmon 1997, Kimball 1996] é outro modelo existente, é basicamente uma extensão do modelo estrela. A diferença deste modelo para o estrela é que, cada tabela de dimensão é normalizada, quebrando a tabela original em hierarquias existentes em seus atributos. Kimball [Kimball 1996,

Kimball et al. 1998] desaconselha projetistas a transformarem esquemas estrelas em floco de neve, devido a diminuição do desempenho, pois um número de junções maior é exigido, enquanto o ganho em termos de espaço de armazenamento são insignificantes (aproximadamente 1%).

Um DWG (Data Warehouse Geográfico) tem como objetivo gerenciar, eficientemente, grandes quantidades de dados históricos que também incluem dados espaciais. Para isso, DWG deve combinar as características de um DW com as funcionalidade de um BDG. Ou seja, combina métodos eficientes de acesso e gerenciamento de grandes volumes de dados com uma tecnologia apropriada para a manipulação de dados espaciais [Malinowski e Zimanyi 2004].

De acordo com Fidalgo et al. [Fidalgo et al. 2004b], para se construir um DWG, basta acrescentar um componente geográfico no DW tradicional. Basicamente, deve-se acrescentar propriedades geométricas (descritivas e geométricas) ao esquema estrela de um DW, essas propriedades podem ser definidas como dimensões e/ou medidas do DWG. Enquanto as medidas espaciais armazenam somente as geometrias, as dimensões podem armazenar tanto as geometrias quanto as descrições dos objetos geográficos. Dito isto, deve-se ressaltar que um DWG deve manter as características tradicionais de um DW (orientado ao assunto, integrado, não volátil e variante no tempo) assim como oferecer suporte ao armazenamento, à indexação, à agregação e às análises, em mapas ou tabelas, de dados georeferenciados [Fidalgo et al. 2004b].

3.3 Arquitetura GOLAPA

GOLAPA é uma arquitetura para a integração de ferramentas multidimensionais e geográficas. *GOLAPA* visa contribuir para a realização dessa integração através de uma arquitetura de software que favoreça o reuso de componentes de *OLAP* e *SIG* e seja baseada no uso de um *DWG* e de um repositório de metadados. Além disso, *GOLAPA* visa o uso de abordagens abertas e extensíveis (e.g., utilizando tecnologias como *Java*, *MOF*, *XML* e serviços *Web*), no entanto, outros trabalhos podem implementar *GOLAPA* usando as tecnologias que acharem mais pertinentes.

A arquitetura *GOLAPA* é ilustrada na Figura 3.2. Ela está dividida em cinco camadas, onde cada uma fornece serviços para a camada imediatamente superior, fazendo uso dos serviços fornecidos pela camada imediatamente inferior. Além disso, cada uma dessas camadas agrupa seus componentes de forma a prover o máximo de coesão e o mínimo de acoplamento. Podemos relacionar as camadas da seguinte forma:

- ***Camada A*** ou ***Camada de Suporte Operacional***: mantém as aplicações e os dados voltados para o processamento transacional.
- **Camada B** ou ***Camada de Extração, Transformação e Carga de Dados***: converte os dados do ambiente transacional para o ambiente de suporte à decisão.
- ***Camada I*** ou ***Camada de Dados Estratégicos***: armazena os dados do DWG, dividindo o ambiente transacional do ambiente de suporte à decisão.
- ***Camada II*** ou ***Camada de Processamento Multidimensional e/ou Geográfico***: sincroniza a integração das operações analíticas e/ou geográficas.
- ***Camada III*** ou ***Camada de Apresentação dos Dados***: interface gráfica que permite ao usuário a visualização e manipulação dos dados.

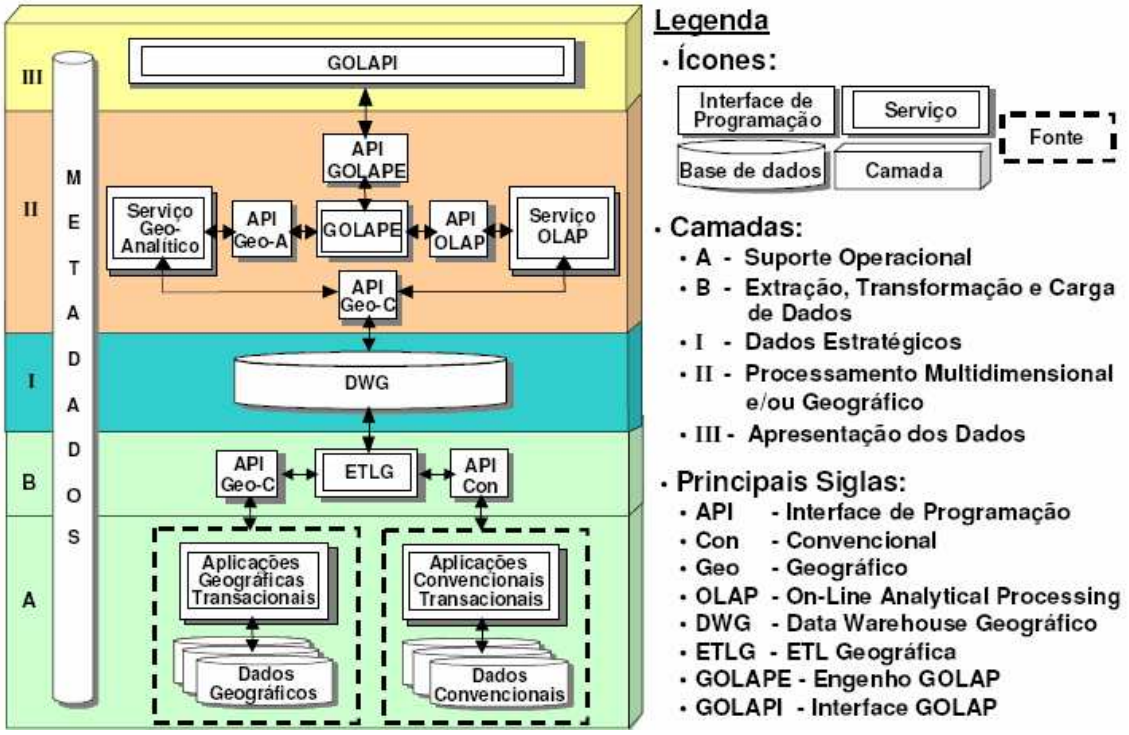

Figura 3.2 – Arquitetura GOLAPA [Fidalgo 2005]

Ainda de acordo com a arquitetura *GOLAPA*, temos os seguintes componentes:

- ***ETLG***: é a junção da tradicional ferramenta de extração, transformação e carga (ETL – Extraction, Transformation e Load) [Kimball et al. 1998] com o suporte a dados geográficos, resultando assim em um processo de ETL Geográfico (ETLG).
- ***DWG*****:** Data Warehouse Geográfico.
- ***Serviço Geo-Analítico*****:** processa dados geográficos para suporte à decisão não transacional.
- ***Serviço OLAP*****:** processa dados multidimensionais.
- ***Metadados*****:** Nesse repositório, encontram-se metadados referentes à construção do DWG e a integração entre os processamentos multidimensional (OLAP) e geográfico (SIG) [Fidalgo et al. 2003].
- ***GOLAPE*****:** é o engenho de consultas multidimensionais e/ou geográficas de GOLAPA. Fazendo uso de sua interface de programação (API GOLAPE), recebe e responde consultas enviadas pela interface gráfica (GOLAPI).
- ***API GOLAPE*****:** possibilita que o componente GOLAPI requisite e receba dados do engenho GOLAPE.
- ***GOLAPI*****:** permite ao usuário realizar, visualizar e analisar consultas multidimensionais (OLAP), geográficas (SIG) ou multidimensionais e geográficas (OLAP + SIG).
- ***API Com, GEO-C, GEO-A e OLAP*****:** são interfaces de programação utilizadas, respectivamente, para acessar os serviços convencional, geo-convencional, geo-analítico e multidimensional.

3.4 GeoDWFrame

Visando orientar a definição do projeto do esquema dimensional e geográfico de um DWG, Fidalgo et al. [Fidalgo et al. 2004b, Fidalgo 2005] propuseram o arcabouço GeoDWFrame. Este possui as seguintes orientações: 1) não usar conceito de medidas espaciais; 2) normalizar os dados geométricos referentes aos objetos espaciais; 3) empregar um conjunto de conceitos, tipos de dimensões e princípios de projetos para gerenciar mais de uma dimensão espacial; 4) utilizar objetos espaciais

em qualquer nível dimensional e 5) armazenar os dados descritivos sobre as localizações dos objetos espaciais. Visando resolver estas questões, GeoDWFrame propõe dois tipos de dimensões: geográfica e híbrida. A dimensão geográfica pode ser classificada como primitiva ou composta, já a dimensão híbrida pode ser micro, macro ou conjunta. Além disso, GeoDWFrame suporta dimensões convencionais que geralmente estão presentes nos DW tradicionais.

A dimensão geográfica primitiva armazena o componente espacial (geometrias) de um objeto geográfico. Pode ser implementado a partir de duas abordagens:

1. *Com campos longos (e.g., Text, Long ou BLOB):* permite armazenar o componente espacial em um formato textual, mas o processamento espacial (e.g. operação e indexação espacial) é executado por um software geográfico e não pelo SGBD. Esta abordagem possui algumas desvantagens [Câmara et al. 2006]: 1) campos longos não possuem semântica; 2) estes campos não possuem métodos de acesso; 3) SQL oferece apenas operadores elementares para tratá-los.

2. *Com extensões para tipos abstratos de dados espaciais (e.g., PostGIS GEOMETRY, Oracle SDO_GEOMETRY):* o SGBD é responsável pela realização do processamento espacial. De acordo com Güting [Güting 1994], as extensões espaciais possuem as seguintes características: 1) fornecem tipos de dados espaciais em seu modelo de dados e mecanismos para manipulá-los; 2) estendem SQL para incluir operações sobre tipos de dados espaciais, transformando-a em uma linguagem para consultas espaciais; 3) adaptam outras funções de nível mais interno ao SGBD para manipular dados espaciais eficientemente, tais como métodos de armazenamento e acesso, e métodos de otimização de consultas.

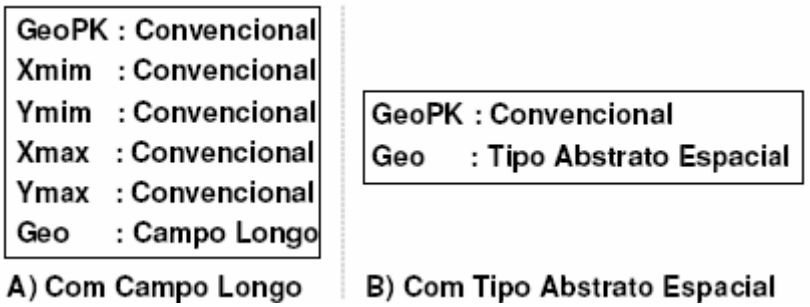

Figura 3.3 – Esquemas para representar uma dimensão geográfica primitiva. [Fidalgo 2005]

Por sua vez, a dimensão geográfica composta manipula as descrições das localizações de um objeto geográfico. Seu esquema contém sua chave primária, campos que descrevem as localizações dos objetos geográficos e campos que contém as chaves estrangeiras de suas respectivas dimensões geográficas primitivas. A importância das dimensões geográficas primitivas está relacionado ao fato de que, se as mesmas não existissem, uma dimensão com objetos geográficos teria que armazenar as geometrias destes objetos. Isto acarretaria um alto custo de armazenamento e gerenciamento de dados, pois existe uma redundância de dados natural das dimensões de um DW e o custo de armazenamento dessas geometrias é alto.

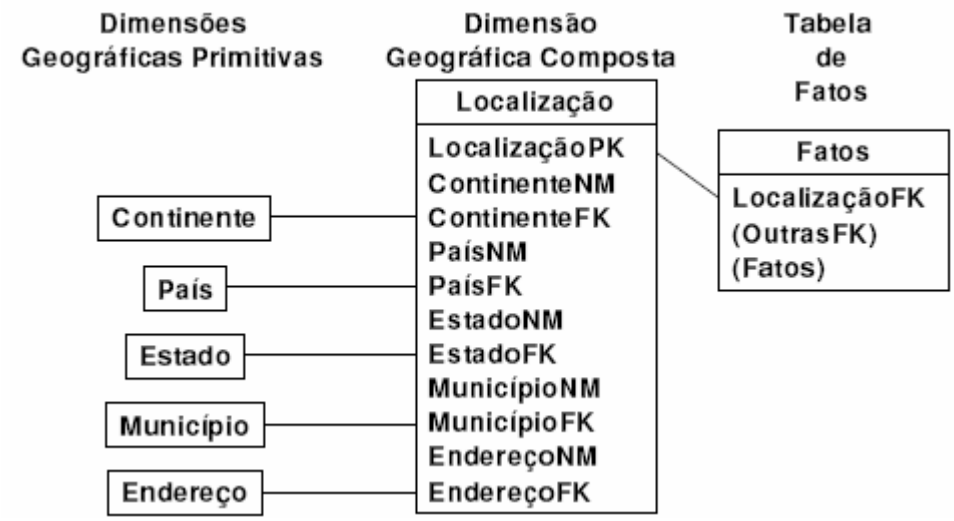

Figura 3.4 – Esquema de uma dimensão geográfica composta. [Fidalgo 2005]

A dimensão híbrida, como visto anteriormente, pode ser classificada como micro, macro ou conjunta. A dimensão híbrida micro trata tanto dados convencionais quanto descrições das localizações geográficas, sendo que estas descrições representam a menor granularidade espacial que, dificilmente são compartilhadas com outros registros.

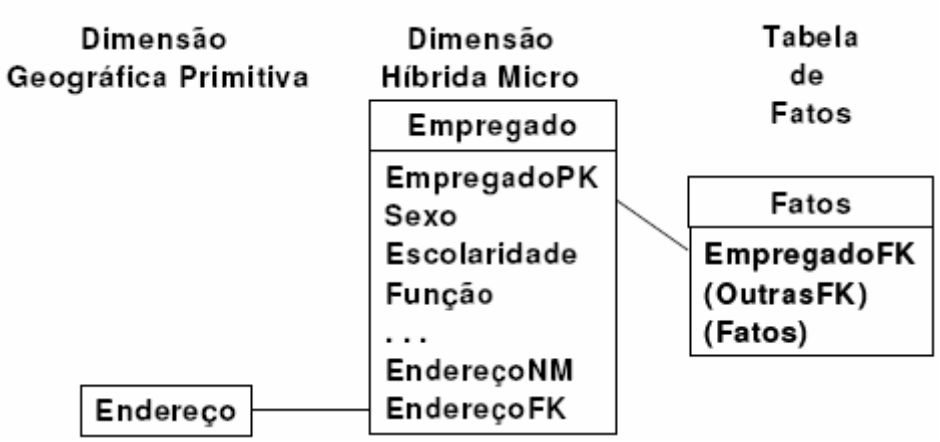

Figura 3.5 – Esquema de uma dimensão híbrida micro. [Fidalgo 2005]

A dimensão híbrida macro, por sua vez, manipula dados de alta granularidade que normalmente são compartilhados com outros registros. O esquema de uma dimensão híbrida macro possui duas abordagens: 1) *única junção* que consiste em um campo convencional mais uma chave estrangeira para uma dimensão geográfica composta. Possui como característica minimizar o número de chaves estrangeira em uma dimensão híbrida macro e permite que a dimensão geográfica composta seja usada como uma mini-dimensão ou uma dimensão baseada em papéis [Kimball 1996] [Kimball et al. 1998]; 2) *múltiplas junções* consiste em campos convencionais mais múltiplas chaves estrangeiras para dimensões geográficas primitivas. Esta abordagem é recomendada quando se deseja minimizar alterações do seu esquema. A Figura 3.6 representa a abordagem de *única junção* enquanto a Figura 3.7 ilustra a de *múltiplas junções.*

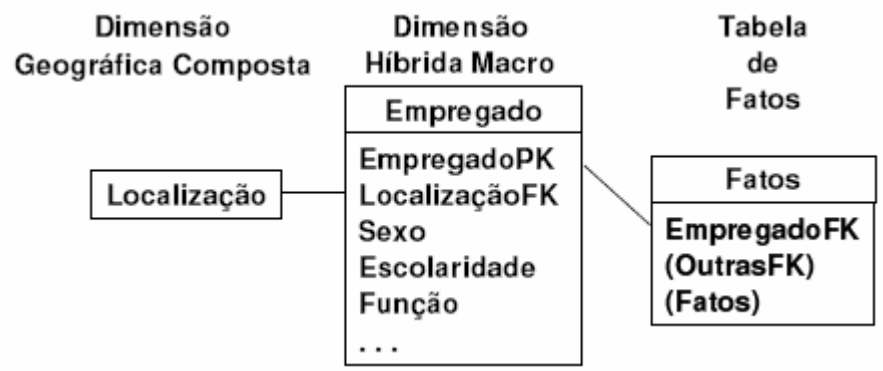

Figura 3.6 – Esquema de uma dimensão híbrida macro com única junção. [Fidalgo 2005]

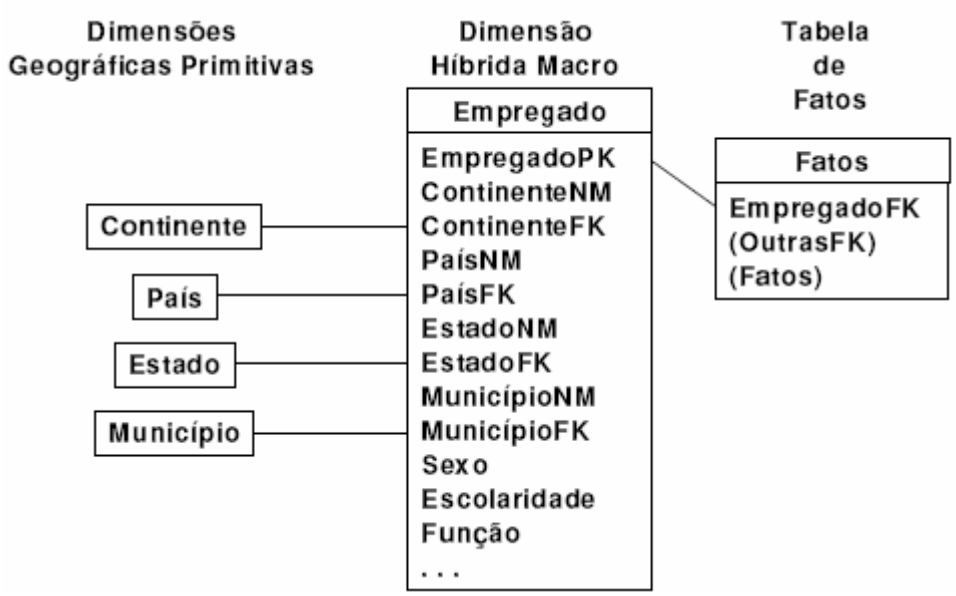

Figura 3.7 – Esquema de uma dimensão híbrida macro com múltiplas junções. [Fidalgo 2005]

A dimensão hibrida conjunta é a união dos conceitos das dimensões híbridas micro e macro em uma única dimensão, pode utilizar as abordagens de *única junção* e *múltiplas junções*. A Figura 3.8 ilustra um exemplo de uma dimensão híbrida conjunta com uma *única junção*.

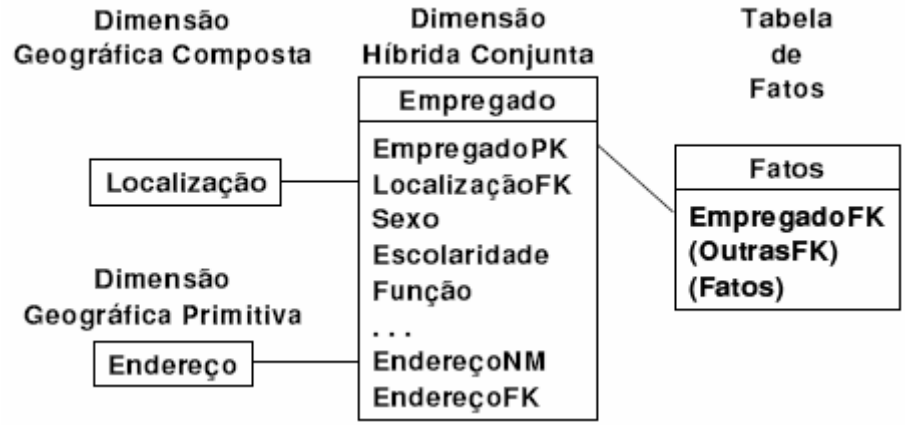

Figura 3.8 – Esquema de uma dimensão híbrida conjunta com única junção. [Fidalgo 2005]

3.5 GeoDWM

GeoDWM [Fonseca 2007] é um metamodelo que:

1. possui uma especificação não-ambígua e de fácil entendimento, para isso, é especificado utilizando restrições OCL (Object Constraint Language) [OMG 2006] e diagramas de classes da UML [OMG 2007];
2. baseia-se no pacote Relational de DWM e na SFS-SQL do OGC para facilitar a sua utilização e extensão por outros trabalhos;
3. define como os conceitos de um modelo dimensional e geográfico podem ser organizados e relacionados para descrever um DWG;
4. fornece um conjunto de estereótipos com pictogramas que têm o objetivo de facilitar e orientar o projetista na atividade de modelagem do DWG;
5. serve de metamodelo base para ferramentas CASE que busquem a modelagem conceitual e geração automática de esquemas lógicos de DWG; e,
6. possibilita a verificação de consistência dos esquemas desenvolvidos através de suas restrições OCL.

A figura 3.9 mostra o metamodelo GeoDWM.

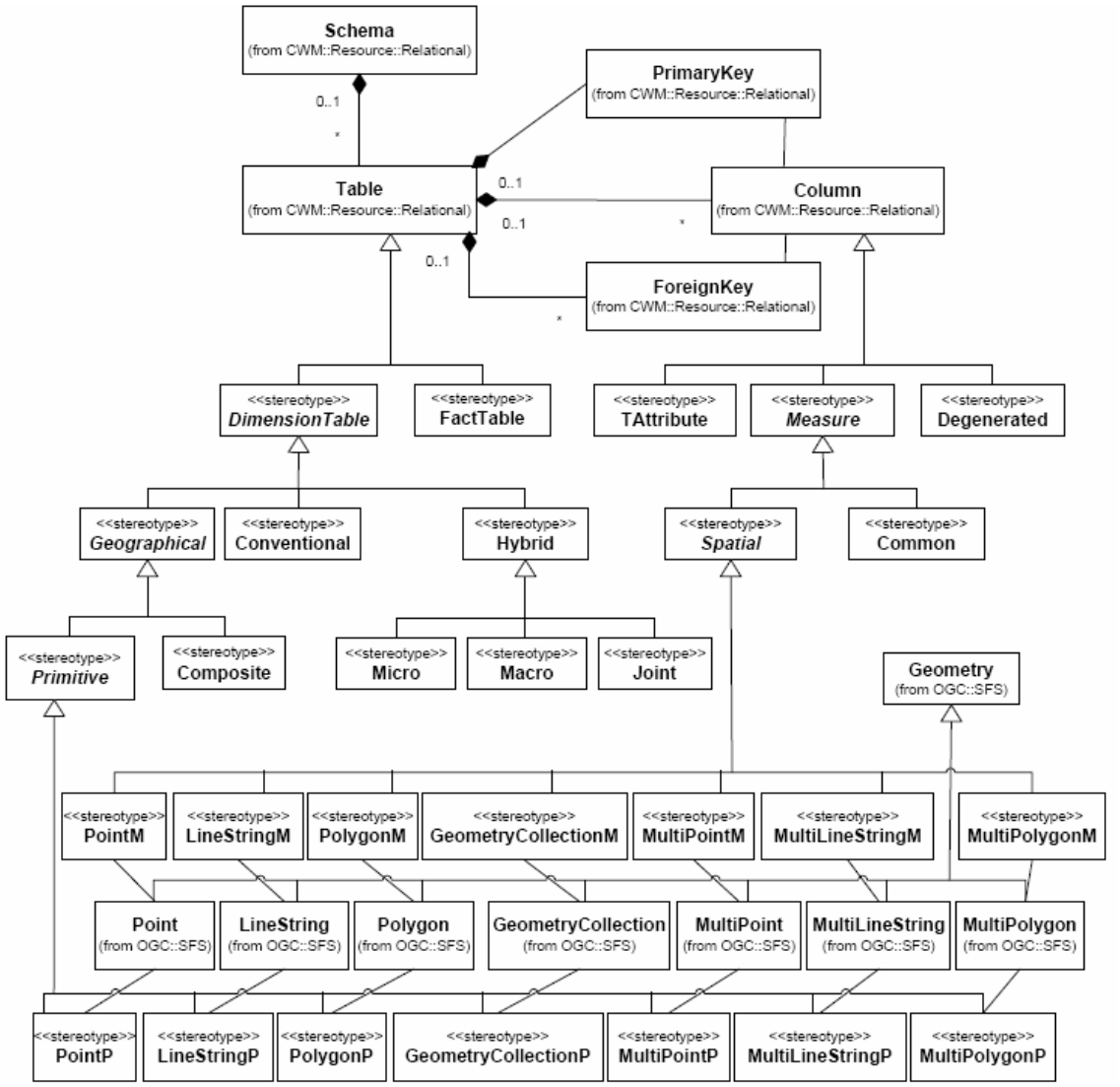

Figura 3.9 – O Metamodelo GDWM [Fonseca 2007]

As classes Esquema (Schema), Tabela (Table), Coluna (Column), Chave Primária (PrimaryKey) e Chave Estrangeira (ForeignKey) são do pacote Relational de CWM. A classe Esquema (Schema) é a base da navegação em GeoDWM e representa o esquema de um DWG. Um esquema é um conjunto nomeado de zero ou mais tabelas (Table). Tabelas são compostas por zero ou mais Colunas (Column), por no máximo uma restrição de Chave Primária (PrimaryKey) e por zero ou mais Chaves Estrangeiras (ForeignKey). Estas últimas associam colunas de uma tabela com colunas de outra tabela. Em um diagrama com base em GeoDWM, o relacionamento entre duas tabelas implica na existência de uma chave estrangeira na tabela de origem para a tabela de destino do relacionamento. Tabelas podem ser especializadas

em Tabelas de Fatos (FactTable) e Tabelas de Dimensão (DimensionTable), enquanto que colunas são especializadas em Atributos de uma tabela (TAttribute), Dimensões Degeneradas (Degenerated) e Medidas (Measure). Estas últimas, por sua vez, podem ser especializadas em Medidas Comuns (Common) e Medidas Espaciais (Spatial). Medidas espaciais são especializadas em classes que estão associadas a uma classe da SFS-SQL para padronizar e representar geometrias do tipo ponto (PointM), cadeia de linhas (LineStringM), polígono (PolygonM), coleção de geometrias (GeometryCollectionM), múltiplos pontos (MultiPointM), múltiplas cadeias de linhas (MultiLineStringM) e múltiplos polígonos (MultiPolygonM).

GeoDWM usa estereótipos UML [OMG 2007] para melhorar a representação gráfica do modelo dimensional e geográfico. Além disso, GeoDWM usa pictogramas para seus estereótipos, de forma a melhorar visualização dos elementos da modelagem. Os Quadros 1 e 2 especificam, respectivamente, os estereótipos de GeoDWM que estão relacionados a fatos e dimensões de um DWG .

Quadro 1 – Estereótipos de GeoDWM relacionados a fatos. [Fonseca et al. 2007a]

Estereótipo	**Pictograma**	**Descrição**
FactTable	F	Tabela de Fatos.
TAttribute	a	Atributo de uma tabela.
Degenerated	d	Dimensão degenerada.
Measure		Medida Abstrata.
Common	$	Medida Convencional.
Spatial		Medida Espacial Abstrata.
PointM		Medida Espacial com geometria Ponto.
LineStringM		Medida Espacial com geometria Cadeia de Linhas.
PolygonM		Medida Espacial com geometria Polígono.
GeometryCollectionM		Medida Espacial com geometria Coleção de Geometrias.
MultiPointM		Medida Espacial com geometria Múltiplos Pontos.
MultiLineStringM		Medida Espacial com geometria Múltiplas Cadeias de Linhas.
MultiPolygonM		Medida Espacial com geometria Múltiplos Polígonos.

Quadro 2 – Estereótipos de GeoDWM relacionados à dimensões. [Fonseca et al. 2007a]

Estereótipo	Pictograma	Descrição
DimensionTable		Tabela de Dimensão Abstrata.
Conventional	D	Dimensão Convencional.
Geographical		Dimensão Geográfica Abstrata.
Composite	□	Dimensão Geográfica Composta.
Primitive		Dimensão Geográfica Primitiva Abstrata.
PointP	□	Dimensão Geográfica Primitiva com geometria Ponto.
LineStringP	□	Dimensão Geográfica Primitiva com geometria Cadeia de Linhas.
PolygonP		Dimensão Geográfica Primitiva com geometria Polígono.
GeometryCollectionP		Dimensão Geográfica Primitiva com geometria Coleção de Geometrias.
MultiPointP		Dimensão Geográfica Primitiva com geometria Múltiplos Pontos.
MultiLineStringP		Dimensão Geográfica Primitiva com geometria Múltiplas Cadeias de Linhas.
MultiPolygonP		Dimensão Geográfica Primitiva com geometria Múltiplos Polígonos.
Hybrid	gH	Dimensão Híbrida genérica.
Micro	μH	Dimensão Híbrida Micro.
Macro	□H	Dimensão Híbrida Macro.
Joint	∪H	Dimensão Híbrida Conjunta.

3.6 GeoDWCASE

GeoDWCASE [Fonseca 2007] é uma ferramenta que possui uma interface com recursos gráficos, implementada sobre a plataforma Eclipse [Eclipse 2007] e é baseada no padrão XMI (XML Metadata Interchange) [OMG 2005] para armazenamento, manipulação, recuperação e intercâmbio de metadados. Oferece, também, suporte para modelagem de classes UML, permite que o esquema do DWG seja validado através das restrições OCL e traduz automaticamente o esquema conceitual em um esquema lógico compatível com o SGBD escolhido.

GeoDWCASE foi desenvolvida sobre a plataforma Eclipse, por causa dos seguintes motivos: é uma plataforma open-source, multiplataforma (disponível para qualquer sistema operacional que possua uma implementação da máquina virtual java) e possui um conjunto de *frameworks*. O principal framework utilizado em GeoDWCASE foi GMF (Geographical Modeling Framework) [GMF 2007], que fornece uma infra-estrutura para o desenvolvimento de editores gráficos baseados em um metamodelo.

Devido a sua arquitetura modularizada, GeoDWCASE permite que extensões sejam implementadas com facilidade. Este trabalho pretende, justamente, realizar duas extensões para GeoDWCASE. O objetivo deste trabalho é: estender a ferramenta GeoDWCASE para que gere automaticamente o modelo lógico para o Oracle Spatial e o MySQL.

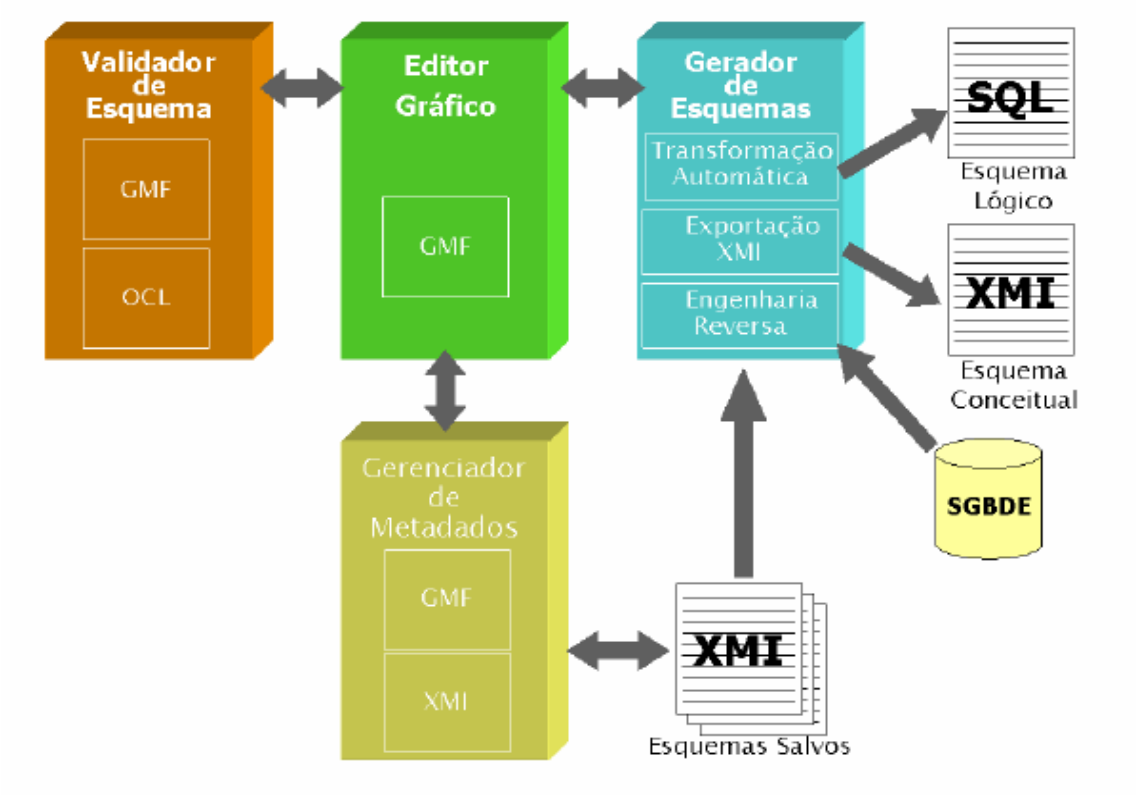

Figura 3.10 – Arquitetura de GeoDWCASE [Fonseca 2007].

Editor Gráfico: interface gráfica que fornece ao usuário uma área de edição, uma paleta de elementos e barras de ferramentas para formatação, validação, exportação, etc.

Gerenciador de Metadados: sistema de repositório de metadados utiliza XMI para gerenciar os metadados do esquema do DWG que está sendo modelado.

Validador de Esquema: implementa o sistema de validação sintática, utilizando restrições OCL definidas em GeoDWM, que verifica a consistência do esquema do DWG.

Gerador de Esquema: implementa o sistema que faz a transformação, importação e exportação de esquemas modelados em GeoDWCASE. Possui três módulos:

1. *Transformação Automática:* transformar automaticamente os esquemas conceituais modelados e salvos em esquemas lógicos de um SGBD espacial em questão;
2. *Exportação XMI:* realizar a exportação dos esquemas conceituais em XMI, e
3. *Engenharia Reversa:* permitir que esquemas lógicos, que estejam em conformidade com GeoDWM, possam ser importados e transformados em esquemas conceituais.

Esquema Conceitual: representa o esquema de um DWG em alto nível.

Esquemas Salvos: esquemas salvos pelo usuário no formato XMI.

Esquema Lógico: representa um arquivo contendo a DDL (Data Definition Language) em SQL (Structured Query Language) de um esquema conceitual GeoDWCASE.

SGBDE: um SGBD com extensão espacial pode onde o módulo de engenharia reversa pode importar um esquema lógico e transformá-lo em um esquema conceitual.

4. Implementação dos Plugins

Este capítulo mostra como foi feita a implementação dos plugins para GeoDWCASE. São mostrados também as principais dificuldades encontradas.

4.1 Apresentação

Este trabalho tem como principal objetivo estender a ferramenta GeoDWCASE, para que gere automaticamente o código DDL para os banco de dados Oracle e MySQL. Para isso, os plugins lêem um arquivo, baseado no padrão XMI (XML Metadata Interchage) [OMG 2005] que representa o esquema de um DWG modelado em GeoDWCASE. Com base nesta representação de alto nível do modelo gerado pelo usuário, os plugins geram todo o código DDL em SQL.

4.2 Mapeamento entre os tipos de dados dos SGBD

O framework GeoDWCASE gera um arquivo baseado no padrão XMI, que contém todas as informações necessárias (atributos, relacionamentos entre as tabelas etc), em alto nível, para a geração do código DDL em SQL. A figura 4.1 mostra um trecho de um código XMI gerado por GeoDWCASE. Na mesma figura, está destacado como é especificado o tipo de dado de cada atributo da tabela em questão.

```
<tables xsi:type="geodwm:Joint" name="Bacia HidrogrÃ¡fica" foreignkeys="//@tables.2 //@tables.3">
  <columns xsi:type="geodwm:TAttribute" name="nm_bacia" type="varchar"/>
  <columns xsi:type="geodwm:TAttribute" name="area_bacia" type="float4"/>
</tables>
```

Figura 4.1 – Exemplo de código XMI gerado por GeoDWCASE.

Inicialmente, GeoDWCASE foi projetado para dar suporte apenas ao banco de dados PostgreSQL com a extensão PostGis. Como dito anteriormente, este presente trabalho, tem como objetivo estender as funcionalidades deste framework para que dê

suporte aos bancos de dados Oracle e MySQL. No entanto, foram encontradas algumas dificuldades no mapeamento entre os tipos de dados dos SGBD, pois, o arquivo XMI gerado pelo framework leva em consideração apenas as especificidades do banco de dados PostgreSQL. Abaixo são listadas algumas dificuldades encontradas:

- Em PostgreSQL é possível especificar o tipo de dado *varchar(n)* (que representa uma cadeia de caracteres com tamanho máximo *n*) sem que seja necessário informar o parâmetro *n* (ou seja, o tipo de dado passa a possuir tamanho ilimitado de caracteres), que indica o tamanho máximo da cadeia de caracteres. Ou seja, PostgreSQL aceita que seja escrito somente *varchar* para representar uma cadeia de caracteres. No entanto, tanto Oracle quanto MySQL, exigem a especificação do tamanho máximo da cadeia de caracteres. No framework GeoDWCASE, não é possível especificar o tamanho do atributo *varchar*. Para solucionar essa dificuldade, sempre que é especificada uma tabela que possui um atributo *varchar*, é criado um atributo *varchar* com tamanho máximo de 100 caracteres (*varchar2(100)* para Oracle, *varchar(100)* para MySQL).

- Oracle não possui o tipo de dado *boolean*. MySQL possui o tipo *boolean* da mesma forma que PostgreSQL. Para solucionar esse problema, para cada tabela no modelo que possui um atributo do tipo boolean, este atributo é mapeado para o tipo de dado *char(1)*, que deverá armazenar os seguintes valores: ‘V’ para representar o valor booleano true e ‘F’ para representar false.

Na tabela 4.1 são apresentados os mapeamentos dos tipos de dados entre os sistemas de bando de dados em questão.

PostgreSQL	**Oracle**	**MySQL**
INT2	SMALLINT	INT2
INT4	INTEGER	INT4
INT8	INTEGER	INT8

VARCHAR	VARCHAR(100)	VARCHAR2(100)
TEXT	TEXT	VARCHAR2(1000)
FLOAT4	REAL	FLOAT4
FLOAT8	FLOAT	FLOAT8
TIMESTAMP	TIMESTAMP	TIMESTAMP
BOOL	CHAR(1)	BOOL

Tabela 4.1 – Apresenta o mapeamento realizado.

Com a tabela acima, notamos que a diferença entre os tipos de dados dos SGBD PostgreSQL e MySQL é mínima. Isso se deve ao fato de MySQL mapear os tipos de dados de outros banco de dados. Esse mapeamento facilita a importação de definições de tabelas de outros sistemas de banco de dados para MySQL. Notamos também, que o Oracle possue grandes diferenças dos tipos de dados em relação ao dois outros SGBD, principalmente no dados textuais e no tipo de dados booleano.

4.3 Adição do componente geográfico

Com base no seguinte estudo de caso [Fonseca 2007]. Será feito uma análise das principais diferenças na adição do componente geográfico, entre os três sistemas de banco de dados em questão (Oracle, PostgreSQL e MySQL). Esta análise se concentra na tabela de Fato "Distribuição Agrícola" que possui uma medida convencional "QuantidadePlantacao" para guardar a quantidade colhida e a medida espacial "AreaPlantacao" que, por sua vez, armazena a geometria da área plantada.

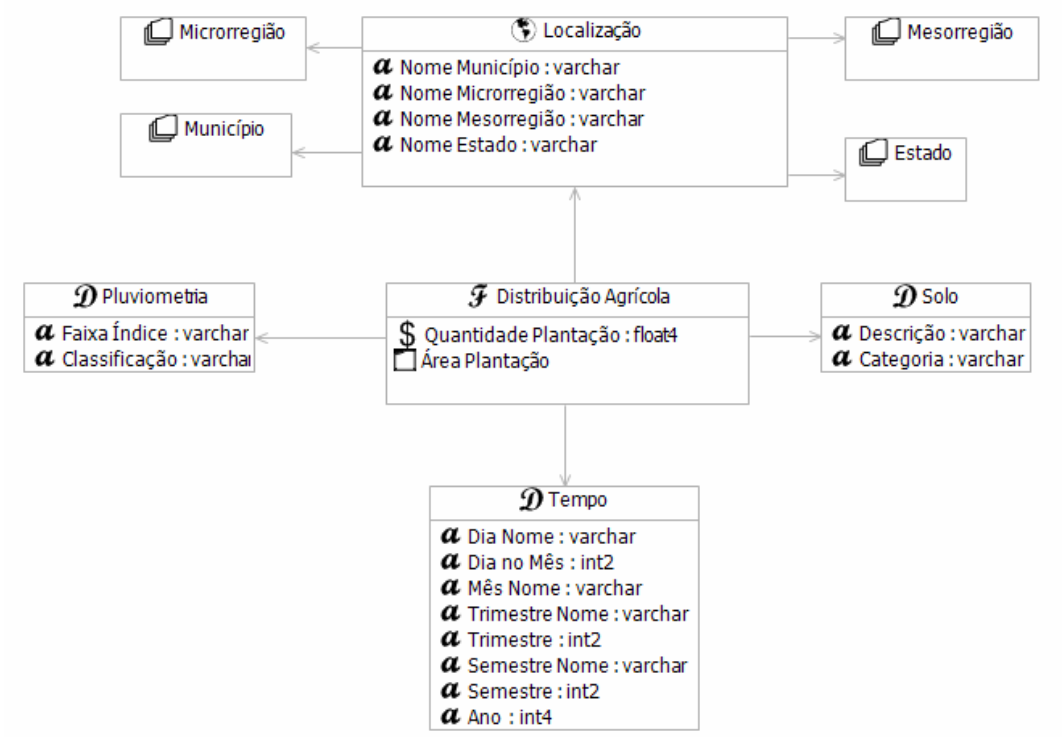

Figura 4.2 – Esquema DWG agrícola [Fonseca 2007].

Cada sistema de banco de dados possui um modo de fornecer suporte a dados geográficos. No Postgis a criação de uma tabela com tipo espacial é construída em duas etapas. Na primeira, os atributos básicos (alfanuméricos) são definidos e na segunda, a função AddGeometryColumn é utilizada para adicionar a coluna com o tipo espacial. O trecho de código abaixo mostra como é codificada, em PostgreSQl com extensão Postgis, a tabela de fato “Distribuição Agrícola”.

```
CREATE TABLE f_distribuicao_agricola
(
fk_cgd_localizacao int4,
fk_d_solo int4,
fk_d_pluviometria int4,
fk_d_tempo int4,
com_quantidade_plantacao float4,
```

```
      CONSTRAINT pk_f_distribuicao_agricola PRIMARY KEY
(fk_cgd_localizacao, fk_d_solo, fk_d_pluviometria, fk_d_tempo)

      )

      WITHOUT OIDS;

      SELECT
AddGeometryColumn('f_distribuicao_agricola','spm_area_plantacao'
,-1,'POLYGON',2);
```

No Oracle Spatial, por utilizar um modelo objeto-relacional, cada geometria é armazenada em um objeto SDO_GEOMETRY. O trecho de código abaixo mostra como é codificada, Oracle Spatial, a tabela de fato “Distribuição Agrícola”.

```
      CREATE TABLE f_distribuicao_agricola

      (

      fk_cgd_localizacao integer,

      fk_d_solo integer,

      fk_d_pluviometria integer,

      fk_d_tempo integer,

      com_quantidade_plantacao real,

      spm_area_plantacao SDO_GEOMETRY,

      CONSTRAINT pk_f_distribuicao_agricola PRIMARY KEY
(fk_cgd_localizacao, fk_d_solo, fk_d_pluviometria, fk_d_tempo)

      );
```

No MySQL, os tipos geométricos espaciais suportados são os mesmos do PostGIS (*POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON e GEOMETRYCOLLECTION*). No entanto, os tipos geométricos são especificados dentro da tabela junto com os atributos convencionais. O trecho de código abaixo mostra como é codificada, em MySQL, a tabela de fato “Distribuição Agrícola”.

```
CREATE TABLE f_distribuicao_agricola

(

fk_cgd_localizacao int4,

fk_d_solo int4,

fk_d_pluviometria int4,

fk_d_tempo int4,

com_quantidade_plantacao float4,

spm_area_plantacao POLYGON,

CONSTRAINT pk_f_distribuicao_agricola PRIMARY KEY
(fk_cgd_localizacao, fk_d_solo, fk_d_pluviometria, fk_d_tempo)

);
```

5 Estudo de Caso

Este capítulo apresenta alguns estudos de casos para ilustrar a utilização da ferramenta GeoDWCASE. É gerado o código DDL para as extenções espaciais Oracle e MySQL.

5.1 Estudo de Caso 1: DWG Agrícola

Neste estudo de caso [Fonseca 2007] existe uma tabela de fatos sobre a distribuição agrícola (f_distribuição_agricola) que possui uma medida convencional "QuantidadePlantacao" para guardar a quantidade colhida e a medida espacial "AreaPlantacao" que, por sua vez, armazena a geometria da área plantada. Há três dimensões convencionais, "SoloTable", "PluviomatriaTable" e "TempoTable", que armazenam, respectivamente, os tipos de solo, dados pluviométricos e tempo. "LocalizaçãoTable" é uma dimensão geográfica que manipula os dados descritivos das geometrias e referências para os objetos geometricos.

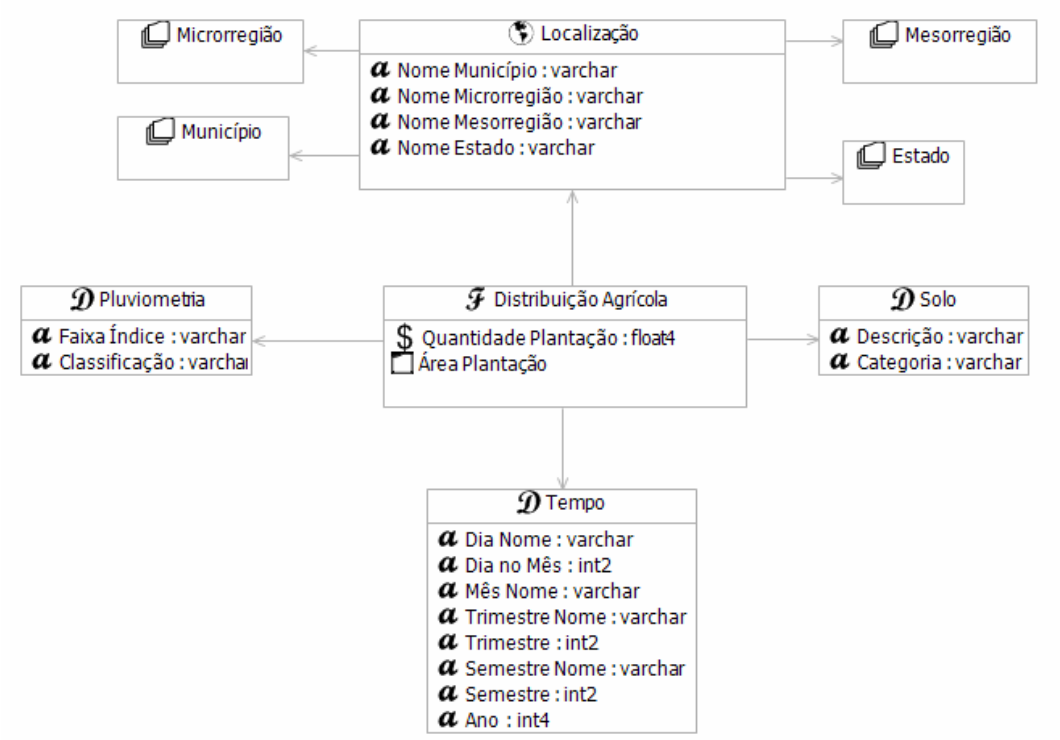

Figura 6.1 – Esquema DWG agrícola [Fonseca 2007].

A seguir é apresentado a DDL do esquema acima para o Oracle Spatial.

```
-- Create Table: f_distribuicao_agricola
CREATE TABLE f_distribuicao_agricola
(
fk_cgd_localizacao number,
fk_d_solo number,
fk_d_pluviometria number,
fk_d_tempo number,
com_quantidade_plantacao number,
spm_area_plantacao SDO_GEOMETRY,
CONSTRAINT pk_f_distribuicao_agricola PRIMARY KEY (fk_cgd_localizacao,
fk_d_solo, fk_d_pluviometria, fk_d_tempo)
);


-- Create Table: d_solo
CREATE TABLE d_solo
(
id_d_solo number NOT NULL,
att_descricao varchar2(100),
att_categoria varchar2(100),
CONSTRAINT pk_d_solo PRIMARY KEY (id_d_solo)
);
```

```
-- Create Table: d_pluviometria
CREATE TABLE d_pluviometria
(
id_d_pluviometria number NOT NULL,
att_faixa_indice varchar2(100),
att_classificacao varchar2(100),
CONSTRAINT pk_d_pluviometria PRIMARY KEY (id_d_pluviometria)
);

-- Create Table: d_tempo
CREATE TABLE d_tempo
(
id_d_tempo number NOT NULL,
att_dia_nome varchar2(100),
att_dia_no_mes number(2),
att_mes_nome varchar2(100),
att_trimestre_nome varchar2(100),
att_trimestre number(2),
att_semestre_nome varchar2(100),
att_semestre number(2),
att_ano number,
CONSTRAINT pk_d_tempo PRIMARY KEY (id_d_tempo)
);


-- Create Table: cgd_localizacao
CREATE TABLE cgd_localizacao
(
id_cgd_localizacao number NOT NULL,
fk_pgd_mesorregiao number,
fk_pgd_estado number,
fk_pgd_microrregiao number,
fk_pgd_municipio number,
att_nome_municipio varchar2(100),
att_nome_microrregiao varchar2(100),
att_nome_mesorregiao varchar2(100),
att_nome_estado varchar2(100),
CONSTRAINT pk_cgd_localizacao PRIMARY KEY (id_cgd_localizacao)
);


-- Create Table: pgd_municipio
CREATE TABLE pgd_municipio
(
id_pgd_municipio number NOT NULL,
spm_pgd_municipio SDO_GEOMETRY,
CONSTRAINT pk_pgd_municipio PRIMARY KEY (id_pgd_municipio)
);


-- Create Table: pgd_microrregiao
CREATE TABLE pgd_microrregiao
(
id_pgd_microrregiao number NOT NULL,
spm_pgd_microrregiao SDO_GEOMETRY,
CONSTRAINT pk_pgd_microrregiao PRIMARY KEY (id_pgd_microrregiao)
);
```

```
-- Create Table: pgd_mesorregiao
CREATE TABLE pgd_mesorregiao
(
id_pgd_mesorregiao number NOT NULL,
spm_pgd_mesorregiao SDO_GEOMETRY,
CONSTRAINT pk_pgd_mesorregiao PRIMARY KEY (id_pgd_mesorregiao)
);


-- Create Table: pgd_estado
CREATE TABLE pgd_estado
(
id_pgd_estado number NOT NULL,
spm_pgd_estado SDO_GEOMETRY,
CONSTRAINT pk_pgd_estado PRIMARY KEY (id_pgd_estado)
);


-- Apply Foreing Key Constraints
ALTER TABLE f_distribuicao_agricola ADD CONSTRAINT
fk_f_distribuicao_agricola_cgd_localizacao FOREIGN KEY
(fk_cgd_localizacao) REFERENCES cgd_localizacao(id_cgd_localizacao);
ALTER TABLE f_distribuicao_agricola ADD CONSTRAINT
fk_f_distribuicao_agricola_d_solo FOREIGN KEY (fk_d_solo) REFERENCES
d_solo(id_d_solo);
ALTER TABLE f_distribuicao_agricola ADD CONSTRAINT
fk_f_distribuicao_agricola_d_pluviometria FOREIGN KEY
(fk_d_pluviometria) REFERENCES d_pluviometria(id_d_pluviometria);
ALTER TABLE f_distribuicao_agricola ADD CONSTRAINT
fk_f_distribuicao_agricola_d_tempo FOREIGN KEY (fk_d_tempo) REFERENCES
d_tempo(id_d_tempo);
ALTER TABLE cgd_localizacao ADD CONSTRAINT
fk_cgd_localizacao_pgd_mesorregiao FOREIGN KEY (fk_pgd_mesorregiao)
REFERENCES pgd_mesorregiao(id_pgd_mesorregiao);
ALTER TABLE cgd_localizacao ADD CONSTRAINT
fk_cgd_localizacao_pgd_estado FOREIGN KEY (fk_pgd_estado) REFERENCES
pgd_estado(id_pgd_estado);
ALTER TABLE cgd_localizacao ADD CONSTRAINT
fk_cgd_localizacao_pgd_microrregiao FOREIGN KEY (fk_pgd_microrregiao)
REFERENCES pgd_microrregiao(id_pgd_microrregiao);
ALTER TABLE cgd_localizacao ADD CONSTRAINT
fk_cgd_localizacao_pgd_municipio FOREIGN KEY (fk_pgd_municipio)
REFERENCES pgd_municipio(id_pgd_municipio);
```

A seguir é apresentado a DDL do esquema para o MySQL.

```
-- Create Table: f_distribuicao_agricola
CREATE TABLE f_distribuicao_agricola
(
fk_cgd_localizacao int,
fk_d_solo int,
fk_d_pluviometria int,
fk_d_tempo int,
com_quantidade_plantacao float(4),
spm_area_plantacao POLYGON,
CONSTRAINT pk_f_distribuicao_agricola PRIMARY KEY (fk_cgd_localizacao,
fk_d_solo, fk_d_pluviometria, fk_d_tempo)
);
```

```
-- Create Table: d_solo
CREATE TABLE d_solo
(
id_d_solo int NOT NULL,
att_descricao varchar(100),
att_categoria varchar(100),
CONSTRAINT pk_d_solo PRIMARY KEY (id_d_solo)
);

-- Create Table: d_pluviometria
CREATE TABLE d_pluviometria
(
id_d_pluviometria int NOT NULL,
att_faixa_indice varchar(100),
att_classificacao varchar(100),
CONSTRAINT pk_d_pluviometria PRIMARY KEY (id_d_pluviometria)
);


-- Create Table: d_tempo
CREATE TABLE d_tempo
(
id_d_tempo int NOT NULL,
att_dia_nome varchar(100),
att_dia_no_mes int,
att_mes_nome varchar(100),
att_trimestre_nome varchar(100),
att_trimestre int,
att_semestre_nome varchar(100),
att_semestre int,
att_ano int,
CONSTRAINT pk_d_tempo PRIMARY KEY (id_d_tempo)
);


-- Create Table: cgd_localizacao
CREATE TABLE cgd_localizacao
(
id_cgd_localizacao int NOT NULL,
fk_pgd_mesorregiao int,
fk_pgd_estado int,
fk_pgd_microrregiao int,
fk_pgd_municipio int,
att_nome_municipio varchar(100),
att_nome_microrregiao varchar(100),
att_nome_mesorregiao varchar(100),
att_nome_estado varchar(100),
CONSTRAINT pk_cgd_localizacao PRIMARY KEY (id_cgd_localizacao)
);


-- Create Table: pgd_municipio
CREATE TABLE pgd_municipio
(
id_pgd_municipio int NOT NULL,
spm_pgd_municipio MULTIPOLYGON,
CONSTRAINT pk_pgd_municipio PRIMARY KEY (id_pgd_municipio)
);
```

```
-- Create Table: pgd_microrregiao
CREATE TABLE pgd_microrregiao
(
id_pgd_microrregiao int NOT NULL,
spm_pgd_microrregiao MULTIPOLYGON,
CONSTRAINT pk_pgd_microrregiao PRIMARY KEY (id_pgd_microrregiao)
);


-- Create Table: pgd_mesorregiao
CREATE TABLE pgd_mesorregiao
(
id_pgd_mesorregiao int NOT NULL,
spm_pgd_mesorregiao MULTIPOLYGON,
CONSTRAINT pk_pgd_mesorregiao PRIMARY KEY (id_pgd_mesorregiao)
);


-- Create Table: pgd_estado
CREATE TABLE pgd_estado
(
id_pgd_estado int NOT NULL,
spm_pgd_estado MULTIPOLYGON,
CONSTRAINT pk_pgd_estado PRIMARY KEY (id_pgd_estado)
);



-- Apply Foreing Key Constraints
ALTER TABLE f_distribuicao_agricola ADD CONSTRAINT
fk_f_distribuicao_agricola_cgd_localizacao FOREIGN KEY
(fk_cgd_localizacao) REFERENCES cgd_localizacao(id_cgd_localizacao);
ALTER TABLE f_distribuicao_agricola ADD CONSTRAINT
fk_f_distribuicao_agricola_d_solo FOREIGN KEY (fk_d_solo) REFERENCES
d_solo(id_d_solo);
ALTER TABLE f_distribuicao_agricola ADD CONSTRAINT
fk_f_distribuicao_agricola_d_pluviometria FOREIGN KEY
(fk_d_pluviometria) REFERENCES d_pluviometria(id_d_pluviometria);
ALTER TABLE f_distribuicao_agricola ADD CONSTRAINT
fk_f_distribuicao_agricola_d_tempo FOREIGN KEY (fk_d_tempo) REFERENCES
d_tempo(id_d_tempo);
ALTER TABLE cgd_localizacao ADD CONSTRAINT
fk_cgd_localizacao_pgd_mesorregiao FOREIGN KEY (fk_pgd_mesorregiao)
REFERENCES pgd_mesorregiao(id_pgd_mesorregiao);
ALTER TABLE cgd_localizacao ADD CONSTRAINT
fk_cgd_localizacao_pgd_estado FOREIGN KEY (fk_pgd_estado) REFERENCES
pgd_estado(id_pgd_estado);
ALTER TABLE cgd_localizacao ADD CONSTRAINT
fk_cgd_localizacao_pgd_microrregiao FOREIGN KEY (fk_pgd_microrregiao)
REFERENCES pgd_microrregiao(id_pgd_microrregiao);
ALTER TABLE cgd_localizacao ADD CONSTRAINT
fk_cgd_localizacao_pgd_municipio FOREIGN KEY (fk_pgd_municipio)
REFERENCES pgd_municipio(id_pgd_municipio);
```

6 Conclusões

Este capítulo apresenta as conclusões finais deste trabalho, mostra suas principais contribuições e os possíveis trabalhos a serem realizados no futuro.

6.1 Considerações Finais

O desenvolvimento dos dois plugins para que a ferramenta GeoDWCASE possa dar suporte ao Oracle Spatial e MySQL, não contribui apenas com um aumento das funcionalidades de GeoDWCASE. O desenvolvimento desses dois plugins para a ferramenta, tem como principal contribuição, o aumento da visibilidade da ferramenta, já que as duas extensões feitas englobam dois sistemas de banco de dados de ampla utilização pelas organizações mundiais.

6.2 Contribuições do Trabalho

Esse trabalho fez as seguintes contribuições:

- Desenvolveu dois módulos de geração automática de código DDL, estendendo a ferramenta GeoDWCASE para dar suporte a Oracle Spatial e MySQL.
- Apresentou as dificuldades na elaboração dos plugins, possibilitando que ocorram melhorias na ferramenta GeoDWCASE

6.3 Trabalhos Futuros

- Acompanhar as atualizações do modelo GeoDWM e da ferramenta GeoDWCASE, e fazer as modificações necessárias nos plugins para que dêem suporte as últimas características implementadas em GeoDWM e GeoDWCASE.
- Desenvolver um módulo que ajude na criação dos dados geográficos para o Oracle Spatial, visto que, por se tratar de um modelo objeto-relacional, a complexidade se encontra mais na criação dos dados geográficos do que na definição de tais dados.

Referências Bibliográficas

[Câmara et al. 1996] Câmara, Gilberto, Casanova, Marco A., Hemerly, Andrea S., Magalhães, Geovane C., e Medeiros, Claudia M. B. (1996). "Anatomia de Sistemas de Informação Geográfica". In SBC X Escola de Computação.

[Câmara et. al. 2006] Câmara, G., Laender, A. H. F., Davis, C., Brauner, D. F., Queiroz, G. R., Ferreira, K. R., Borges, K. A. V., Alves, L., Vinhas, L., Carvalho, M. T. M., Casanova, M. A., Vera, M. S., Oliveira, O. F., Lima, P. O., Souza, R. C. M., Dias, T. L. (2006). "Tutorial sobre Bancos de Dados Geográficos". Disponível em: http://www.dpi.inpe.br/TutorialBdGeo_GeoBrasil2006.pdf. Último acesso em: Outubro, 2007.

[Eclipse 2007] Eclipse Platform. (2007). Disponível em: http://www.eclipse.org/. Último acesso em: Outubro, 2007.

[Elmasri e Navathe 2005] Elmasri , R. e Navathe, S. B. (2003). "Sistemas de Banco de Dados". Addison-Wesley, 4ª edição.

[Fidalgo et al. 2001] Fidalgo, R. N., Times, V. C., e Souza, F. F. (2001). Golapa: Uma Arquitetura aberta e extensível para Integração entre SIG e OLAP". In Proceedings of the Brazilian Workshop on Geoinformatics (GeoInfo).

[Fidalgo et al. 2003] Fidalgo, R. N., Silva, J., Times, V. C., e Souza, F. F. (2003). "GMLA: A XML Schema for Integration and Exchange of Multidimensional-Geographical Data". In Proceedings of the Brazilian Workshop on Geoinformatics (GeoInfo).

[Fidalgo et al. 2004a] Fidalgo, R. N., Times, V. C., Silva, J. (2004a). "Providing Multidimensional and Geographical Integration Based on a GDW and Metamodels". In Brazilian Symposium on Databases (SBBD), pages 148–162.

[Fidalgo et al. 2004b] Fidalgo, R. N., Times, V. C., Silva, J., e Souza, F. F. (2004b). "GeoDWFrame: A Framework for Guiding the Design of Geographical Dimensional Schemas". In Proceedings of the International Conference on Data Warehousing and Knowledge Discovery, pages 26–37.

[Fidalgo 2005] Fidalgo, R. N. (2005). "Uma Infra-estrutura para Integração de Modelos, Esquemas e Serviços Multidimensionais e Geográficos". Tese de Doutorado. Universidade Federal de Pernambuco, Recife, PE, BR.

[Fonseca 2007] Fonseca, R (2007). "Um Metamodelo e Uma Ferramenta CASE para Data Warehouse Geográfico". Dissertação de Mestrado. Universidade Federal de Pernambuco, Recife, PE, BR.

[Fonseca et al. 2007a] Fonseca, R., Fidalgo, R. N., Silva, J., e Times, V. C. (2007). "Um Metamodelo para a Especificação de Data Warehouses Geográficos". In Proceedings of the Brazilian Symposium on Databases (SBBD), João Pessoa, Brazil.

[Fonseca et al. 2007b] Fonseca, R., Fidalgo, R. N., Silva, J., e Times, V. C. (2007). "GeoDWCASE: Uma Ferramenta para Projeto de Data Warehouses Geográficos". In Demos Session of the Brazilian Symposium on Databases (SBBD), João Pessoa, Brazil.

[GMF 2007] GMF - Graphical Modeling Framework (2007). Disponível em: http://www.eclipse.org/gmf/. Último acesso em: Outubro, 2007.

[Güting 1994] Güting, R. (1994). "An Introduction to Spatial Database Systems". VLDB Journal, v. 3, n.4.

[Inmon 1997] Inmon, W. H. (1997). "Building the Data Warehouse", John Wiley and Sons, 2 edition.

[Kimball 1996] Kimball, R. (1996). “The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses”. John Wiley and Sons, New York.

[Kimball et al. 1998] Kimball, R., Reeves, L., Ross, M., e Thornthwaite, W. (1998). “The Data Warehouse Lifecycle Toolkit”. John Wiley & Sons.

[Malinowski e Zimanyi 2004] Malinowski, E. e Zimanyi, E. (2004). “Representing Spatiality in a Conceptual Multidimensional Model”. In: Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS 2004), p. 12-21.

[Murray, C. 2003] Oracle Spatial User’s Guide Reference 10g Release 1 (10.1). Redwood City, Oracle Corporation.

[MySQL 2008] MySQL Spatial Extensions. (2008). Disponível em: http://www.mysql.org/doc/refman/5.0/en/gis-introduction.html. Último acesso em: Janeiro, 2008.

[OGC 2007] OGC – Open Geospatial Consortium. (2007). Disponível em: http://www.opengeospatial.org/. Último acesso em: Outubro, 2007.

[OMG 2002] OMG – Object Management Group. (2002). “MetaObject Facility (MOF) Specification 1.4”. Disponível em: http://www.omg.org/cgibin/doc?formal/00-04-03. Último acesso em: Outubro, 2007.

[OMG 2003] OMG – Object Management Group. (2003). “Common Warehouse Metamodel (CWM) Specification 1.1”. Disponível em: http://www.omg.org/docs/formal/03-03-02.pdf. Último acesso em: Outubro, 2007.

[OMG 2005] OMG – Object Management Group (2005). “XML Metadata Interchange (XMI) Specification 2.1”, http://www.omg.org/technology/documents/formal/xmi.htm. Último acesso em: Outubro, 2007.

[OMG 2006] OMG – Object Management Group. (2006). “Object Constraint Language (OCL) Specification 2.0”. Disponível em:http://www.omg.org/technology/documents/formal/ocl.htm. Último acesso em: Outubro, 2007.

[OMG 2007] OMG – Object Management Group. (2007). “Unified Modeling Language (UML) Specification 2.1.1”. Disponível em: http://www.omg.org/technology/documents/formal/uml.htm. Último acesso em: Outubro, 2007.

[Oracle Spatial 2008] Oracle. (2008). Disponível em: http://www.oracle.com/. Último acesso em: Janeiro, 2008.

[Rigaux et al. 2002] Rigaux, P., Scholl, M., e Voisard, A. (2002). “Spatial Databases with Applications to GIS”. Academic Press, New York.

Orientador

Prof. Dr. Robson do Nascimento Fidalgo

Aluno

Paulo Roberto de Melo Rodrigues