

**Proposta para o Banco de Dados do projeto
WebMaps**

Rodrigo Grassi Martins

Dissertação de Mestrado

Proposta para o Banco de Dados do projeto WebMaps

Rodrigo Grassi Martins

Dezembro de 2006

Banca Examinadora:

- Claudia Maria Bauzer Medeiros (Orientadora)
- Ricardo da Silva Torres
Instituto de Computação
- Rubens Augusto Camargo Lamparelli Jr.
CEPAGRI-UNICAMP
- Ariadne Maria Brito Rizzoni de Carvalho (Suplente)
Instituto de Computação

**FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IMECC DA UNICAMP**
Bibliotecária: Maria Júlia Milani Rodrigues – CRB8a /2116

M366p Martins, Rodrigo Grassi
Proposta para o banco de dados do projeto WebMaps /
Rodrigo Grassi Martins -- Campinas, [S.P. :s.n.], 2006.
Orientadora : Claudia Maria Bauzer Medeiros
Dissertação (mestrado) - Universidade Estadual de
Campinas, Instituto de Computação.
1. Banco de dados temporais. 2. Sistemas de Informação
Geográfica. 3. Modelagem de dados.. I. Medeiros, Claudia Maria
Bauzer. II. Universidade Estadual de Campinas. Instituto de
Computação. III. Título.

Título em inglês: A proposal for the database of the WebMaps project.

Palavras-chave em inglês (Keywords): 1. Temporal databases. 2. Geographic information system. 3. Data model.

Área de concentração: Sistema de Informação

Titulação: Mestre em Ciência da Computação

Banca examinadora: Profa. Dra. Claudia Maria Bauzer Medeiros (IC-UNICAMP)
Prof. Dr. Rubens Augusto Camargo Lamparelli Jr. (CEPAGRI-UNICAMP)
Prof. Dr. Ricardo da Silva Torres (IC-UNICAMP)

Data da defesa: 12-12-2006

Programa de Pós-Graduação: Mestrado em Ciência da Computação

Proposta para o Banco de Dados do projeto WebMaps

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Rodrigo Grassi Martins e aprovada pela Banca Examinadora.

Campinas, 12 de dezembro de 2006.

Claudia Maria Bauzer Medeiros (Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 12 de dezembro de 2006, pela Banca examinadora composta pelos Professores Doutores:



Prof. Dr. Rubens Augusto Camargo Lamparelli
CEPAGRI – UNICAMP.



Prof. Dr. Ricardo da Silva Torres
IC – UNICAMP.



Profa. Dra. Claudia Maria Bauzer Medeiros
IC – UNICAMP.

© Rodrigo Grassi Martins, 2006.
Todos os direitos reservados.

Dedicatória

Dedico esta dissertação à minha mãe.

Agradecimentos

Este trabalho não poderia ser terminado sem a ajuda de diversas pessoas às quais presto minha homenagem:

Aos meus pais, Eliana Aparecida Grassi e Mauricio Basiquetto Martins pelo incentivo em todos os momentos da minha vida.

Ao meu irmão Ricardo Grassi Martins, aos meus avós José Grassi e Nair Rosado Grassi, ao meu tio Edson, minha tia Sandra e os meus primos Marcel, Murilo e Márcio pelo apoio em todas as minhas empreitadas.

A minha namorada Sabrina Coelho Rodrigues pelo apoio e carinho.

A minha orientadora, Claudia Bauzer Medeiros, que me mostrou os caminhos a serem seguidos sempre de forma cordial e amistosa. Os pontos de vista e eventuais erros contidos nesta dissertação são de minha responsabilidade, visto que sempre foi me dada liberdade para tomar quaisquer decisões.

A todos os professores do instituto, que ajudaram de forma direta e indireta na conclusão deste trabalho.

A Universidade Estadual de Campinas

Ao CNPq pela bolsa concedida

Aos colegas do LIS Luiz Celso, Luciano, Alan, Andréia, André, Carla, Javier, Senra, Evandro, Jaudete, Cristiano.

Aos colegas do Projeto WebMaps Felipe, Wesley e Danilo.

Aos colegas de curso e de vida Ivan, Bruno, Dadah, Leonel, Leo, Thiago

Aos colegas que tive o prazer de conhecer e conviver durante minha estadia em Campinas: Fernando, Viviane, Frazao, Miguelis, Cilene, Pati, Pudim e Baby.

E por último e não menos importante ao São Paulo Futebol Clube por todas as alegrias proporcionadas ao longo dos anos.

Resumo

O objetivo do projeto WebMaps é a especificação e desenvolvimento de um sistema de informação WEB que sirva de apoio ao monitoramento e planejamento de safras agrícolas no Brasil. Os trabalhos necessários para tais objetivos são alvo de pesquisa multidisciplinar de ponta mundial. Um dos problemas a ser enfrentado é o projeto do banco de dados para o WebMaps. Esta dissertação discute as necessidades do banco de dados do projeto WebMaps e propõe um modelo básico para o mesmo. O banco de dados proposto deve servir como suporte ao cadastro de usuários, propriedades e talhões, além de gerenciar informações sobre os demais dados - em especial - imagens de satélite. As principais contribuições deste trabalho são: especificação de um modelo de dados com suporte ao gerenciamento espaço-temporal, especificação de um conjunto de consultas temporais, espaciais e espaço-temporais, resultando na implementação de um protótipo, utilizando Postgresql/Postgis.

Abstract

The goals of the WebMaps project is the specification and development of a WEB information system to support crop planning and monitoring in Brazil. This kind of project involves state-of-the art research all over world. One of the problems faced by WebMaps is database design. This work attacks this issue, discussing the project's needs and proposing a basic database supports management of users, properties and parciais, as well as others kinds of data, especially satellite images. The main contributions of this work are: specification of a spatio-temporal database model; specification of sets of temporal, spatial and spatio-temporal queries; and the implementation of a prototype, in Postgresql/Postgis.

Sumário

Dedicatória	vii
Agradecimentos	viii
Resumo	ix
Abstract	x
1 Introdução	1
2 Revisão Bibliográfica	3
2.1 Imagens de satélite e aplicações	3
2.2 Sistemas de Informação Geográfica	4
2.3 Bancos de Dados Espaço-temporais	4
2.3.1 Representação do tempo	5
2.3.2 Relacionamentos Temporais	5
2.3.3 Relacionamentos Espaciais	6
2.3.4 Consultas Espaço-Temporais	9
2.4 O modelo de dados geométrico do OGC	11
2.5 PostGis	15
2.5.1 Operadores espaciais	16
2.5.2 Construtores geométricos	17
2.5.3 Funções métricas	17
2.5.4 Saídas geométricas	17
2.5.5 Funções para referenciamento linear	17
2.5.6 Funções de representação geométrica	18
2.5.7 Outras funções	18
2.5.8 Funções de Gerenciamento	18
2.5.9 Funções para suporte a transações longas	18
2.5.10 Usando dados geográficos no PostGis	19

2.6	Conclusões	20
3	Modelo de dados proposto	21
3.1	Características da Aplicação	21
3.2	Modelo de dados	23
3.3	Esquema Relacional	25
3.4	Consultas temporais	27
3.4.1	Consultas que retornam valores de tempo.	27
3.4.2	Consultas que retornam tuplas satisfazendo a um predicado temporal.	28
3.4.3	Consultas que retornam estados temporais.	28
3.5	Consultas espaciais	28
3.5.1	Consultas que retornam componentes espaciais.	28
3.5.2	Consultas que retornam tuplas.	28
3.5.3	Consultas que retornam valores(lógicos ou numéricos).	29
3.6	Consultas espaço-temporais	29
3.6.1	Consultas que retornam componentes espaciais.	29
3.6.2	Consultas que retornam tuplas.	29
3.6.3	Consultas que retornam valores métricos.	30
3.7	Consultas sobre perfis de usuários, evolução de valores de pixels e consultas armazenadas	30
3.8	Resumo	31
4	Implementação	32
4.1	Arquitetura do WebMaps	32
4.2	Funcionamento do Sistema	33
4.2.1	Alguns detalhes da implementação	33
4.2.2	Algumas cópias de tela do protótipo implementado	34
4.3	Consultas Implementadas	35
4.4	Resumo	41
5	Conclusões e Extensões	42
A	Script SQL contendo a implementação do modelo proposto	44
	Bibliografia	50

Lista de Figuras

2.1	Predicados temporais	6
2.2	Exemplos de relacionamento <i>overlap</i>	8
2.3	Exemplos de relacionamento <i>cross</i>	8
2.4	Exemplos de relacionamento <i>touch</i>	9
2.5	Exemplos de relacionamento <i>in</i>	10
2.6	Exemplos de relacionamento <i>disjoint</i>	10
2.7	Modelo de dados geométricos proposto pelo OGC.	12
3.1	Modelo Entidade-Relacionamento.	24
4.1	Arquitetura WebMaps.	33
4.2	Cadastro de usuário.	34
4.3	Cadastro de talhão.	35
4.4	Data inicial da Fazenda Recanto.	35
4.5	Localização e área da Fazenda “Recanto” ao longo do tempo.	36
4.6	Distância entre as fazendas “A” e “Recanto”	37
4.7	Histórico das coordenadas e dos perímetros dos talhões da fazenda “Recanto”	37
4.8	Histórico das coordenadas e dos perímetros em representação geométrica. .	38
4.9	Histórico das culturas cultivadas nos talhões da fazenda “Recanto”	39
4.10	Talhões da Fazenda Recanto.	39
4.11	Relação de fronteira da Fazenda H.	40

Capítulo 1

Introdução

A agricultura tem uma importância significativa na economia brasileira. Estima-se que 40% do PIB Brasileiro provenha das atividades agrícolas [13]. Apesar de lucrativa, a atividade agrícola oferece riscos e necessita de um bom planejamento. Dentro deste contexto a computação aparece como um forte aliado. Esta dissertação está dirigida a este problema, tendo sido desenvolvida como parte do projeto WebMaps.

O objetivo do projeto WebMaps é a especificação e desenvolvimento de um sistema de informação WEB que sirva de apoio ao monitoramento e planejamento de safras agrícolas no Brasil. Os trabalhos necessários para tais objetivos são alvo de pesquisa multidisciplinar de ponta mundial. Do lado da Computação, exigem novos resultados em: bancos de dados e serviços Web (gerenciamento de fontes de dados volumosos e heterogêneos contendo de imagens de satélite, mapas digitais, dados climáticos, econômicos e outros), processamento de imagens (algoritmos específicos para segmentação e recuperação por conteúdo), engenharia de software (especificação, testes e desenvolvimento do software), redes e sistemas distribuídos (para integrar e processar os dados e ferramentas e disponibilizá-los na Web) e interfaces (contemplando múltiplos tipos e propósitos de interação). Do lado do domínio alvo (Ciências Agrárias), problemas ocorrem na identificação e amostragem de dados, uso de sensoriamento remoto e em algoritmos para análise dos dados e sua visualização.

O objetivo desta dissertação é analisar as necessidades para o conteúdo do banco de dados do projeto, propondo um modelo básico. Este modelo deve servir como suporte a diferentes perfis de usuários desde proprietários rurais até pesquisadores em planejamento agrícola. O banco de dados deve permitir gerenciar os tipos de dados do projeto, incluindo imagens de satélite. Em particular, este trabalho envolve análise de questões como gerenciamento de dados geo-referenciados (dados espaciais) que variam ao longo do tempo (dados temporais), caracterizando assim um banco de dados espaço-temporal. Isto envolve o estudo de questões tais como restrições de integridade espacial, representação

do tempo e a manipulação conjunta desses valores. O trabalho se concentra nos desafios relativos à gestão de relacionamentos espaço-temporais e às consultas espaço-temporais. O modelo proposto foi validado através da especificação de um conjunto de consultas e da implementação de um protótipo.

As principais contribuições desta dissertação são:

- Especificação de um modelo de dados com suporte ao gerenciamento espaço-temporal, tendo como estudo de caso o projeto WebMaps, visando atender a usuários de diferentes tipos que desejem usar dados associados à produção agrícola para tomada de decisões (Capítulo 3).
- Especificação de um conjunto de consultas temporais, espaciais e espaço-temporais para validar o modelo de dados proposto (Capítulo 3).
- Implementação de um protótipo, utilizando Postgresql/Postgis, para mostrar a validade do modelo de dados proposto e a adequação das consultas especificadas (Capítulo 4).

O restante desta dissertação está organizado da seguinte forma. O capítulo 2 apresenta a revisão bibliográfica e os conceitos necessários para entendimento do texto. O capítulo 3 apresenta o modelo de dados proposto, seu mapeamento para o modelo relacional e exemplos de consultas permitidas. A seguir o capítulo 4 descreve a arquitetura do WebMaps e algumas consultas implementadas. Finalmente o capítulo 5 apresenta as conclusões da dissertação e propõe algumas extensões para o trabalho. O anexo I contém a especificação em PostGis do esquema do banco de dados gerado.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta a revisão bibliográfica desta dissertação. Para o desenvolvimento desta pesquisa foi necessário o levantamento bibliográfico nas áreas de Sistemas de Informação Geográfica e bancos de dados espaço-temporais. Como o trabalho envolve igualmente pesquisa em banco de dados de imagens de satélite, parte da revisão bibliográfica é dedicada a tal assunto, embora não diretamente usado na dissertação.

2.1 Imagens de satélite e aplicações

Imagens digitais possuem uma posição privilegiada dentro dos dados multimídia. Apesar de vídeo e áudio serem mais explorados pelas indústrias de notícias e entretenimento, imagens ocupam um lugar central em uma imensa variedade de áreas que vão desde história da arte até medicina, passando por astronomia, exploração de petróleo e previsão do tempo. Imagens digitais podem ser utilizadas para auxiliar diversas atividades tais como: agricultura e controle florestal, planejamento urbano e aplicação da lei.

Sensoriamento remoto é uma técnica de aquisição de dados dos objetos existentes na superfície terrestre, sem que haja contato físico direto entre o sensor e o objeto[17].

As aplicações para dados obtidos por sensoriamento remoto são diversas, indo desde a área civil até chegar à área militar e passando por diversas disciplinas. São exemplos de aplicações na área civil: caracterização e estudos da variação na camada de ozônio, identificação e quantificação das causas e efeitos da poluição, previsão de temperatura, monitoramento de erupções vulcânicas, queimadas em florestas, inundações e outros desastres naturais, mapeamento e estudo de mudanças temporais na produção global de biomassa, previsão de colheitas, suporte à agricultura de precisão e monitoramento de mudanças urbanas[21].

A recuperação por conteúdo em imagens de satélite e seu gerenciamento via SGBD são tópicos ainda pouco explorados [6]. Este não é o enfoque do trabalho. Como se verá,

imagens são pre-processadas externamente ao banco de dados.

2.2 Sistemas de Informação Geográfica

O termo Sistemas de Informação Geográfica (SIG) é aplicado para sistemas que realizam o tratamento computacional de dados geográficos e recuperam informações não apenas com base em suas características alfanuméricas, mas também através de sua localização espacial. Oferecem ao administrador (urbanista, planejador, agricultor, engenheiro) uma visão inédita de seu ambiente de trabalho, em que todas as informações disponíveis sobre um determinado assunto estão ao seu alcance, interrelacionadas com base no que lhes é fundamentalmente comum – a localização geográfica [14]. Para que isto seja possível, a geometria e os atributos manipulados em tais sistemas devem estar georreferenciados, isto é, localizados na superfície terrestre e representados segundo alguma escala e projeção cartográfica.

SIGs comportam diferentes tipos de dados e aplicações, em várias áreas do conhecimento. Exemplos de aplicações de SIG são planejamento urbano, administração de recursos naturais, acompanhamento de plantações, controle de epidemias, dentre outros. A utilização de SIGs possibilita uma melhor visualização e interpretação dos resultados, permitindo que diferentes tipos de usuário utilizem o sistema.

O que distingue um SIG de outros tipos de sistemas de informação são as funções de análise espacial, que são aplicadas aos dados de forma a permitir a execução de modelos complexos. Estas funções se baseiam nas coordenadas geográficas para realizar cálculos e comparações. Outro conjunto de funções específicas são as de visualização cartográfica, que transformam o resultado de uma consulta em mapas. A próxima seção apresenta uma visão geral sobre relacionamentos espaciais, uma das bases para construir predicados de consultas espaciais.

2.3 Bancos de Dados Espaço-temporais

Um banco de dados geográfico é um repositório de informação coletada sobre fenômenos do mundo real - as entidades geográficas (por exemplo: florestas, rios, cidades). Um processo geográfico é definido como uma ação exercida ou um efeito sofrido por uma entidade geográfica, modificando-a. O processo começa quando a atividade ou o efeito começa a mudar o estado da entidade e termina quando a atividade ou o efeito cessa, alterando o estado inicial da entidade [4].

O termo *dado espacial* denota qualquer tipo de dado que descreve fenômenos aos quais esteja associada alguma dimensão espacial. Os dados geográficos são comumente carac-

terizados a partir de três componentes fundamentais: atributo, localização e tempo, constituindo assim dados espaço-temporais. O componente *atributo* descreve as propriedades temáticas de uma entidade geográfica, tais como o nome; o componente *localização* informa a localização espacial da entidade. O componente *tempo* descreve os períodos em que os valores daqueles dados geográficos são válidos [11].

O tempo é um aspecto importante de todos os fenômenos do mundo real. Eventos ocorrem em pontos específicos de tempo; objetos e seus relacionamentos têm existência dependente de dimensões temporais. Bancos de dados convencionais (não-temporais) representam o estado do mundo em um único momento de tempo. Dados em um banco de dados não temporal são temporalmente inconsistentes porque se tornam coerentes em pontos de tempo diferentes e desconhecidos. Em contraste, um banco de dados temporal modela o mundo na sua dinâmica, rastreando pontos de mudança e retendo todos os dados [14].

2.3.1 Representação do tempo

Uma representação temporal considera os aspectos de ordem, variação e granularidade[18].

Ordem: Quanto à ordem, o tempo pode ser linearmente ordenado, ramificado ou circular. O tempo linearmente ordenado avança do passado para o futuro de uma maneira totalmente ordenada. O tempo ramificado implica a possibilidade de existirem diferentes histórias futuras ou passadas. Já o tempo circular é utilizado para representar eventos recorrentes.

Variação: Quanto à variação, o tempo pode ser contínuo ou discreto. Para representação computacional é necessário utilizar uma representação discreta do tempo. Uma abordagem bastante utilizada é representar a variação temporal utilizando uma linha de tempo composta por uma sequência de *chronons*. Um chronon é a menor unidade temporal de um sistema, ou seja é um intervalo temporal que não pode ser decomposto.

Granularidade: A granularidade temporal é um parâmetro que corresponde à duração de um chronon. Pode-se considerar, simultaneamente, diferentes granularidades (ano, mês, dia e minuto), para possibilitar uma melhor representação da realidade.

2.3.2 Relacionamentos Temporais

A representação do tempo e do espaço exige a combinação de relacionamentos temporais e espaciais em uma estrutura integrada [3]. Os relações entre intervalos de tempo implicam na definição de um operador de precedência, associado a um conjunto de operadores

típicos da teoria dos conjuntos, tais como união, intersecção, inclusão e igualdade. Allen [1] definiu sete relações : before (antes), meets (toca), during (durante), finishes (finaliza junto com), equal (igual a), overlaps (sobreposição) e starts (inicia junto com). A figura 2.1 ilustra esta situação.

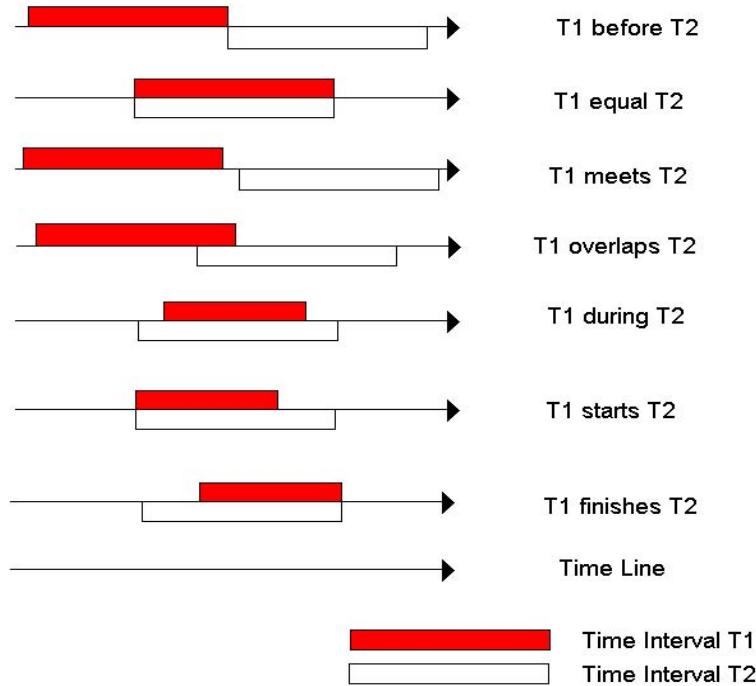


Figura 2.1: Predicados temporais

2.3.3 Relacionamentos Espaciais

Os relacionamentos espaciais são classificados por [12] em três categorias: métricos, topológicos e direcionais e são obtidos a partir de funções aplicadas a um ou mais objetos geográficos. Normalmente aplicados a partir de uma representação vetorial, relacionamentos espaciais são calculados a partir da geometria (ponto, linha, polígono) e localização dos objetos geográficos.

- Ponto é uma geometria 0-dimensional que representa a localização de uma coordenada espacial. Um ponto possui coordenadas x e y. Em espaços 3D, a terceira coordenada pode ser a altura ou o tempo (sistemas espaço-temporais).

- Linha é um conjunto de pontos, com interpolação linear, limitado por dois pontos.
- Área (também chamada de região) é uma superfície planar limitada por uma borda exterior.

Um dos conceitos básicos para o estudo de relacionamentos espaciais são os modelos de 4 e 9 intersecções. Estes modelos se ocupam de restrições topológicas binárias entre regiões sem buracos, sendo baseadas em conceitos algébricos e na teoria dos conjuntos. O modelo de 4-intersecções [8] compara as intersecções das fronteiras (representado pelo símbolo δ) e interiores (representado pelo símbolo $^\circ$) de duas regiões. Os resultados válidos para estas intersecções são vazio ou não vazio. As possíveis combinações para estes relacionamentos resultaram em seis relacionamentos válidos entre regiões: *disjoint, in, touch, equal, cover* e *overlap*.

Em [8], Engenhofer estendeu as 4-intersecções para o método das 9-intersecções que considera as intersecções com o exterior (complemento) da região. A matriz 3X3 a seguir mostra as 9-intersecções entre dois objetos geométricos A e B, onde B^- representa o exterior de B.

$$I_9(A, B) = \begin{bmatrix} A^\circ \cap B^\circ & A^\circ \cap \delta B & A^\circ \cap B^- \\ \delta A \cap B^\circ & \delta A \cap \delta B & \delta A \cap B^- \\ A^- \cap B^\circ & A^- \cap \delta B & A^- \cap B^- \end{bmatrix}$$

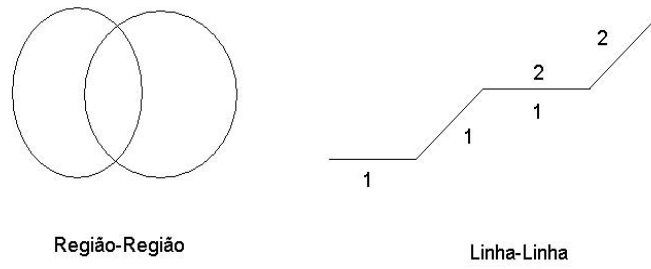
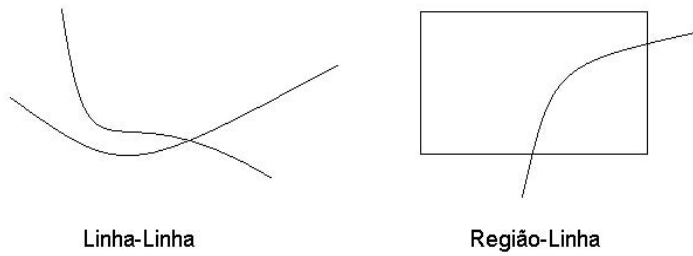
O trabalho de Clementini et al.[7] partiu deste estudo e analisou a dimensão (dim) da intersecção: vazia, pontual, linha ou região (dim=0,1,2), gerando um total de 256 relacionamentos, dos quais apenas 52 são aplicáveis. Estes 52 relacionamentos podem ser especificados pela combinação de operadores de fronteira (from, to, δ) e os relacionamentos topológicos mutuamente exclusivos *disjoint, touch, overlap, in* e *cross*. A seguir são apresentadas as definições destes cinco relacionamentos. As figuras 2 a 5 mostram alguns relacionamentos.

- O relacionamento *overlap*, figura 2.2, somente é aplicável às situações região-região e linha-linha.

$$\langle A, \text{overlap}, B \rangle \iff (\dim(A^\circ) = \dim(B^\circ) = \dim(A^\circ \cap B^\circ)) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$$

- O relacionamento *cross*, figura 2.3, somente é aplicável às situações linha-região e linha-linha.

$$\langle A, \text{cross}, B \rangle \iff (\dim(A^\circ \cap B^\circ) = (\max(\dim(A^\circ), \dim(B^\circ)) - 1)) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$$

Figura 2.2: Exemplos de relacionamento *overlap*.Figura 2.3: Exemplos de relacionamento *cross*.

- O relacionamento *touch*, figura 2.4, não é aplicável somente para situações ponto-ponto.

$$\langle A, touch, B \rangle \iff (A^\circ \cap B^\circ = 0) \wedge (A \cap B) \neq 0$$

- O relacionamento *in*, figura 2.5, é aplicável a todas as situações.

$$\langle A, in, B \rangle \iff (A \cap B = A) \wedge (A^\circ \cap B^\circ \neq 0)$$

- O relacionamento *disjoint*, figura 2.6, é aplicável a todas as situações.

$$\langle A, disjoint, B \rangle \iff A \cap B = 0$$

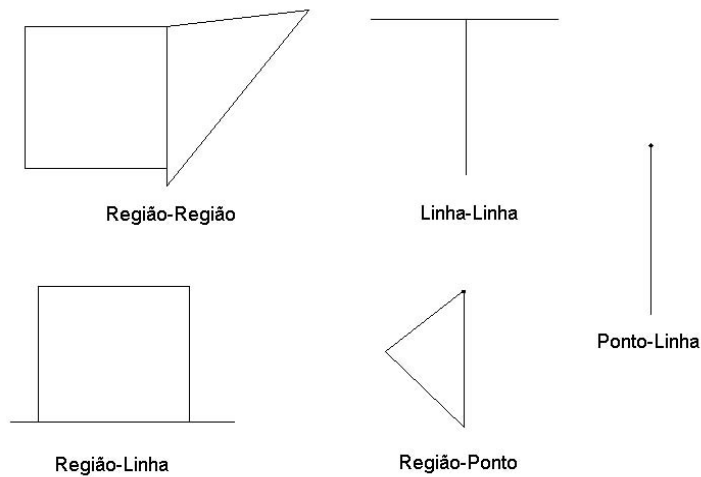


Figura 2.4: Exemplos de relacionamento *touch*.

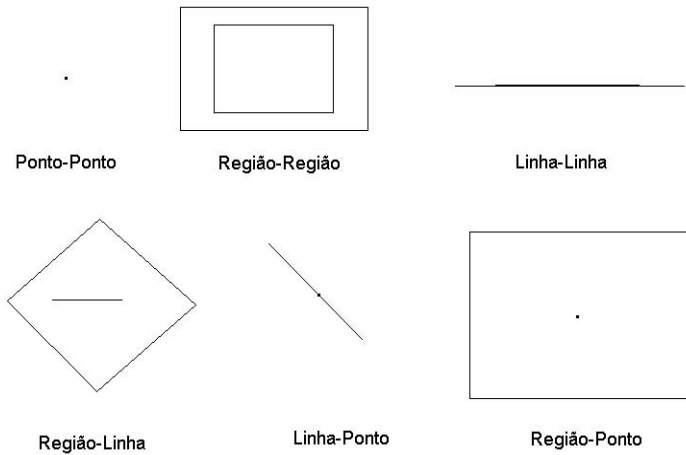
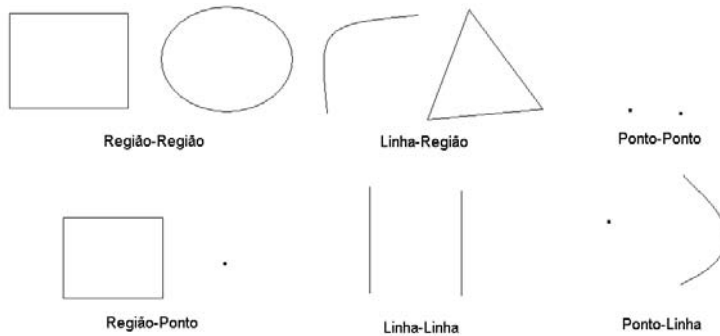
2.3.4 Consultas Espaço-Temporais

Alguns trabalhos feitos na área de consultas geográficas classificam um conjunto básico de consultas espaço-temporais. Em [2], Botelho caracteriza um conjunto de consultas típicas de aplicações geográficas. Essa abordagem permite que o usuário faça três tipos básicos de perguntas.

- quando/onde/o que: descreve o conjunto de fenômenos geográficos (o que) presentes em uma localização ou em um conjunto de localizações (onde), dada uma referência temporal (quando);
- quando/o que/onde: descreve uma localização ou seu conjunto (onde) ocupada por um ou vários fenômenos geográficos (o que) em um dado intervalo de tempo;
- o que/onde/quando: descreve o(s) intervalo(s) de tempo (quando) em que um determinado conjunto de fenômenos geográficos (o que) ocupou um conjunto de localizações.

Ainda em [2], Botelho cita ainda outras classificações de tipos de consultas de um sistema espaço-temporal. Uma delas, é a seguinte:

- Apresentação dos dados armazenados;
- Determinação dos relacionamentos espaciais entre entidades diferentes;
- Simulação e comparação de cenários alternativos;

Figura 2.5: Exemplos de relacionamento *in*.Figura 2.6: Exemplos de relacionamento *disjoint*.

- Previsão do futuro através de análise de tendências.

Há vários trabalhos na literatura sobre aspectos espaço-temporais. Em [16], Erwig et al. propõe uma abordagem onde pontos e regiões móveis são visualizadas como entidades de três dimensões (espaço 2D + tempo) cuja estrutura e comportamento é obtida através da modelagem de tipos abstratos de dados. Estes tipos abstratos de dados podem ser integrados com banco de dados relacionais, orientado a objetos, etc. Em [15], Erwig et al. apresentam um estudo a respeito dos predicados espaço-temporal, no qual é descrito o desenvolvimento dos relacionamentos topológicos espaciais. Baseado nesses relacionamentos um framework foi desenvolvido no qual predicados espaço-temporais são obtidos através da agregação de elementos temporais aos relacionamentos espaciais.

2.4 O modelo de dados geométrico do OGC

O Open Geospatial Consortium(OGC)[5] é um consórcio com mais de 250 companhias, agências governamentais e universidades, criado para promover o desenvolvimento de tecnologias que facilitem a interoperabilidade entre sistemas envolvendo informação espacial e localização. Os produtos do trabalho do OGC são apresentados sob forma de especificações de interfaces e padrões de intercâmbio de dados.

Em maio de 1999, o OGC propôs o Simple Features Specification[5]. O objetivo dessa especificação é definir um esquema padrão SQL que suporte armazenamento, recuperação, consultas e atualizações de features geo-espaciais através da API ODBC. A especificação abstrata do OpenGIS permite atributos espaciais e não-espaciais. Os atributos espaciais são os valores geométricos, e as features são baseadas na geometria 2D com interpolação linear entre os vértices. Para esta dissertação é fundamental o entendimento do modelo de dados proposto em [5]. As funções geométricas usam conceitos de geometria computacional, descritas em [19].

O modelo de dados geométricos do OGC é centrado em uma hierarquia de classes denominada *Geometry*. A figura 2.7 apresenta esta hierarquia. A classe abstrata *Geometry* é a raiz da hierarquia. As 4 subclasses hierárquicas (*Point*, *Curve*, *Surface* e *GeometryCollection*) são instanciáveis e definem se um determinado objeto geométrico possui 0, 1 ou 2-dimensões. As três primeiras se referem a objetos isolados, enquanto *Geometry Collection* se refere a objetos cuja geometria é composta por um conjunto de objetos geométricos. O OGC propõe uma série de métodos para cada classe integrante da hierarquia. Para a implementação do estudo de caso desta dissertação serão necessárias as classes: *Point*, *Line* e *Polygon*.

A classe raiz *Geometry* apresenta alguns métodos que são classificados em três diferentes tipos: básicos, relacionamentos espaciais e análise espacial.

Métodos Básicos

- *GeometryType*(*Geometry A*): Retorna uma string com o nome da sub-classe instanciada da classe *Geometry*.
- *SRID*(*Geometry A*): Retorna um inteiro que é o ID do Sistema Espacial de Referência para um objeto *Geometry*.
- *Envelope*: Retorna um objeto *Geometry* que é o MBB para uma dada geometria. O MBB é o minimum boundary box, o polígono definido pelo quadrante ((*MINX*, *MINY*), (*MAXX*, *MINY*), (*MAXX*,*MAXY*), (*MINX*, *MAXY*), (*MINX*, *MINY*)).

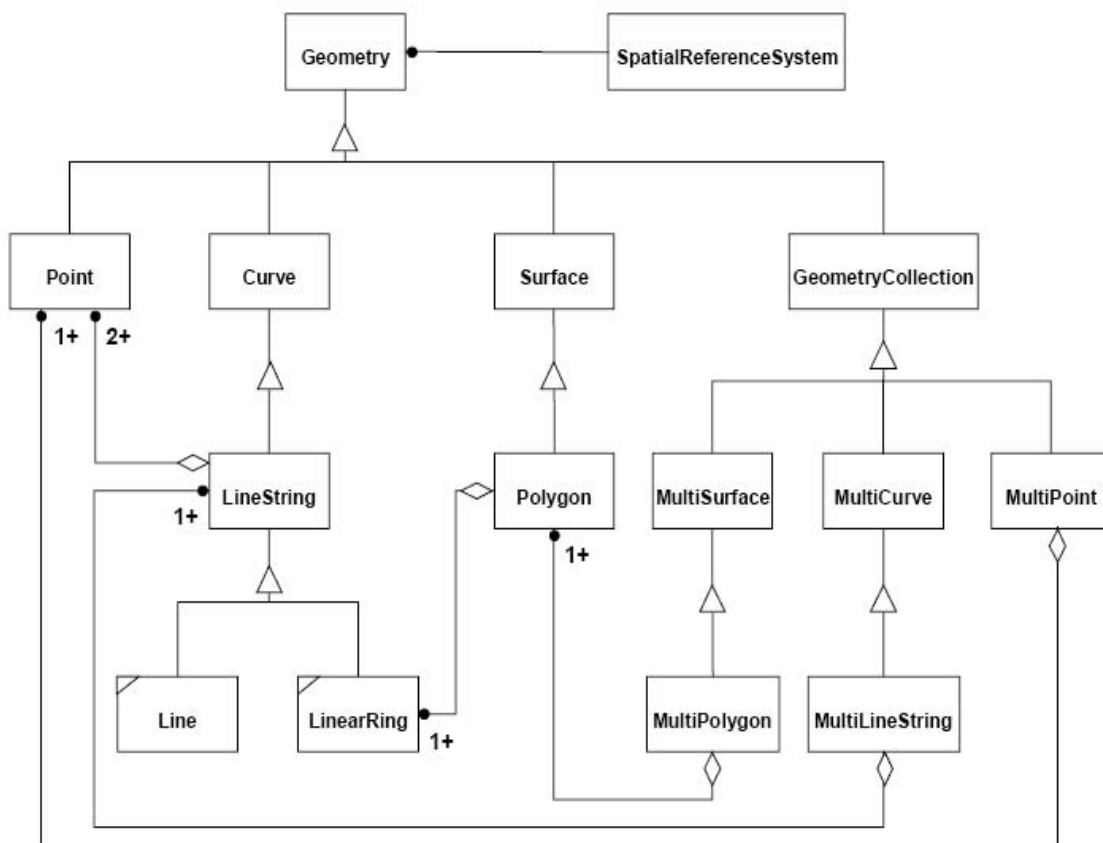


Figura 2.7: Modelo de dados geométricos proposto pelo OGC.

- `AsText(Geometry A)`: Retorna uma string, que é uma representação textual conhecida para um objeto `Geometry`.
- `AsBinary(Geometry A)`: Retorna um binário, que é uma representação binária conhecida para um objeto `Geometry`.
- `IsEmpty(Geometry A)`: Retorna um inteiro, 1 (Verdadeiro) se o objeto `Geometry` é uma geometria vazia no sistema cartesiano de coordenadas
- `IsSimple(Geometry A)`: Retorna um inteiro, 1 (Verdadeiro) se o objeto `Geometry` não tiver nenhum ponto geométrico anômalo, tal qual uma auto-intersecção ou uma auto-tangência.
- `Boundary(Geometry A)`: Retorna um objeto do tipo `Geometry` que é o fecho do limite combinatorial do objeto `Geometry`.

Relacionamentos Espaciais

Os relacionamentos espaciais são baseados nos conceitos da seção 2 e retornam 1(Verdadeiro) se o predicado é verdadeiro.

- `Equals(Geometry A, Geometry B)`: É o operador de igualdade. Verifica se as geometrias dadas são iguais.
- `Disjoint(Geometry A, Geometry B)`: Retorna 1 (Verdadeiro) se as geometrias dadas são espacialmente disjuntas.
- `Intersects(Geometry A, Geometry B)`: Verifica se existe intersecção entre as geometrias dadas.
- `Touches(Geometry A, Geometry B)`: Verifica se existe pelo menos um ponto em comum entre as geometrias dadas.
- `Crosses(Geometry A, Geometry B)`: Verifica se as geometrias dadas se cruzam espacialmente.
- `Within(Geometry A, Geometry B)`: Verifica se a geometria A contém a geometria B.
- `Contains(Geometry A, Geometry B)`: Verifica se A está contida em B. É a operação inversa da operação `Within`.
- `Overlaps(Geometry A, Geometry B)`: Verifica se existe sobreposição entre as geometrias dadas.
- `Relate(Geometry A, Geometry B)`: Recebe como parâmetros um objeto do tipo `Geometry` e uma matriz 9X9, verificando se a matriz 9X9 resultante das operações `Interior`, `Boundary` e `Exterior` entre os objetos `Geometry` é igual à matriz 9X9 passada como parâmetro.

Análise Espacial

As funções de análise espacial são aplicadas a geometrias.

- `Distance(Geometry A, Geometry B)`: Retorna a distância cartesiana entre duas geometrias.
- `Buffer(Geometry A, Geometry B, Double x)`: Recebe como parâmetros um objeto do tipo `Geometry` e um número real e retorna o objeto do tipo `Geometry` que representa todos os pontos cuja distância entre os dois objetos envolvidos é menor ou igual ao número real passado por parâmetro.

- `ConvexHull(Geometry A)`: Retorna um objeto do tipo `Geometry` que é a casca convexa do objeto `A`.
- `Intersection(Geometry A, Geometry B)`; `Union(Geometry A, Geometry B)`; `Difference(Geometry A, Geometry B)`: Retorna um objeto do tipo `Geometry` formado pelo conjunto de pontos resultantes da operação lógica intersecção (união, diferença) entre dois objetos do tipo `Geometry`.
- `SymDifference`: Retorna um objeto do tipo `Geometry`, que é a diferença simétrica entre dois objetos do tipo `Geometry`.

Métodos das classes geométricas

Os métodos das classes geométricas são aqueles aplicados a Ponto, Linha e Polígono.

Os métodos pertencentes à classe `Point` são:

- `X`: Retorna um número real que é o valor da coordenada `x` para este ponto.
- `Y`: Retorna um número real que é o valor da coordenada `y` para este ponto.

Os métodos pertencentes à classe `Line` podem ser divididos em dois grupos: herdados da classe `Curve` e métodos próprios.

Os métodos herdados da classe `Curve` são:

- `Length(Geometry A, Geometry B)`: Retorna um real que é o comprimento do objeto do tipo `Curve`.
- `StartPoint(Geometry A)`; `EndPoint(Geometry A)`: Retorna um objeto do tipo `Point` que é o ponto inicial (final) do objeto do tipo `Curve`.
- `IsClosed(Geometry A)`: Retorna um inteiro, 1 (Verdadeiro) se o objeto do tipo `Curve` é fechado (`StartPoint () = EndPoint ()`).
- `IsRing(Geometry A)`: Retorna um inteiro, 1 (Verdadeiro) se o objeto do tipo `Curve` é fechado e simples (não passa mais de uma vez pelo mesmo ponto).

Os métodos da classe `Line` são:

- `NumPoints(Geometry A)`: Retorna um inteiro que é o número de pontos do objeto do tipo `Line`.
- `PointN(Geometry A, integer N)`: Retorna um objeto do tipo `Point` que é o `N`ésimo ponto do objeto do tipo `Line`.

Os métodos pertencentes à classe Polygon podem ser divididos em dois grupos: herdados da classe Surface e métodos próprios.

Os métodos herdados da classe Surface são:

- Area(Geometry A): Retorna um número real que é a área da geometria dada.
- Centroid(Geometry A): Retorna um objeto do tipo Point que o centro matemático do objeto do tipo Surface. Este ponto não necessariamente se encontra inserido na superfície em questão.
- PointOnSurface(Geometry A): Retorna um objeto do tipo Point que necessariamente se encontra inserido na superfície em questão.

Os métodos da classe Polygon são:

- ExteriorRing(Geometry A): Retorna um objeto do tipo LineString que o anel exterior para um objeto do tipo Polygon.
- NumInteriorRing(Geometry A): Retorna um inteiro que é número de anéis interiores para um objeto do tipo Polygon.
- InteriorRingN(Geometry A, integer N): Retorna um objeto do tipo LineString que é o enésimo anel interior de um determinado tipo Polygon.

2.5 PostGis

O PostGis é uma extensão do SGBD PostgreSQL que provê suporte a dados geográficos. O PostGis implementa o modelo de dados proposto pelo OGC[5], incluindo todas as suas funcionalidades apresentadas em [5]. Além disso, o PostGis estende o modelo de dados proposto adicionando suporte a dados geográficos com coordenadas 3D e 4D. As funções implementadas no PostGis[10] e que foram especificadas pelo OGC podem ser agrupadas em 7 tipos: Funções de Gerenciamento, Funções métricas, Funções de representação geométrica, Saídas geométrica, Funções para referenciamento linear, Funções para suporte a transações longas e Outras. Utilizam Operadores espaciais específicos e Construtores Geométricos.

O PostGis manipula vários tipos de representação geométrica, usando padrões OGC e outros. Exemplos de padrões usados são WKB e WKT. Ambos, WKT e WKB, incluem informação sobre o tipo do objeto e as coordenadas a qual forma o objeto. Exemplos de operadores usando esses padrões são:

- POINT(0 0)

Representa o ponto de coordenadas $x = 0, y = 0$, no plano cartesiano.

- POLYGON(0 0,4 0,4 4,0 4,0 0)

Representa o polígono de quatro arestas cujas coordenadas são: $p1(x = 0, y = 0), p2(x = 4, y = 0), p3(x = 4, y = 4)$ e $p4(x = 0, y = 4)$.

A seguir são descritos exemplos de funções disponíveis no PostGIS. Outras funções estão descritas e detalhadas em [10].

2.5.1 Operadores espaciais

Os operadores específicos operam em cima dos retângulos envolventes de dois objetos geométricos e servem de base para análise espacial. Esses operadores retornam verdadeiro ou falso para os relacionamentos espaciais entre os retângulos. A sintaxe de utilização desses operadores é: $A < op > B$, onde A e B são os objetos geométricos em questão e $< op >$ é o operador desejado. Este grupo é formado pelos seguintes operadores:

$<$ retorna verdadeiro se o retângulo envolvente de A sobrepuser ou estiver à esquerda do retângulo envolvente de B.

$>$ retorna verdadeiro se o retângulo envolvente de A sobrepuser ou estiver à direita do retângulo envolvente de B.

$<<$ retorna verdadeiro se o retângulo envolvente de A estiver estritamente à esquerda do retângulo envolvente de B.

$>>$ retorna verdadeiro se o retângulo envolvente de A estiver estritamente à direita do retângulo envolvente de B.

$< |$ retorna verdadeiro se o retângulo envolvente de A estiver abaixo do retângulo envolvente de B.

$| >$ retorna verdadeiro se o retângulo envolvente de A estiver acima do retângulo envolvente de B.

$<< |$ retorna verdadeiro se o retângulo envolvente de A estiver estritamente abaixo do retângulo envolvente de B.

$| >>$ retorna verdadeiro se o retângulo envolvente de A estiver estritamente acima do retângulo envolvente de B.

$=$ é o operador de igualdade, testa vértice por vértice do retângulo envolvente de A e B e retorna verdadeiro se os mesmos forem iguais.

@: retorna verdadeiro se o retângulo envolvente de A estiver completamente contido no retângulo envolvente de B.

2.5.2 Construtores geométricos

As funções enquadradas neste grupo têm como funcionalidades: adicionar ou remover objetos geométricos, alterar o número de coordenadas de um objeto, trabalhar com a projeção de objetos. São exemplos de construtores geométricos:

GeomFromEWKT(text): Retorna um objeto geométrico a partir do formato texto.

LineFromMultiPoint(multipoint): Retorna um objeto geométrico LineString a partir de um objeto geométrico do tipo MultiPoint.

2.5.3 Funções métricas

As funções enquadradas nesse grupo realizam operações de medidas em dados geográficos. São exemplos de funções métricas:

area2d(geometry): Retorna a área de um polígono ou multi-polígono.

distance(geometry, geometry): Retorna a menor distância entre os objetos geométricos a partir do centróide.

perimeter(geometry): Retorna o perímetro de um polígono ou multi-polígono.

2.5.4 Saídas geométricas

As funções enquadradas neste grupo recebem como parâmetros de entrada: objetos geométricos, uma coleção de objetos geométricos, dados textuais ou pontos e retornam objetos geométricos em algum tipo de representação por exemplo: EWKT, EWKB, etc. Um exemplo de saída geométrica é:

AsEWKT(geometry): Retorna um objeto geométrico no formato texto.

2.5.5 Funções para referenciamento linear

As funções enquadradas neste grupo combinam objetos geométricos com números reais para obtenção de novos objetos geométricos. Um exemplo de função de referenciamento linear é:

lineinterpolatepoint(linestring, location): Retorna uma interpolação de pontos ao longo de um objeto geométrico do tipo line.

2.5.6 Funções de representação geométrica

As funções enquadradas nesse grupo realizam operações geométricas em dados geográficos. Um exemplo de função de representação geométrica é:

AddBBOX(geometry): Retorna o mínimo retângulo envolvente do objeto geométrico.

2.5.7 Outras funções

Outras funções não se enquadram em nenhum grupo específico, fazendo parte do grupo “MISC”. Exemplos são as que realizam operações em objetos geométricos tais como: cálculo do número de dimensões, cálculo do número de anéis, verificação da validade de um objeto geométrico, etc. São exemplos de funções Misc:

npoints(geometry): Retorna o número de pontos do objeto geométrico.

ndims(geometry): Retorna um inteiro representando o número das dimensões da geometria. Os valores possíveis são: 2,3 ou 4.

2.5.8 Funções de Gerenciamento

As funções enquadradas neste grupo têm como objetivo auxiliar o gerenciamento do banco de dados. São exemplos de funções de gerenciamento:

DropGeometryTable(schema_name, table_name) : Destrói uma tabela e todas suas referências nas colunas geométricas.

postgis_version(): Retorna o número de versão das funções de PostGIS instaladas neste banco de dados.

2.5.9 Funções para suporte a transações longas

As funções deste grupo auxiliam a realização de transações longas. São exemplos de funções para suporte a transações longas: do número de anéis, verifica a validade de um objeto geométrico, etc. São exemplos de funções:

EnableLongTransactions(): Habilita o suporte a funções de transações longas.

UnlockRows(authid): Remove todos os locks efetuados por um determinado authid.

2.5.10 Usando dados geográficos no PostGis

O OpenGIS define tipos padrão de objetos geográficos, as funções requeridas para sua manipulação e uma especificação de uma tabela de metadados. A criação de qualquer tabela espacial precisa ser feita em duas etapas: criação da tabela e adição das colunas dos atributos espaciais. Para assegurar que os metadados permaneçam consistentes, operações como criar e remover uma coluna espacial são carregadas fora através de procedimentos especiais.

Existem duas tabelas de metadados especificadas pelo OpenGIS e implementadas no PostGis: *SPATIAL_REF_SYS* e *GEOMETRY_COLUMNS*.

SPATIAL_REF_SYS

É a tabela onde são armazenados os identificadores numéricos e as descrições textuais dos sistemas de coordenadas utilizados no banco de dados espacial. Compõem esta tabela os seguintes atributos: *SRID*, *AUTH_NAME*, *AUTH_SRID*, *SRTEXT* e *PROJ4TEXT*. Esta tabela é a implementação do sistema de referência espacial. Um sistema de referência espacial é um conjunto de parâmetros que inclui:

- O nome do sistema de coordenadas a partir do qual as coordenadas são derivadas, armazenado no atributo *AUTH_NAME*.
- O identificador numérico que identifica exclusivamente o sistema de referência espacial, armazenado no atributo *AUTH_SRID*.
- Coordenadas que definem a máxima extensão possível de espaço referido por um determinado intervalo de coordenadas, armazenado no atributo *SRTEXT*.
- Números que, quando aplicados em certas operações matemáticas, convertem coordenadas recebidas como entrada em valores que podem ser processados com eficiência máxima, armazenados no atributo *PROJ4TEXT*.

O atributo *SRID* é o responsável por identificar unicamente um sistema de referência espacial no PostGis. O PostGis permite que qualquer usuário defina um sistema de referência espacial e disponibiliza aproximadamente 2500 sistemas de referência espacial.

GEOMETRY_COLUMNS

É a tabela onde responsável pelas colunas geométricas implementadas em um banco de dados. Compõem esta tabela os seguintes atributos:

- *F_TABLE_CATALOG*: É um atributo do tipo cadeia de caracteres definido pelo OGC mas não utilizado no SGBD PostgreSQL. Todas as tuplas desta tabela recebem valor “vazio”.
- *F_TABLE_SCHEMA*: É o atributo que armazena o nome de banco de dados no PostgreSQL. O valor padrão é public.
- *F_TABLE_NAME*: É o atributo que armazena o nome da tabela que contém a coluna geométrica.
- *F_GEOMETRY_COLUMN*: É o atributo que armazena o nome da coluna geométrica.
- *COORD_DIMENSION*: É o atributo que armazena o número de dimensões por ponto.
- *SRID*: É o atributo que identifica o sistema de referência espacial, não é necessário utilizar o mesmo sistema para colunas geométricas de um mesmo banco.
- *TYPE*: É o tipo de geométrico que a coluna irá armazenar. Os tipos permitidos estão descritos na seção 2.4.

2.6 Conclusões

Este capítulo apresenta alguns conceitos associados à dissertação e apresenta o OGC e o PostGis, que serão usados na implementação do protótipo

Capítulo 3

Modelo de dados proposto

Este capítulo apresenta o modelo de dados proposto, seu mapeamento para o modelo relacional e exemplos de consultas permitidas.

3.1 Características da Aplicação

O objetivo do projeto WebMAPS é atender a usuários de diferentes tipos (desde cooperativas até usuários individuais) que desejem usar dados associados à produção agrícola para tomada de decisões. A modelagem do banco de dados para o projeto foi realizada a partir das necessidades constatadas (perfis de consultas) e dos tipos de dados disponibilizados. Esta modelagem fez parte de um processo de várias reuniões com usuários e participantes do projeto, dentro de uma metodologia descrita em [20].

Os dados básicos a serem usados na primeira versão do projeto são: imagens de satélite, caracterizadas pela região coberta, dispositivo e sensor usado e data de captura; informações cadastrais fornecidas por usuários (propriedades rurais e seus talhões, com as coordenadas associadas) e culturas associadas aos talhões. Todos esses dados podem variar no tempo, caracterizando assim um banco de dados espaço-temporal.

Em mais detalhes, todos os atributos de uma propriedade podem variar no tempo - como o seu nome - mas o fator complicador para efeito do trabalho é que a sua caracterização espacial também pode variar. Da mesma forma, talhões podem ter sua geometria modificada ao longo do tempo; além disso, podem sofrer rotação de cultura (atributo não espacial). Para o domínio alvo do projeto, talhões têm nome e, na maior parte das vezes, o nome está associado a uma geometria específica. Caso haja grandes variações desta geometria, o talhão deixa de existir e novo talhão é criado.

O banco de dados deveria também armazenar imagens de satélite para permitir consultas diretamente às imagens - por exemplo, variação do conteúdo (valores de pixels) dentro de uma região ao longo do tempo. Isto significaria, no entanto, desenvolver todo um con-

junto de primitivas em cima do SGBD para acesso a regiões de imagens. Optou-se, neste caso, por realizar estas operações externamente ao SGBD, ficando as tabelas restritas a cadastrar imagens e regiões de imagens correspondentes a propriedades ou talhões, via o conceito de MBB (minimum bounding box) e máscara (usada pelos algoritmos externos de processamento dos valores de pixels das regiões delimitada pelo MBB).

Além das necessidades dos usuários do ponto de vista semântico, o sistema deve prever possibilidade de controlar os diferentes perfis de usuários - restrição a acesso, operações autorizadas e perfis de consultas. Como se verá na próxima seção, uma parte da modelagem do banco de dados foi dedicada a estas considerações.

Dadas estas restrições, exemplos de consultas típicas de usuários são:

1. Consultas de usuário com perfil de pesquisador (voltadas à tomada de decisão para por exemplo definição de novas funcionalidades ou estudos econômicos).
 - Quais as últimas 10 consultas relativas à região C.
 - Quais as consultas efetuadas voltadas à cultura Y.
 - Quais as consultas realizadas na data Z.
 - Qual a cultura mais freqüentemente consultada nos últimos x anos.
 - Recuperação da evolução do plantio nas propriedades de uma região para estudo comparativo
 - Dado um perfil de evolução de valores de pixels de uma região, que outras regiões cadastradas apresentam o mesmo perfil.
 - Consultas comuns aos outros tipos de usuários.
2. Consultas do usuário final (voltadas à tomada de decisão para plantio)
 - Qual a evolução temporal dos valores de pixels de uma certa região.
 - Qual a evolução espacial de uma propriedade ou de um talhão.
 - Quais as culturas plantadas em uma propriedade ou talhão em um intervalo de tempo.
 - Características espaciais ou temporais de uma propriedade ou talhão.
3. Consultas de administradores de sistemas
 - Quais os últimos 10 usuários cadastrados.
 - Quais os usuários que mais freqüentemente fazem consultas ao sistema.

- Quais as permissões de um determinado usuário.
- Outras consultas para gerenciamento e desempenho do sistema, incluindo acesso aos logs do banco de dados.

As próximas seções descrevem o modelo do banco de dados especificado para atender a estes dados e perfil de uso e mostram como o modelo se adequa aos principais tipos de consulta.

3.2 Modelo de dados

A figura 3.1 mostra o diagrama ER do banco de dados criado, para cadastramento de usuários, propriedades e talhões. O diagrama apresentado na figura pode ser dividido em dois grupos: grupo dos dados relacionados às imagens de satélite e o grupo dos dados referentes às propriedades agrícolas.

O grupo dos dados relacionados às imagens de satélite é formado pelas entidades Image Mask, Image MBB e Satellite Image. O MBB (minimum bounding box) delimita, para uma imagem, a região de interesse - uma propriedade ou talhão. Isto visa o acesso rápido às áreas de cada propriedade, durante o processamento de consultas, evitando perda de tempo neste tipo de cálculo. A máscara (Mask) define quais valores de pixels do MBB serão utilizados para o o processamento desejado pelo usuário. Máscara e MBB são usadas para todas as imagens do banco de dados, armazenadas na entidade Satellite Image, sendo a forma encontrada para otimizar as consultas por região a partir da imagem. Cada MBB corresponde a uma propriedade ou a um talhão. As entidades associadas às imagens têm um atributo *path*, que armazena o nome do arquivo onde os dados (recortes de imagens) estão armazenados.

O grupo dos dados referentes às propriedades agrícolas está centrado na noção de Propriedade (Property) composto por Talhões (parcel). Também fazem parte deste grupo as entidades User, Query e crop. A entidade usuário(User) representa os usuários do sistema: pesquisadores, usuários-alvo ou administradores. Usuários podem cadastrar propriedades e, eventualmente, disponibilizar esses dados para outros usuários. A entidade Query armazena a especificação das consultas realizadas por um usuário em uma propriedade ou talhão. A entidade crop representa as culturas cultivadas em um determinado talhão, variável ao longo do tempo. Desta forma, o relacionamento crop-parcel é um relacionamento temporal. As entidades Property e parcel possuem o atributo *coordinates*, responsável por armazenar a geometria da propriedade/talhão, a qual é, também, variável ao longo do tempo.

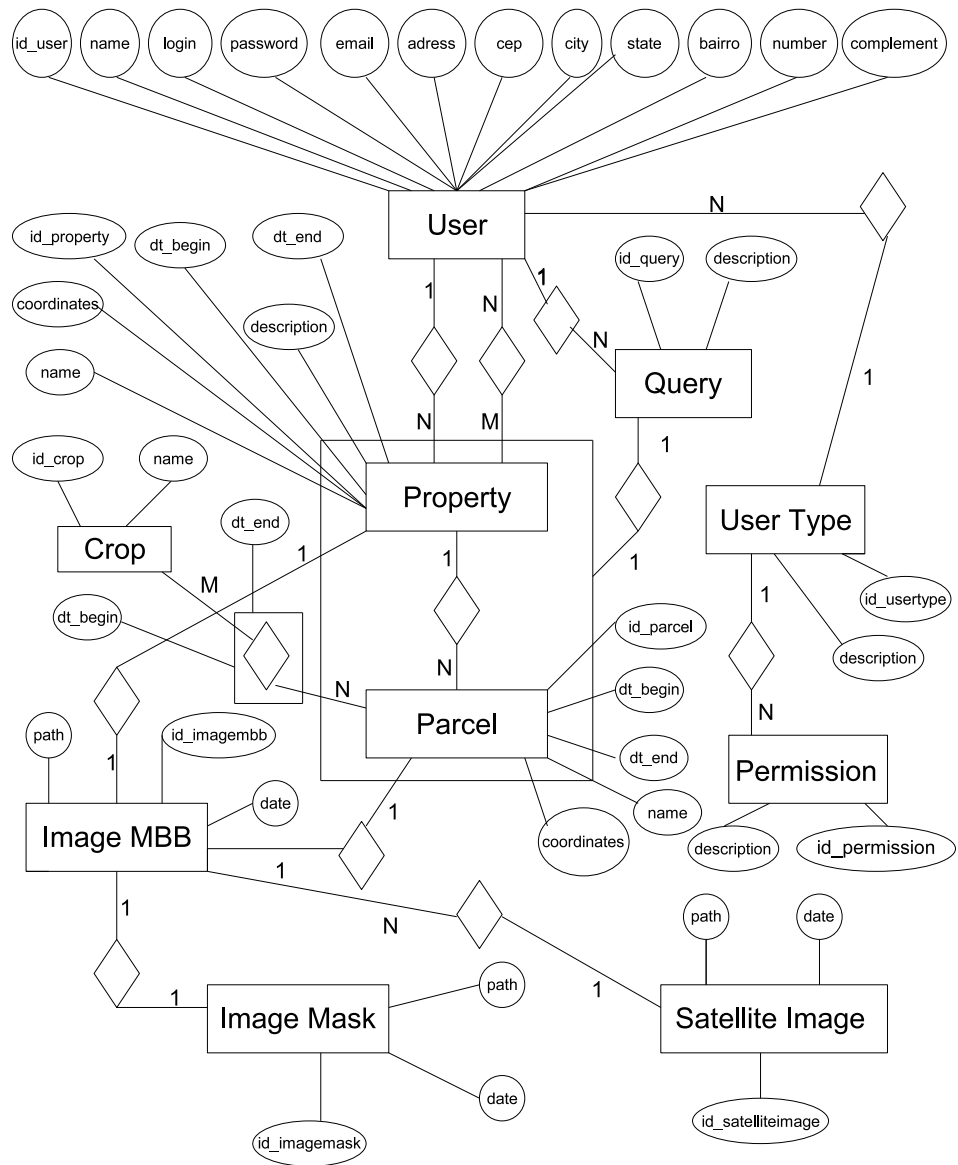


Figura 3.1: Modelo Entidade-Relacionamento.

3.3 Esquema Relacional

O diagrama apresentado na figura deu origem ao seguinte esquema relacional:

- User(id_user, name, login, password, email, address, number, complement, bairro, cep, city, state, id_usertype)

Esta tabela armazena as informações referentes aos usuários do sistema. Os campos: *address*, *number*, *complement*, *bairro*, *cep*, *city* e *state* guardam informações a respeito do endereço dos usuários, os campos *name*, *login*, *password* e *email* armazenam informações pessoais dos usuários, o campo *id_usertype* identifica o tipo de usuário (chave estrangeira) e o campo *id_user* identifica o usuário univocamente.

- Usertype(id_usertype, description)

A tabela Usertype armazena as informações referentes aos tipos de usuário do sistema. O campo *description* armazena uma descrição textual sobre o tipo de usuário do sistema. Os tipos de usuário previstos atualmente são: agricultores, pesquisadores, administradores e cooperativas. O campo *id_usertype* é a chave primária.

- Query(id_query, description, dt_query, id_user, id_property, dt_begin_property, id_parcel, dt_begin_parcel)

A tabela Query armazena as consultas que um determinado usuário realiza no sistema. O campo *id_query* é a chave primária da relação. Os campos *id_user*, *id_property*, *dt_begin_property*, *id_parcel* e *dt_begin_parcel* identificam o usuário que realizou a consulta e qual propriedade e talhão foi o alvo da consulta. Os campos *description* e *dt_query* armazenam respectivamente uma descrição da consulta, fornecida pelo próprio usuário, e a data em que foi realizada a consulta. O objetivo desta relação é permitir estudar estatísticas sobre perfis de consulta, visando posterior otimização.

- Property(id_property, dt_begin, dt_end, coordinates, name, description, id_user, id_imagembb)

A tabela Property armazena as informações referentes às propriedades (fazendas) cadastradas no sistema. *Name* e *description* armazenam respectivamente o nome e uma descrição da propriedade fornecido pelo usuário que a cadastrou, o campo *coordinates* armazena as coordenadas geográficas da localização da propriedade, podendo estas variar ao longo do tempo. Os campos *dt_begin* e *dt_end* definem o período para o qual as demais informações são válidas, *id_property* e *dt_begin* são as chaves primárias desta relação e o campo *id_user* identifica o usuário que cadastrou os dados correspondentes.

- Parcel(id_parcel, dt_begin, dt_end, coordinates, name, id_property, dt_begin_property, id_imagembb)

A tabela Parcel armazena as informações referentes aos talhões de uma propriedade. *Name* armazena o nome do talhão, o campo *coordinates* armazena as coordenadas geográficas do talhão, podendo estas variar ao longo do tempo. Os campos *dt_begin* e *dt_end* definem o período para o qual as informações de um talhão são válidas; *id_parcel* e *dt_begin* são as chaves primárias desta relação enquanto *id_property* e *dt_begin_property* identificam a fazenda onde está localizado o talhão (chave estrangeira).

- Crop(id_crop, name)

A tabela Crop armazena as informações referentes às culturas agrícolas, *Name* armazena o nome das culturas cadastradas e o campo *id_crop* é a chave primária desta relação. Esta tabela é gerenciada pelos administradores do sistema, que determinam as culturas válidas.

- ImageMBB(id_imagembb, date, path, id_satelliteimage)

A tabela ImageMBB armazena o MBB (minimum bounding rectangle) referente a um determinado talhão ou propriedade, sendo usada para processar polígonos em imagens de satélite. O campo *id_imagembb* identifica a tupla, o campo *date* armazena a data em que o mbb foi gerado, o campo *path* o caminho onde se encontra o arquivo correspondente e o campo *id_satelliteimage* a qual imagem o mbb está relacionado.

- ImageMask(id_imagemmask, path, date, id_imagembb)

A tabela ImageMask armazena a máscara referente a um determinado MBB em uma dada imagem de satélite. O campo *id_imagemmask* identifica uma máscara, o campo *date* armazena a data em que essa máscara foi gerada, o campo *path* o caminho onde se encontra o arquivo da máscara gerado e o campo *id_imagembb* a qual MBB a máscara gerada corresponde. A máscara é gerada a partir das coordenadas de uma propriedade/talhão. Estas coordenadas fornecem a latitude e longitude mínimas (bottom) e máximas (top) do retângulo mínimo envolvente de uma propriedade(talhão). Sendo assim a máscara é uma imagem retangular binária que envolve a propriedade toda do usuário, onde os valores 0 são valores de pixels fora da propriedade e os valores 1 são valores de pixels dentro da propriedade(talhão).

- SatelliteImage(id_satelliteimage, path, date)

A tabela SatelliteImage armazena informações referentes às imagens de satélite. O campo *id_satelliteimage* identifica uma imagem, o campo *date* identifica a data em que essa imagem foi armazenada, o campo *path* o caminho onde se encontra o arquivo da imagem de satélite.

- UserProperty(id_user, id_property, dt_begin)

A tabela UserProperty representa o relacionamento m:n entre as entidades User e Property. O usuário que cadastra uma propriedade pode disponibilizar estes dados para um outro usuário cadastrado acessá-los. O relacionamento identifica desta forma quais usuários podem acessar dados de uma propriedade.

- ParcelCrop(id_crop, id_parcel, dt_begin_parcel, dt_begin_crop, dt_end_crop)

A tabela ParcelCrop representa o relacionamento m:n entre as entidades Parcel e Crop, sendo este um relacionamento temporal em que determinado talhão - identificado pelos campos *id_parcel* e *dt_begin_parcel* - pode ter mais de uma determinada cultura ao longo do tempo - identificada pelo campo *id_crop* . Os campos *dt_begin_crop* e *dt_end_crop* determinam o intervalo em que a cultura esteve associada ao talhão.

- PropertyParcel(id_property, dt_begin_property, id_parcel, dt_begin_parcel)

A tabela PropertyParcel representa a agregação propriedade e talhão. Os atributos são as chaves primárias das tabelas Property e Parcel. Esta tabela corresponde à agregação, havendo sido mantida para facilitar consultas. Normalmente, não seria necessária pois a chave de Property é chave estrangeira de Parcel.

- Permission(id_permission, description, id_usertype)

A tabela Permission armazena as informações referentes às permissões do sistema, *description* armazena as permissões do sistema e o campo *id_permission* é a chave primária desta relação. O campo *id_usertype* define quais tipos de usuário recebem uma determinada permissão. Esta tabela é gerenciada pelos administradores do sistema, que determinam as permissões válidas.

3.4 Consultas temporais

As consultas temporais são caracterizadas por predicados temporais. Podem ser divididas em: consultas que retornam valores de tempo, consultas que retornam tuplas satisfazendo a um predicado temporal e consultas que retornam estados temporais [11]. Para todas as consultas, supõe-se que o estado atual é aquele em que *dt_end*=NULL. A notação usada é cálculo relacional[9].

3.4.1 Consultas que retornam valores de tempo.

- Quando a fazenda “X” começou a existir.

$\min(p.dt_begin) | p \in \text{Property} \wedge p.name = \text{“X”}$

3.4.2 Consultas que retornam tuplas satisfazendo a um predicado temporal.

- Quais fazendas existiam em 2002.

$$p | p \in \text{Property} \wedge p.\text{dt_begin} < \text{"01/01/2003"} \wedge \\ (p.\text{dt_end} > \text{"31/12/2001"} \vee p.\text{dt_end} = \text{"NULL"})$$

3.4.3 Consultas que retornam estados temporais.

- Retorne a evolução ao longo do tempo das culturas cultivadas no talhão “A” da fazenda “X”.

$$c.\text{name}, cp.\text{dt_begin}, cp.\text{dt_end} | c \in \text{crop} \wedge cp \in \text{ParcelCrop} \wedge \\ \exists p, y (p \in \text{Parcel} \wedge y \in \text{Property} \wedge y.\text{name} = \text{"X"} \\ \wedge p.\text{name} = \text{"A"} \wedge p.\text{id_property} = cp.\text{id_property} \\ \wedge c.\text{id_crop} = cp.\text{id_crop})$$

3.5 Consultas espaciais

As consultas espaciais são caracterizadas por predicados espaciais. Podem ser divididas em: consultas que retornam componentes espaciais, consultas que retornam tuplas satisfazendo a um predicado espacial e consultas que retornam valores métricos obtidos através de predicados temporais [11]. Supõe-se para estas consultas que o valor do componente temporal é o instante atual.

3.5.1 Consultas que retornam componentes espaciais.

- Retorne as coordenadas atuais da fazenda “X”

$$p.\text{coordinates} | p \in \text{Property} \wedge \text{Property.name} = \text{"X"} \\ \wedge \text{Property.dt_end} = \text{"NULL"}$$

3.5.2 Consultas que retornam tuplas.

- Dado o polígono “W”, retorne todos os atributos das regiões que se encontram nesse espaço.

$$p | p \in \text{Property} \wedge p.\text{dt_end} = \text{"NULL"} \wedge \text{within}((0 \ 0, 0 \ 100, 100 \ 0, 100 \ 100), \\ p.\text{coordinates})$$

3.5.3 Consultas que retornam valores(lógicos ou numéricos).

- Qual a distância entre as fazendas “X” e “Y”.

$distance(p.coordinates,y.coordinates)|p \in Property \wedge y \in Property \wedge p.name = \text{“X”} \wedge y.name = \text{“Y”}$

- Retorne a área atual de todas as fazendas.

$Area(p.coordinates)| p \in Property \wedge p.date_end = \text{“NULL”}$

3.6 Consultas espaço-temporais

As consultas espaço-temporais se caracterizam por predicados espaço-temporais. Podem ser divididas em: consultas que retornam componentes espaciais em um determinado intervalo de tempo, consultas que retornam tuplas satisfazendo a predicados espaciais em um determinado intervalo de tempo e consultas que retornam valores obtidos através de predicados espaciais e consultas que apresentam projeções de tuplas [11].

3.6.1 Consultas que retornam componentes espaciais.

- Localização atual de talhões que cultivaram cana entre “01/01/2000” e ”01/07/2001”.

$x.coordinates|x \in Parcel \wedge x.dt_begin < \text{“30/06/2001”} \wedge (x.dt_end > \text{“31/12/1999”} \vee x.dt_end = \text{“NULL”}) \wedge \exists c (c \in crop \wedge c.name=\text{“cana”} \wedge c.id_crop = x.id_crop)$

3.6.2 Consultas que retornam tuplas.

- Retorne a evolução espacial da fazenda “X” ao longo do tempo.

$y.coordinates, y.dt_begin, y.dt_end|y \in Property \wedge y.name = \text{“X”}$

- Retorne a evolução espacial dos talhões da fazenda “X” ao longo do tempo.

$p.coordinates, p.dt_begin, p.dt_end |p \in parcel \wedge \exists y (y \in Property \wedge y.name = \text{“Fazenda X”} \wedge p.id_property = y.id_property)$

3.6.3 Consultas que retornam valores métricos.

- Qual o perímetro dos talhões que cultivaram cana no ano 2000.

$$\begin{aligned} & \text{perimeter}(\text{coordinates}(p)) \mid p \in \text{Parcel} \wedge \exists c \\ & (c \in \text{Crop} \wedge c.\text{name} = \text{"Cana"} \wedge cp \in \\ & \text{ParcelCrop} \wedge cp.\text{id_crop} = c.\text{id_crop} \wedge \\ & cp.\text{id_parcel} = p.\text{id_parcel} \wedge cp.\text{dt_begin} < \\ & \text{"01/01/2001"} \wedge (cp.\text{dt_end} > \text{"31/12/1999"} \vee cp.\text{dt_end} = \text{"NULL"}) \end{aligned}$$

- Qual a área ocupada por fazendas entre “01/07/2000” e “01/07/2001”.

$$\begin{aligned} & \text{sum}(\text{area}(\text{coordinates } y)) \mid y \in \text{Property} \wedge y.\text{dt_begin} < \text{"01/07/2001"} \wedge \\ & (x.\text{dt_end} > \text{"01/07/2000"} \vee x.\text{dt_end} = \text{"NULL"}) \end{aligned}$$

3.7 Consultas sobre perfis de usuários, evolução de valores de pixels e consultas armazenadas

As consultas sobre perfis de usuários são basicamente dirigidas às relações `Usertype` e `Permission` e não apresentam grandes dificuldades. As consultas relativas a perfis de consultas executadas exigiriam que a relação `Query` pudesse armazenar atributos como `Cultura`, `Coordenadas` etc. Isto não foi previsto nesta versão do banco de dados, pois exigiria que se implementasse uma camada que interagisse com o processador de consultas para armazenar campos especificados nos predicados de uma consulta.

As consultas envolvendo imagens consideram o valor médio dos valores de pixels dentro de uma máscara. Não são especificáveis via cálculo relacional, sendo processadas por rotinas especialmente desenvolvidas para este fim. Assim sendo, escapam do escopo deste capítulo. No entanto seu processamento envolve a descoberta do caminho em que os arquivos correspondentes estão armazenados, a saber, são do tipo.

- Qual o caminho do MBB da propriedade X.

$$\begin{aligned} & y.\text{path} \mid p \in \text{Property} \wedge p.\text{name} = \text{"Fazenda X"} \\ & \wedge p.\text{dt_end} = \text{"NULL"} \wedge \exists y \in \text{ImageMBB} \\ & \wedge p.\text{id_imagemb} = y.\text{id_imagemb} \end{aligned}$$

3.8 **Resumo**

Este capítulo apresentou o perfil básico das necessidades dos usuários do sistema e especificou o modelo do banco de dados criado para atender tais necessidades, exemplificando através de algumas consultas básicas como o modelo corresponde ao desejado.

O próximo capítulo descreve a implementação realizada para o banco de dados.

Capítulo 4

Implementação

Este capítulo apresenta a arquitetura do WebMaps, algumas consultas implementadas e as dificuldades do projeto.

4.1 Arquitetura do WebMaps

A figura 4.1 apresenta a arquitetura do WebMaps. A arquitetura possui cinco módulos principais: Interface Web, Módulos de Processamento e Controle, Módulos de Processamento de imagens, Banco de Imagens e o Banco de Dados. O módulo de interface web é a interface do sistema, responsável pela interação entre o usuário e o sistema. A implementação deste módulo foi realizada em HTML, JSP e JAVAScript. O módulo de Processamento e Controle é responsável pela comunicação entre as camadas inferiores e a interface. A implementação deste modulo foi realizada em Java. Os Módulos de Processamento de imagens são responsáveis por processar e extrair dados das imagens de satélite. Estes módulos foram implementados em C e a comunicação com os módulos de processamento e controle é feita através de JNI (Java Native Interface). O Banco de Imagens é o módulo responsável por gerenciar os arquivos de imagens de satélite. Atualmente as imagens de satélite estão organizadas em um sistema de diretório (NTFS). Banco de Dados é o módulo responsável por gerenciar os demais dados do sistema. A ligação entre ele e o Banco de Imagens indica que os caminhos dos arquivos está armazenado no Banco de Dados. O foco desta dissertação é a implementação desse módulo, que foi desenvolvido em PostgreSQL/PostGis.

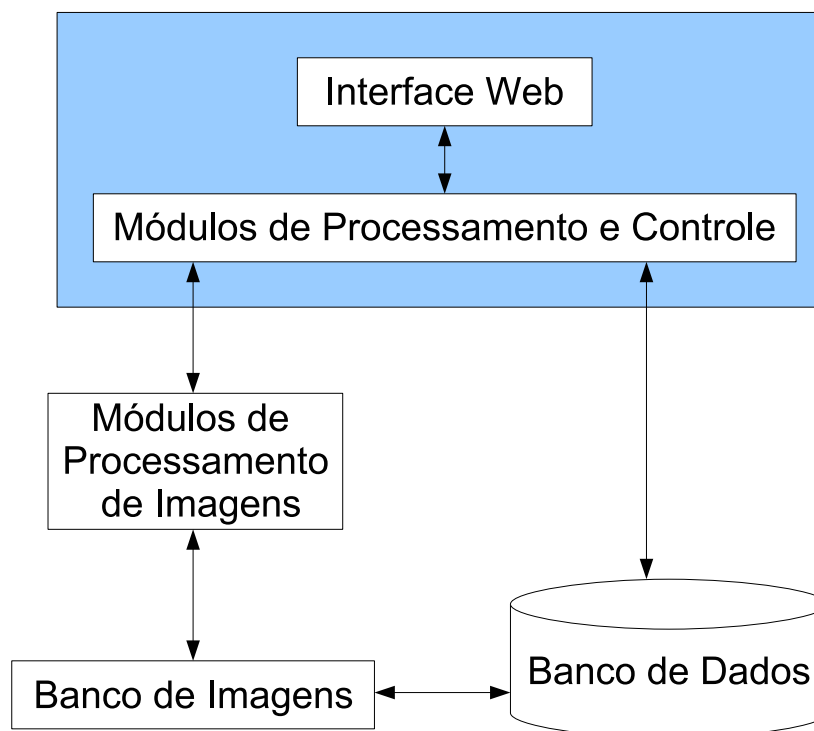


Figura 4.1: Arquitetura WebMaps.

4.2 Funcionamento do Sistema

Esta seção apresenta alguns detalhes da implementação e algumas cópias de telas do sistema.

4.2.1 Alguns detalhes da implementação

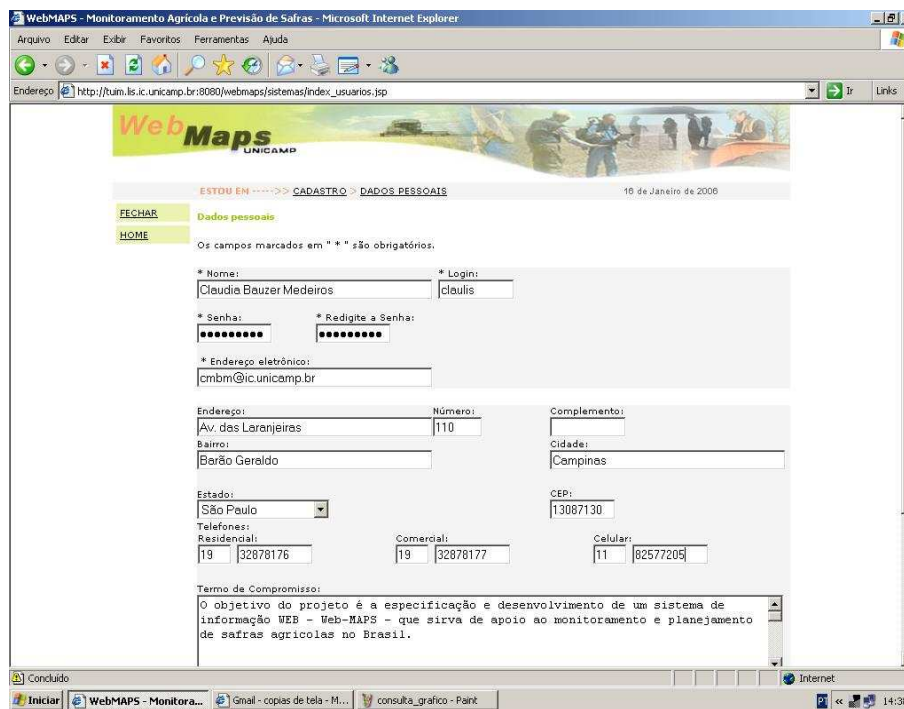
A implementação do banco de dados foi realizada em Postgresql/PostGis. O Postgis permite a inclusão de um campo geométrico em uma tabela. Isto foi realizado em dois passos: criação de uma tabela não espacial e inclusão do campo geométrico. O comando que permite a inclusão do campo geográfico possui a seguinte sintaxe:

- `SELECT AddGeometryColumn('nome da tabela', 'nome do campo geométrico', identificador do campo geométrico, 'tipo da geometria', número de coordenadas);`

No caso da tabela Property, o campo geométrico recebe o nome de *coordinates* sendo um polígono com duas dimensões e o identificador do campo geométrico é 1.

- SELECT AddGeometryColumn('property', 'coordinates', 1, 'POLYGON', 2);

4.2.2 Algumas cópias de tela do protótipo implementado



The screenshot shows a web browser window titled "WebMAPS - Monitoramento Agrícola e Previsão de Safras - Microsoft Internet Explorer". The address bar shows the URL "http://tium.lis.ic.unicamp.br:8060/webmaps/sistemas/index_usuarios.jsp". The page content includes a header with the "Web Maps UNICAMP" logo and a navigation menu with "FECHAR" and "HOME" buttons. The main section is titled "Dados pessoais" and contains a registration form with the following fields and values:

- * Nome: Claudia Bauzer Medeiros
- * Login: claulis
- * Senha: [masked]
- * Redigite a Senha: [masked]
- * Endereço eletrônico: cmbm@ic.unicamp.br
- Endereço: Av. das Laranjeiras
- Número: 110
- Complemento: [empty]
- Bairro: Barão Geraldo
- Cidade: Campinas
- Estado: São Paulo
- CEP: 13087130
- Residencial: 19 32878176
- Comercial: 19 32878177
- Celular: 11 82577205

At the bottom of the form, there is a "Termo de Compromisso:" section with the text: "O objetivo do projeto é a especificação e desenvolvimento de um sistema de informação WEB - Web-MAPS - que sirva de apoio ao monitoramento e planejamento de safras agrícolas no Brasil."

Figura 4.2: Cadastro de usuário.

A figura 4.2 apresenta uma cópia de tela do sistema WebMaps. Esta cópia de tela apresenta o cadastro de um usuário. O usuário preenche uma série de informações tais como: nome, login, senha e dados relativos ao endereço. O usuário interage com o sistema através de uma interface simples, os dados são processados pelos módulos de processamento e controle e então armazenados no banco de dados.

A figura 4.3 apresenta outra cópia de tela do sistema WebMaps. Esta cópia de tela apresenta o cadastro de um talhão. Um usuário já cadastrado no sistema escolhe uma propriedade já cadastrada e preenche os campos relativos ao nome do talhão e as de início e término do talhão. O usuário também anexa um arquivo que contém as coordenadas do talhão. Os módulos de processamento e controle processam esses dados e os enviam ao banco de dados, que irá armazená-los.

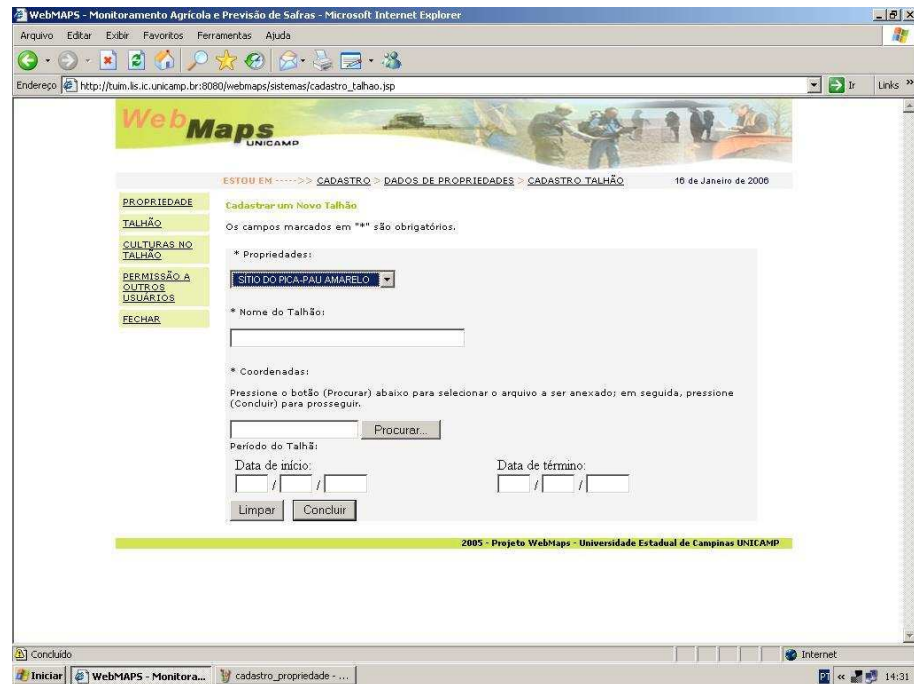


Figura 4.3: Cadastro de talhão.

4.3 Consultas Implementadas

Esta seção apresenta algumas cópias de telas das consultas implementadas.

min (date)
2003-06-28

Figura 4.4: Data inicial da Fazenda Recanto.

A figura 4.4 apresenta o resultado de uma consulta temporal que retorna a data inicial da fazenda “Recanto”. A consulta correspondente é:

```
select min(date_begin_property)
from property
```



```
where name = ‘Fazenda Recanto’
```

A resposta mostra que a fazenda começou a existir no dia 28 de junho de 2003.

date_begin...	date_end (...)	astext (text)	area...
2005-06-28		POLYGON((0 0,0 1,1 1,0 0))	1
2004-06-28	2005-06-28	POLYGON((0 0,0 2,2 2,0 0))	4
2003-06-28	2004-06-28	POLYGON((0 0,0 3,3 3,0 0))	9

Figura 4.5: Localização e área da Fazenda “Recanto” ao longo do tempo.

A figura 4.5 apresenta o resultado de uma consulta espaço-temporal que retorna a localização e a área da fazenda “Recanto”, ao longo do tempo. A consulta correspondente é:

```
select date_begin_property, date_end,
astext(coordinates), area(coordinates)
from property
where name = ‘Fazenda Recanto’
```

A resposta mostra que houve várias modificações nas coordenadas da fazenda e em sua área. No período de 28 de junho de 2003 até 28 de junho de 2004 a fazenda era um polígono fechado (0 0, 0 3, 3 3, 3 0) com área equivalente a 9 metros quadrados. No período de 28 de junho de 2004 até 28 de junho de 2005 a fazenda era um polígono fechado (0 0, 0 2, 2 0, 2 2) com área equivalente a 4 metros quadrados. E finalmente, no período de 28 de junho de 2005 até os dias de hoje a fazenda é um polígono fechado (0 0, 0 1, 1 0, 1 1) com área equivalente a 1 metro quadrado.

A figura 4.6 apresenta o resultado de uma consulta espacial que retorna a distância entre as fazendas “Recanto” e “A” atualmente. A consulta correspondente é:

```
select astext(a.coordinates), astext(b.coordinates),
distance(a.coordinates, b.coordinates)
from property as a, property as b
where a.name = ‘Fazenda Recanto’ and b.name = ‘Fazenda A’
and b.date_end is NULL and a.date_end is NULL
```

astext (text)	astext (text)	distance (float8)
POLYGON((0 0,0 1,1 1,0,0 0))	POLYGON((10 10,10 11,11 11,10 10))	12.7279220613579

Figura 4.6: Distância entre as fazendas “A” e “Recanto”

A resposta mostra que a fazenda “Recanto” é um polígono fechado (0 0, 0 1, 1 0, 1 1), a fazenda “A” é polígono fechado (10 10, 10 11, 11 10, 11 11) e a distância geométrica entre as duas fazenda é de 12.72 metros.

..	id_portion (...)	date_begin...	date_end (...)	name (varc...	astext (text)	perimeter (...)
1		2003-06-28	2004-06-28	Talhão X	POLYGON((0 0,1.5 0,1.5 1.5,0 1.5,0 0))	6
1		2004-06-28	2005-06-28	Talhão X	POLYGON((0 0,1 0,1 1,0 1,0 0))	4
1		2005-06-28		Talhão X	POLYGON((0 0,0 0.5,0.5 0.5,0.5 0,0 0))	2
2		2004-06-28	2005-06-28	Talhão Y	POLYGON((1 1,1.5 1,1.5 1.5,1 1.5,1 1))	2
2		2003-06-28	2004-06-28	Talhão Y	POLYGON((2 2,2 1.5,1.5 1.5,1.5 2,2 2))	2
2		2005-06-28		Talhão Y	POLYGON((1 1,1 0.5,0.5 0.5,0.5 1,1 1))	2

Figura 4.7: Histórico das coordenadas e dos perímetros dos talhões da fazenda “Recanto”

A figura 4.7 apresenta o resultado de uma consulta espaço-temporal que retorna o histórico das coordenadas e dos perímetros dos talhões da fazenda “Recanto”. A consulta correspondente é:

```
select id_parcel, date_begin_parcel, parcel.date_end,
parcel.name, astext(parcel.coordinates),
perimeter(parcel.coordinates)
from property, parcel
where property.name = ‘Fazenda Recanto’
and property.id_property=parcel.id_property
and property.date_begin_property=parcel.date_begin_property
order by(id_parcel)
```

A resposta mostra que a fazenda “Recanto” possuiu ao longo do tempo dois talhões “X” e “Y”. O talhão “X” passou por três mudanças ao longo do tempo tanto no que

name (varc...	date_begin...	date_end (...	astext (text)	name (varc...
Talhão X	2003-06-28	2004-06-28	POLYGON((0 0,1.5 0,1.5 1.5,0 1.5,0 0))	cafe
Talhão Y	2003-06-28	2004-06-28	POLYGON((2 2,2 1.5,1.5 1.5,1.5 2,2 2))	cana
Talhão X	2004-06-28	2005-06-28	POLYGON((0 0,1 0,1 1,0 1,0 0))	cana
Talhão Y	2004-06-28	2005-06-28	POLYGON((1 1,1.5 1,1.5 1.5,1 1.5,1 1))	cafe
Talhão Y	2005-06-28	2006-06-28	POLYGON((1 1,1 0.5,0.5 0.5,0.5 1,1 1))	soja
Talhão X	2005-06-28	2006-06-28	POLYGON((0 0,0 0.5,0.5 0.5,0.5 0,0 0))	cana
Talhão Y	2006-06-28		POLYGON((1 1,1 0.5,0.5 0.5,0.5 1,1 1))	cafe
Talhão X	2006-06-28		POLYGON((0 0,0 0.5,0.5 0.5,0.5 0,0 0))	soja

Figura 4.9: Histórico das culturas cultivadas nos talhões da fazenda “Recanto”

```
and property.date_begin_property=parcel.date_begin_property
order by (parcelcrop.date_begin)
```

A resposta mostra que a fazenda “Recanto” possuiu ao longo do tempo dois talhões “X” e “Y”. No talhão “X” no período de 28 de junho de 2003 até 28 de junho de 2004 era cultivado café. No período seguinte, de 28 de junho de 2004 até 28 de junho de 2005, optou-se pelo cultivo de cana. No período posterior, de 28 de junho de 2005 até 28 de junho de 2006, a cana continuou a ser a cultura cultivada mas houve mudança na geometria do talhão. E finalmente, a partir de 28 de junho de 2006, a soja passou a ser cultivada no talhão “X”. A resposta mostra também que cana foi cultivada no talhão “Y” no período de 28 de junho de 2003 até 28 de junho de 2004. No período seguinte, de 28 de junho de 2004 até 28 de junho de 2005, optou-se pelo cultivo de café. No período posterior, de 28 de junho de 2005 até 28 de junho de 2006, a soja passou a ser a cultura cultivada. E finalmente, a partir de 28 de junho de 2006, o café voltou a ser cultivada no talhão “Y”.

name (varc...	astext (text)	astext (text)	within (bool)
Talhão X	POLYGON((0 0,0 0.5,0.5 0.5,0.5 0,0 0))	POLYGON((0 0,0 1,1 1,0 0,0 0))	t
Talhão Y	POLYGON((1 1,1 0.5,0.5 0.5,0.5 1,1 1))	POLYGON((0 0,0 1,1 1,0 0,0 0))	t

Figura 4.10: Talhões da Fazenda Recanto.

A figura 4.10 apresenta o resultado de uma consulta espacial que retorna quais os

talhões da fazenda “Recanto” atualmente estão contidos na própria fazenda. A consulta correspondente é:

```
select parcel.name, astext(parcel.coordinates),
astext(property.coordinates),
within(parcel.coordinates, property.coordinates)
from property, parcel
where property.id_property=parcel.id_property
and property.date_begin_property=parcel.date_begin_property
and property.name = 'Fazenda Recanto'
and property.date_end is NULL and parcel.date_end is NULL
```

A resposta mostra, conforme se esperava, que todos os talhões da fazenda “Recanto” se situam nela. O último campo indica se o resultado do “within” é falso ou verdadeiro.

name (varchar)	astext (text)	name (v...	astext (text)	intersects
Fazenda Recanto	POLYGON((0 0,0 1,1 1,1 0,0 0))	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda A	POLYGON((10 10,10 11,11 11,11 10...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda E	POLYGON((40 40,40 41,41 41,41 40...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda F	POLYGON((50 50,50 51,51 51,51 50...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda G	POLYGON((200 200,200 201,201 20...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda I	POLYGON((70 70,70 71,71 71,71 70...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda J	POLYGON((80 80,80 81,81 81,81 80...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda K	POLYGON((90 90,90 91,91 91,91 90...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda L	POLYGON((300 300,300 301,301 30...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda B	POLYGON((100 100,100 101,101 10...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda C	POLYGON((15 15,15 16,16 16,16 15...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda D	POLYGON((35 35,35 36,36 36,36 35...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	f
Fazenda X	POLYGON((61 61,61 62,62 62,62 61...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	t
Fazenda Y	POLYGON((59 59,59 60,60 60,60 59...	Fazenda H	POLYGON((60 60,60 61,61 61,61 60,60 60))	t

Figura 4.11: Relação de fronteira da Fazenda H.

A figura 4.11 apresenta o resultado de uma consulta espacial que retorna quais fazendas intersectam a fazenda “H”. A consulta correspondente é:

```
select p.name, astext(p.coordinates), t.name, astext(t.coordinates),
Intersects(p.coordinates,t.coordinates)
from property as p, property as t
where t.date_end is NULL and t.name = 'Fazenda H' and
p.name != 'Fazenda H' and p.date_end is NULL
```

A resposta mostra que a fazenda “H” é um polígono (60 60,60 61, 61 60, 61) que possui relação de intersecção com as fazendas “X”, que é o polígono (61 61, 61 62, 62 61, 62 62) e “Y”, que é o polígono(59 59, 59 60, 60 59, 60 60).

4.4 **Resumo**

Este capítulo apresentou brevemente a implementação realizada, em exemplos de resultados de consultas temporais, espaciais e espaço-temporais. A implementação foi realizada com dados fictícios, no Postgresql versão 8.0 rodando em um ambiente Linux Debian.

Capítulo 5

Conclusões e Extensões

Esta dissertação propôs um modelo do banco de dados para o projeto WebMaps. As principais contribuições são:

- Especificação de um modelo de dados com suporte ao gerenciamento espaço-temporal, tendo como estudo de caso o projeto WebMaps, visando atender a usuários de diferentes tipos (desde cooperativas até usuários individuais) que desejem usar dados associados à produção agrícola para tomada de decisões (Capítulo 3).
- Especificação de um conjunto de consultas temporais, espaciais e espaço-temporais para validar o modelo de dados proposto (Capítulo 3).
- Implementação de um protótipo, utilizando Postgresql/Postgis, para mostrar a validade do modelo de dados proposto e a adequação das consultas especificadas (Capítulo 4).

As dificuldades ao longo do desenvolvimento desta dissertação foram várias, entre as quais podemos citar:

- Integração do banco de dados implementado ao sistema desenvolvido sobre as imagens. Pela dificuldade encontradas, esta integração foi abandonada.
- Entendimento das consultas necessárias aos usuários, para então propor o modelo de dados.
- Instalação e uso do Postgres e extensão para o Postgis, devido aos tipos de dados geográficos deste último, que permitem vários formatos. Além disso, a documentação de instalação do Postgis ainda apresenta lacunas.

As extensões a este trabalho podem ser tanto teóricas quanto práticas. As seguintes extensões podem ser consideradas:

- A versão atual do projeto WebMaps não inclui a implementação em PostGis do banco de dados. Portanto, existe a necessidade da implementação dos módulos necessários para a utilização desse modelo no projeto.
- Implementação das consultas especificadas e não implementadas, como por exemplo as relativas aos perfis de consultas armazenadas.
- Estudo sobre a utilização de índices espaciais e temporais visando a melhoria do desempenho das consultas espaço-temporais.
- Levantamento de dados agrícolas não inclusos no modelo tais como: dados de solo, de clima, de chuva, etc. Dado esse fato, é necessário a ampliação do modelo ER e do modelo relacional dando suporte a estes dados. Além disso, surge a oportunidade de se especificar novas consultas espaço-temporais utilizando estes dados.
- Integração das imagens de satélite com o modelo de dados. Essa integração pode ser feita através da utilização de metadados. Para obter um melhor gerenciamento desses metadados, uma boa opção pode ser o uso de ontologias.
- Desenvolvimento de uma interface gráfica para interação do usuário.

Apêndice A

Script SQL contendo a implementação do modelo proposto

Criação das tabelas.

```
CREATE TABLE User1 (  
    id_user          INTEGER NOT NULL,  
    name             CHAR(40) NOT NULL,  
    login            CHAR(10) NOT NULL,  
    password         CHAR(20) NOT NULL,  
    e_mail           CHAR(30) NOT NULL,  
    adress           CHAR(40) NOT NULL,  
    bairro           CHAR(20) NOT NULL,  
    number           INTEGER NOT NULL,  
    city             CHAR(20) NOT NULL,  
    state            CHAR(20) NULL,  
    cep              CHAR(20) NULL,  
    complement       CHAR(20) NULL,  
    id_usertype      INTEGER NOT NULL,  
    PRIMARY KEY (id_user)  
)
```

```
CREATE TABLE Crop (  
    id_crop          int NULL,  
    name             varchar(40) NULL,  
    PRIMARY KEY (id_crop)  
)
```

```
CREATE TABLE Property (  
    id_property          INTEGER NOT NULL,  
    date_begin_property  date NOT NULL,  
    date_end             date NULL,  
    name                 VARCHAR(40) NULL,  
    description          VARCHAR(80) NULL,  
    id_user              INTEGER NOT NULL,  
    PRIMARY KEY (id_property, date_begin_property)  
)  
  
CREATE TABLE Parcel (  
    id_parcel            int NOT NULL,  
    date_begin_parcel   date NOT NULL,  
    id_property          INTEGER NULL,  
    date_begin_property  date NULL,  
    date_end             date NULL,  
    name                 VARCHAR(40) NULL,  
    PRIMARY KEY (id_parcel, date_begin_parcel)  
)  
  
CREATE TABLE UserProperty (  
    id_user              INTEGER NOT NULL,  
    id_property          INTEGER NOT NULL,  
    date_begin_property  date NOT NULL,  
    PRIMARY KEY (id_User, id_property, date_begin_property)  
)  
  
CREATE TABLE ParcelCrop (  
    id_parcel            int NOT NULL,  
    date_begin_parcel   date NOT NULL,  
    id_crop              int NOT NULL,  
    date_begin           date NOT NULL,  
    date_end             date NULL,  
    Primary key (id_crop, id_parcel,  
                date_begin, date_begin_parcel)  
)
```

```
CREATE TABLE PropertyParcel (  
    id_parcel          int NOT NULL,  
    date_begin_parcel date NOT NULL,  
    id_property        int NOT NULL,  
    date_begin_property date NOT NULL,  
    Primary key (id_parcel, id_property,  
    date_begin_property, date_begin_parcel)  
)
```

```
CREATE TABLE Query (  
    id_query          int NOT NULL,  
    description       VARCHAR(80) NULL,  
    id_user           int NOT NULL,  
    id_parcel         int NOT NULL,  
    date_begin_parcel date NOT NULL,  
    id_property        int NOT NULL,  
    date_begin_property date NOT NULL,  
    PRIMARY KEY (id_query)  
)
```

```
CREATE TABLE Usertype (  
    id_usertype       int NOT NULL,  
    description       VARCHAR(80) NULL,  
    PRIMARY KEY (id_usertype)  
)
```

```
CREATE TABLE Permission (  
    id_permission     int NOT NULL,  
    id_usertype       int NOT NULL,  
    description       VARCHAR(80) NULL,  
    PRIMARY KEY (id_permission)  
)
```

```
CREATE TABLE ImageMBB (  
    id_imagembb       int NOT NULL,  
    date              date NOT NULL,  
    path              VARCHAR(120) NULL,
```

```

        id_imagesatellite      int NOT NULL,
        PRIMARY KEY (id_imagembb)
    )

```

```

CREATE TABLE ImageMask (
    id_imagemask      int NOT NULL,
    date              date NOT NULL,
    path              VARCHAR(120) NULL,
    id_imagembb      int NOT NULL,
    PRIMARY KEY (id_imagembb)
)

```

```

CREATE TABLE ImageSatellite (
    id_imagesatellite      int NOT NULL,
    date                    date NOT NULL,
    path                    VARCHAR(120) NULL,
    PRIMARY KEY (id_imagesatellite)
)

```

Inserção das chaves estrangeiras

```

ALTER TABLE User1
ADD FOREIGN KEY (id_usertype)
REFERENCES Usertype;

```

```

ALTER TABLE ParcelCrop
ADD FOREIGN KEY (id_parcel, date_begin_parcel)
REFERENCES parcel;

```

```

ALTER TABLE ParcelCrop
ADD FOREIGN KEY (id_crop)
REFERENCES crop;

```

```

ALTER TABLE Property
ADD FOREIGN KEY (id_user)
REFERENCES User1;

```

```

ALTER TABLE PropertyParcel
ADD FOREIGN KEY (id_property, date_begin_property)

```

```
REFERENCES Property;
```

```
ALTER TABLE PropertyParcel  
ADD FOREIGN KEY (id_parcel, date_begin_parcel)  
REFERENCES parcel;
```

```
ALTER TABLE Parcel  
ADD FOREIGN KEY (id_property, date_begin_property)  
REFERENCES Property;
```

```
ALTER TABLE Query  
ADD FOREIGN KEY (id_user)  
REFERENCES User1;
```

```
ALTER TABLE Query  
ADD FOREIGN KEY (id_parcel, date_begin_parcel)  
REFERENCES parcel;
```

```
ALTER TABLE Query  
ADD FOREIGN KEY (id_property, date_begin_property)  
REFERENCES Property;
```

```
ALTER TABLE ImageMBB  
ADD FOREIGN KEY (id_imagesatellite)  
REFERENCES ImageSatellite;
```

```
ALTER TABLE ImageMask  
ADD FOREIGN KEY (id_imagembb)  
REFERENCES ImageMBB;
```

```
ALTER TABLE Property  
ADD FOREIGN KEY (id_imagembb)  
REFERENCES ImageMBB;
```

```
ALTER TABLE Parcel  
ADD FOREIGN KEY (id_imagembb)  
REFERENCES ImageMBB;
```

```
ALTER TABLE Permission  
ADD FOREIGN KEY (id_usertype)  
REFERENCES Usertype;
```

Inserção das colunas geométricas.

```
SELECT AddGeometryColumn('property', 'coordinates', 1, 'POLYGON', 2);
```

```
SELECT AddGeometryColumn('parcel', 'coordinates', 1, 'POLYGON', 2);
```

Referências Bibliográficas

- [1] J. F. Allen. Maintaining knowledge about temporal intervals. volume 26, pages 832–843, New York, NY, USA, 1983. ACM Press.
- [2] M. A. Botelho. Incorporação de facilidades espaço-temporais em bancos de dados orientados a objetos, MSC thesis, Universidade Estadual de Campinas, 1995.
- [3] C. Claramunt and B. A. Jiang. A representation of relationships in temporal spaces. In *Innovations in GIS VII: GeoComputation*, 2000.
- [4] G. Câmara, M. A. Casanova, A. S. Hemerly, G. C. Magalhães, and C. M. B. Medeiros. *Anatomia de Sistemas de Informação Geográfica*. X Escola de Computação Unicamp, first edition, 1996.
- [5] Open GIS Consortium. *OpenGIS Simple Features Specification For SQL*. Open GIS Consortium, first edition, 1997.
- [6] L. M. D. Cura. Controle de versão em banco de dados para aplicações gis, MSC thesis, Universidade Estadual de Campinas, 1997.
- [7] P V Oosterom E Clementini, P D Felice. A small set of formal topological relationships suitable for end-user interaction. In *SSD '93: Proceedings of the Third International Symposium on Advances in Spatial Databases*, pages 277–295, London, UK, 1993. Springer-Verlag.
- [8] M. J Egenhofer and R. Ranzosa. Point-set topological spatial relations. In *International Journal of Geographical Information Systems*, pages 161–174, 1991.
- [9] R. A. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley Publishing, 3^a edition, 1999.
- [10] S. Santilli et al. Postgis manual. In *PostGIS Manual*, pages 1–67, 2005.

- [11] G. Faria. Um banco de dados espaço-temporal para desenvolvimento de aplicações em sistemas de informação geográfica, MSC thesis, Universidade Estadual de Campinas, 1998.
- [12] R.H Guting. An introduction to spatial database systems. In *VLDB Journal*, volume 3, pages 82–94.
- [13] J. G. S. Lima. Gerenciamento de dados climatológicos heterogêneos para aplicações em agricultura, MSC thesis, Universidade Estadual de Campinas, 2003.
- [14] C. Davis L. Vinhas G. R. Queiroz M Casanova, G. Câmara. *Bancos de Dados Geográficos*. MundoGEO, first edition, 2005.
- [15] M. Schneider M. Erwig. Spatio-temporal predicates. *IEEE Transactions on Knowledge and Data Engineering*, 2002.
- [16] M. Schneider M. Vazirgiannis M. Erwig, R. H. Guting. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296, 1999.
- [17] P. A. Meneses. *Fundamentos de Radiometria Óptica Espectral*. In: *Sensoriamento Remoto - reflectância dos alvos naturais*. Universidade de Brasília. Brasília/DF., 1ª edition, 2001.
- [18] B Pernici N. Edelweiss, J. P.M. Oliveira. *Modelagem de Aspectos Temporais de Informação*. IX Escola de Computação de Recife, 1994.
- [19] P. J. Rezende and J. Stolfi. *Fundamentos de Geometria Computacional*. Recife, PE: IX Escola de Computação, 1ª edition, 1994.
- [20] J. Schimiguel. *Padrões de Qualidade para o Design de Interfaces em SIG Web*. PhD thesis, Universidade Estadual de Campinas, 2006.
- [21] L. D. Bergman V. Castelli. *Image Databases - Search and Retrieval of Digital Imagery*. Wiley Inter-Science, first edition, 2002.