Visões em Sistemas de Informações Geográficas– modelo e mecanismos

Nevton Cereja

Visões em Sistemas de Informações Geográficas– modelo e mecanismos

Nevton Cereja[1]

Instituto de Computação
UNICAMP

Banca Examinadora:

- Claudia M. B. Medeiros[2] (Orientador)
- Geovane Cayres Magalhães[2]
- Regina Ruschel[3]
- Eliane Martins[2] (Suplente)

[1]Bacharel em Ciência da Computação pela Universidade Estadual de Londrina - Pr.
[2]Instituto de Computação - UNICAMP.
[3]Faculdade de Engenharia Civil - UNICAMP.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Cereja, Nevton

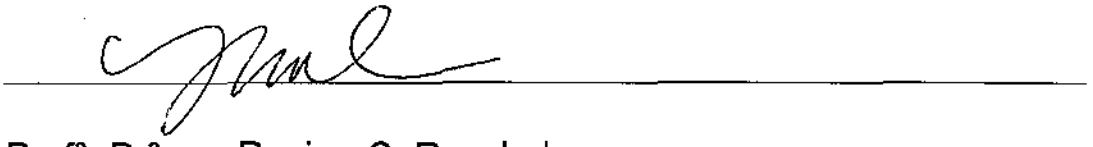
C334v Visões em sistemas de informações geográficas - modelo e mecanismos / Nevton Cereja -- Campinas, [S.P. :s.n.], 1996.

Orientador : Medeiros, Claudia Bauzer

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

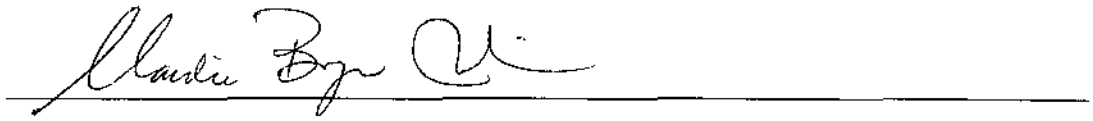
1. Sistemas de informações geográficas. 2. Banco de dados orientados a objetos. I. Medeiros, Claudia Bauzer. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Tese de Mestrado defendida e aprovada em 16 de dezembro de 1996 pela Banca Examinadora composta pelos Professores Doutores



Profª. Drª. Regina C. Ruschel

Profº. Drº. Geovane Magalhães



Profª. Drª. Claudia Maria B. Medeiros

Visões em Sistemas de Informações Geográficas– modelo e mecanismos

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. Nevton Cereja e aprovada pela Comissão Julgadora.

Campinas, 18 de dezembro de 1996

Profa. Dra. Claudia Bauzer Medeiros
Orientadora

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

Àqueles que sofreram com minha ausência;
tão poucos que sabem quem são.

Agradecimentos

Eu gostaria de agradecer aos alunos, professores e funcionários do IC pelo agradável ambiente de trabalho.
Aos companheiros do grupo de Banco de Dados do IC, Fátima, Juliano, Mariano Cilia, Márcio Botelho, Luis Mariano, Marcos André, Gláucia, Walter, Alexandre e Marco Antônio pela efetiva participação e questionamento nas apresentações feitas ao grupo.
Ao meu amigo Henrique Wiermann Paques, por todo apoio e incentivo incontestes.
Ao meu companheiro de trabalho e amigo, Luis Mariano del Val Cura, por todas as discussões, que desencadearam muitos dos erros e acertos que contribuíram para o nosso amadurecimento no trabalho de pesquisa.
Aos meus amigos e companheiros de moradia, Luciano, Vitor Hugo e Márcio, por terem propiciado a criação de um ambiente amigável e fraterno durante nossos 20 meses de convivência.
Ao Professor Cid Carvalho de Souza, pela acolhida profissional como professor e pela orientação extra-acadêmica, que recebi como sinal de amizade.
Ao Juliano Lopes de Oliveira, por toda orientação, apoio e companheirismo enquanto colega de trabalho mas, principalmente, pelas incontáveis demonstrações de amizade. É um grande privilégio poder chamá-lo de AMIGO.
Ao meu irmão Edson, meu grande incentivador e amigo de todas as horas, exemplo de ser humano. Obrigado pelo exemplo.
À minha familia, Jacinto, Victória, Aparecida, Nelson, Edson, Mônica, Rovilso, Marta e Pithina, por existirem.
À minha orientadora, Profa. Dra. Claudia Bauzer Medeiros, por sua orientação e dedicação, fundamentais para a elaboração e conclusão deste trabalho.
E, principalmente, a Deus, que é meu pastor e torna minha vida farta. Obrigado pelos desafios e pela coragem de enfrentá-los.
Este trabalho foi desenvolvido dentro do projeto do CNPq PROTEM/CC-GEOTEC e financiado parcialmente pela CAPES e pela FAPESP.

Resumo

Esta dissertação investiga as funcionalidades oferecidas por modelos e mecanismos de visões e as necessidades de SIG que podem ser satisfeitas através de tais funcionalidades. As principais contribuições deste trabalho são (1) um estudo detalhado do papel de visões em SIG, (2) a proposta de um modelo de visões orientado a objetos a ser usado em SIG, mostrando a necessidade de dados e informação semântica adicional em relação aos modelos covencionais, (3) apresentação da arquitetura de um mecanismo para implementar tal modelo e (4) uma linguagem para especificar visões no modelo. A validação do trabalho é feita através da modelagem de uma aplicação real em SIG.

Abstract

This thesis analyses the functionality offered by view mechanisms in order to satisfy specific GIS *needs. The main results presented are: (1) a detailed analysis of the role views can play in the* GIS *context; (2) the specification of an object oriented view model to be used in* GIS*, which shows the need for additional data and semantic information in order to support the required functionality; (3) the presentation of a mechanism to support the model; and (4) a language to specify views in this model. The work developed is validated through the modelling of a real world application using the model and language proposed.*

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Esta dissertação identifica problemas em Sistemas de Informações Geográficas (SIG) que podem ser solucionados com o uso de visões e propõe um modelo e um mecanismo de visões, definidos como uma camada sobre um Sistema de Gerenciamento de Banco de Dados orientado a objetos (SGBDOO) genérico, como meio para solucionar tais problemas. O modelo e o mecanismo de visões são validados através de um estudo de caso, onde um problema real é especificado na linguagem de definição de visões geográficas (GeoVDL), que reflete os conceitos do modelo de visões proposto.

SGBD são divididos em 3 níveis: externo, lógico e interno [ANS75]. O nível externo é utilizado para apresentar aos usuários as informações contidas no banco de dados e por isso é chamado de *visão* do usuário. Visões são abstrações que compõem um bloco de trabalho bem definido e que possibilitam integrar diversas funcionalidades de bancos de dados, como por exemplo controle de acesso, limitação e re-estruturação de dados. O usuário de uma visão pode ser uma aplicação ou um ser humano.

Originalmente definido sobre o modelo de dados relacional, o conceito de visão está ligado à re-estruturação e limitação da informação. Em bancos de dados, a re-estruturação da informação é fundamental devido à fragmentação dos dados que descrevem as entidades do mundo real mapeadas para o banco de dados. Tal fragmentação é causada por modelagem (visão parcial do usuário) e armazenamento (por razões de desempenho ou funcionamento do SGBD). No caso de SGBD Relacionais (SGBDR), existe ainda a fragmentação devido à normalização das relações do banco de dados.

Em SGBDR, visões são relações virtuais[1] não-normalizadas definidas com base em relações preexistentes, virtuais ou não. Por serem relações, visões podem ser descritas como tendo *esquema* (definição dos atributos e domínios) e *extensão* (dados). Como uma visão é uma relação virtual, sua extensão é não-persistente, sendo criada cada vez que a visão é *invocada* pelo usuário. Nesta dissertação, a criação da extensão de uma visão receberá o nome de *povoamento*, para indicar que se trata de uma extensão não-

[1] Relações cujos dados não estão fisicamente armazenados e que só existem durante a ativação da visão.

persistente. Em SGBDR, o povoamento é obtido através da execução de uma operação de consulta, expressa por meio de comandos de consulta escritos na linguagem de consulta do SGBD. Comandos de consulta incluem predicados que permitem particionar o banco de dados, determinando o conjunto de dados relevantes à aplicação que utilizará a visão. Em SGBD Orientados a Objetos (SGBDOO) não há consenso sobre o que é uma visão. Alguns autores consideram uma visão como um contexto no qual métodos são associados a objetos [HZ88, HZ90] enquanto outros tentam reproduzir no contexto de orientação a objetos as funcionalidades oferecidas por visões em SGBDR [Wie86, SS89, MM91, AB91, dSAD93, Med94, Bra92].

Neste texto, utilizamos o modelo de objetos baseado em classes de [Bee89]. Um objeto é uma instância de uma classe e é caracterizado por seu *estado* (conjunto de valores de atributos) e *comportamento* (conjunto de operações ou métodos que podem ser aplicados ao objeto). Um objeto o pode ser composto de outros objetos $o_1,...,o_n$, caso em que o é chamado *complexo* e $o_1,...,o_n$ são os *componentes* de o. Este processo de composição é realizado através da aplicação de *construtores* – por exemplo, construtores de conjunto – que permitem especificação progressiva de objetos cada vez mais complexos a partir de componentes previamente definidos. Objetos não complexos são denominados *simples*. As classes são estruturadas em hierarquias de herança; os ancestrais de uma classe C na hierarquia são as *superclasses* de C e seus descendentes as *subclasses* de C. Os descendentes de uma classe *herdam* sua estrutura e operações.

Mecanismos de visões implementados em bancos de dados convencionais oferecem diversas funcionalidades, como re-estruturação das informações presentes no banco de dados e derivação de informações não fisicamente armazenadas, dentre outras. Além dessas funcionalidades, visões também são utilizadas para resolver outros problemas em bancos de dados: integração de bancos de dados heterogêneos e simulação de evolução de esquema.

Sistemas de Informações Geográficas (SIG) são sistemas de informação que manipulam dados sobre fenômenos geográficos, associados a sua localização física e relacionamentos espaciais [OM95]. SIG são fortemente dependentes de bancos de dados espaciais e podem ser implementados sobre bancos de dados distintos: relacionais, relacionais estendidos e orientados a objetos. Logo, problemas comuns a bancos de dados em geral também são encontrados no contexto de SIG. Acredita-se que o paradigma de orientação a objetos ofereça um ambiente mais propício para SIG devido, principalmente, à possibilidade de representar as entidades do mundo real diretamente no modelo conceitual. Além disso, as características de reutilização e extensibilidade contribuem para atender aos requisitos dos usuários de SIG, cujas necessidades são distintas e carecem de definições mais precisas.

As aplicações geográficas têm como características marcantes o grande volume de dados envolvidos e o geo-referenciamento destes dados. Especialmente por estes fatores, aplicações geográficas têm necessidades que extrapolam os recursos oferecidos por SGBD

convencionais. A ausência de visões, tanto em SIG como em SGBD, exige que as funcionalidades normalmente por elas oferecidas sejam supridas através de programas de aplicação, aumentando custo e tempo de desenvolvimento e a possibilidade de surgimento de toda a gama de problemas relativos a desenvolvimento, teste e validação de softwares.

O papel de visões em SIG não é claro, motivando seu estudo neste trabalho. As peculiaridades das aplicações e a vasta gama de usuários deste tipo de sistema abrem um grande leque de necessidades a serem atendidas. Neste contexto, os mecanismos de visões implementados em SGBD convencionais, embora úteis, não suprem as necessidades de visões no que diz respeito às aplicações geográficas que, por serem extremamente específicas, não são atendidas pelos modelos e mecanismos de visão convencionais.

Dentre as possíveis utilizações em SIG, a dissertação propõe o uso de visões na aplicação de funções de análise sofisticadas, em consultas exploratórias, na limitação da informação (passando pela criação de abstrações de geoprocessamento e de cenários) e na criação de representações alternativas para uma mesma entidade geográfica.

O conceito de cenário pode ser traduzido, em terminologia de banco de dados, como a especificação de uma visão e diz respeito a um conjunto de dados relevante a uma aplicação geográfica em relação a uma determinada área-alvo, onde podem ser combinados dados geográficos acrescidos de informações adicionais (contexto) que permitam a correta interpretação semântica do cenário e do resultado das análises realizadas sobre ele.

Do ponto de vista de bancos de dados pode-se considerar diferentes representações como visões da mesma entidade. Em SIG, múltiplas representações de entidades geográficas podem ser obtidas através da derivação de visões (representações) de uma representação base ou canônica (seção 3.2.2).

Visões podem ser utilizadas como blocos de construção para a definição de outras visões, as chamadas *visões aninhadas*, as quais são especializações da visão original. Visões aninhadas podem também ser utilizadas para suportar consultas exploratórias em SIG (seção 3.2.2).

O resto da dissertação está dividido da seguinte forma: o capítulo 2 contém uma revisão bibliográfica sobre visões na literatura, tanto do ponto de vista de aplicabilidade quanto de problemas de implementação, e discute os fatores envolvidos em projetos de modelos de visões. O capítulo 3 traz uma revisão dos conceitos de SIG. Além disto, revê algumas propostas de mecanismos de visões para SIG e levanta alguns problemas que podem ser solucionados com o uso de visões . O capítulo 4 apresenta um modelo e um mecanismo de visões para SIG, oferecendo soluções para os problemas levantados no capítulo 3. Ao final, apresenta a linguagem de definição de visões utilizada para definir as visões em SIG de acordo com o modelo proposto. O capítulo 5 mostra como o modelo e a linguagem propostos podem ser usados para resolver problemas de uma aplicação do mundo real. Finalmente, o capítulo 6 apresenta as conclusões finais da dissertação e as direções para futuros trabalhos.

Capítulo 2

Revisão Bibliográfica de Visões

Uma *visão* é uma abstração que reflete um enfoque particular de um usuário sobre o mundo. O uso de visões permite a re-estruturação e re-modelagem do conjunto de dados sobre o qual é definida. A re-estruturação é obtida através da combinação, ocultamento e criação de novas informações a partir daquelas já armazenadas e permite que informações inter-relacionadas possam ser agrupadas em uma nova unidade semântica, mais útil e compreensível para o usuário.

Este capítulo faz uma revisão bibliográfica de propostas de visões na literatura e identifica as principais funcionalidades oferecidas por mecanismos de visões. Além disso, mostra como tais funcionalidades são inter-dependentes e apresenta os fatores que devem ser considerados na definição de um modelo e mecanismo de visões orientadas a objetos.

2.1 Visões e suas Aplicações

O conceito de visões é bastante difundido. Muitas definições abordam diferentes utilizações e características. Do ponto de vista de banco de dados, uma visão pode possuir algumas propriedades que modificam sua semântica de utilização e manutenção. Uma visão pode refletir o momento atual do banco de dados (*snapshot*) ou pode refletir estados anteriores (*visões históricas*), pode ser derivada diretamente do banco de dados ou pode ser resultante de operações sobre os dados do banco de dados e, por fim, podem servir de apoio a sistemas de banco de dados de tempo real [AGMK95].

Em sistemas gerenciadores de bancos de dados relacionais (*SGBDR*), visões são encaradas como relações virtuais não-normalizadas, definidas em função de relações preexistentes e obtidas como resultado do processamento de um comando de consulta [Sto75, FSdS79, BH88, AKK94, Wie86].

Não há um consenso na comunidade de banco de dados sobre o que é uma visão em SGBD orientados a objetos (*SGBDOO*)[MM91]. Alguns autores consideram visões como um contexto no qual métodos são associados a objetos [HZ88, HZ90]. Outros consideram

visões como uma nova unidade de abstração que, como classes, possui propriedades e métodos [Wie86, SS89, MM91, AB91, dSAD93, Med94, Bra92].

A seguir são comentados alguns dos principais contextos em que mecanismos de visões são executados: limitação de dados, geração de informações, integração de bases heterogêneas, desempenho, segurança e integridade do banco de dados e evolução do esquema. A apresentação será feita em ordem de modelo (primeiro relacional, depois orientado a objetos) e, para cada modelo, em ordem cronológica.

2.1.1 Limitação de Dados

Um ponto comum encontrado nas propostas de mecanismos de visões é sua utilização na limitação e segurança dos dados [Sto75, FSdS79, BH88, SJGP90, PMSL94, AKK94, Wie86, HZ88, SS89, HZ90, AB91, Ber91, MM91, Bra92, dSAD93, Med94]. O conceito de *limitação* corresponde ao fato de que cada aplicação limita o conjunto de dados a que tem acesso de acordo com suas necessidades.

Entidades com as mesmas propriedades são agrupadas em coleções. Em bancos de dados relacionais, cada entidade é representada por um conjunto de tuplas, de relações potencialmente diferentes (uma relação não necessariamente representa uma entidade). Em sistemas orientados a objetos, entidades são objetos com propriedades e comportamentos semelhantes, agrupados em *classes*. Visões possibilitam o *particionamento* destas coleções, permitindo a seleção das entidades que satisfazem determinadas condições. A grande maioria dos autores destina o trabalho de *particionamento* ao processador de consultas [Sto75, FSdS79, BH88, AKK94, Wie86, HZ90, AB91, Ber91, MM91, Ber92, Bra92, dSAD93, Med94].

Em [AB91] o particionamento pode também ser realizado por meio do mecanismo de *herança parametrizada*, onde um predicado é associado à definição da subclasse. Na ativação da subclasse especializada pode-se ou não passar parâmetros, que serão avaliados na determinação das instâncias da nova subclasse. Somente os objetos que satisfazem o predicado farão parte dos dados da visão. As *classes invariantes* [dSAD93] são mecanismos de particionamento semelhantes à herança parametrizada. Na definição da classe é anexado um predicado com uma invariante, relacionada a algum atributo da classe. Somente os objetos que atendem ao predicado farão parte da extensão da classe.

2.1.2 Geração de Informações

Muitas vezes, dados necessários a certas aplicações não são incluídos no banco de dados. Entre outros motivos, esta ausência pode ocorrer devido a decisões de projeto ou mesmo pela evolução do sistema de informação. Quando uma mesma realidade é percebida de forma diferente por diversos usuários, o projetista do banco de dados pode optar por armazenar apenas uma representação da informação, derivando as demais representações

quando necessário. Algumas vezes tal derivação pode não ser possível. Além disso, por questões de economia de espaço de armazenamento, pode-se optar por não armazenar dados que podem ser obtidos através da computação de outros dados já armazenados.

Alguns mecanismos de visões permitem a geração de novos dados a partir daqueles já existentes no banco de dados. Um exemplo deste tipo de mecanismo são os *atributos virtuais* [AB91, dSAD93], funções especiais cujo valor retornado faz parte da estrutura da visão. Os atributos virtuais não são armazenados fisicamente, mas podem ser utilizados como um dado comum, inclusive na computação de novos atributos virtuais.

Em [Bra92], funções de mapeamento entre representações diferentes do mesmo dado permitem que apenas um formato seja armazenado. Quando outros formatos tornam-se necessários, eles podem ser derivados do formato armazenado.

O mecanismo de visão proposto por [Ber91] permite a inclusão na visão de propriedades e métodos não presentes na classes sobre as quais a visão foi definida, permitindo que novos dados sejam derivados dos dados já existentes e que o comportamento do objeto seja alterado.

2.1.3 Integração de Bases Heterogêneas

Os dados que compõem o esquema de uma visão são normalmente derivados de um banco de dados único, que armazena as informações relevantes àquela aplicação. Contudo, muitas aplicações exigem que dados de bancos de dados diferentes sejam combinados, proporcionando uma nova unidade de informação para o usuário. Uma das formas de realizar tal integração é através de visões.

Normalmente, os bancos de dados que serão integrados apresentam algum tipo de heterogeneidade, quer seja no modelo de dados utilizado, quer seja na semântica dos dados representados. Independentemente da causa e nível de heterogeneidade apresentado, alguns autores consideram visões como sendo o mecanismo ideal para solucionar o problema, realizando a integração ao nível externo do banco de dados [dA95a]. Este nível de integração possibilita guardar certa independência entre aplicações e representações internas do banco de dados. A integração de bancos de dados heterogêneos pode ser obtida de diversas formas. A definição de um esquema global ou um dicionário de dados global são opções que permitem gerenciar a distribuição de dados com um bom nível de transparência de localização. Por outro lado, estas definições também constituem uma base de dados e estão sujeitas aos mesmos problemas das bases que desejam integrar.

Em [BH88] a proposta de integração tem por objetivo a autonomia local em detrimento à transparência. A integração é obtida através da avaliação dos comandos de consulta da linguagem de consulta distribuída. O esquema de uma visão é armazenado no local (*site*) onde ela foi definida e nos locais que possuem dados por ela referenciados. A árvore sintática do comando de consulta é dividida em partes tais que cada local armazena a

parte que referencia dados locais. No processamento, as árvores sintáticas são combinadas, gerando uma árvore sintática complexa que contém somente referências a relações do banco de dados subjacente. Quando uma informação muda de lugar, o sistema armazena apontadores para o novo endereço da informação, criando níveis de indireção no acesso às informações.

O modelo de dados FUGUE [HZ90] é implementado sobre um sistema de gerenciamento de objetos (*Object Management System* - OMS) cujo objetivo é permitir a interoperabilidade entre objetos heterogêneos. Toda solicitação de serviços é passada para o OMS que se encarrega da comunicação com o objeto capaz de atendê-la.

[dSAD93] propõe o conceito de *esquemas genéricos*, isto é, visões genéricas definidas sobre bases de dados heterogêneas para integrá-las. Uma visão genérica é definida sobre um esquema de um banco de dados base e sobre o esquema de uma base heterogênea. Na definição da visão, a identificação da base heterogênea se transforma em argumento, resolvido quando da ativação da visão.

[DGS94] apresenta uma plataforma de integração de bancos de dados heterogêneos chamada *Object View Broker* (OVB). O OVB é uma camada que faz a comunicação entre aplicações (baseadas no paradigma de orientação a objetos) e banco de dados (tipicamente relacionais) que funciona como cliente (para os bancos de dados) e servidor (para as aplicações) e proporciona suporte ao ambiente distribuído e independência entre aplicações e dados.

2.1.4 Desempenho

Uma das formas de se melhorar o desempenho do processamento de consultas é a materialização visões [SJGP90]. Assim, a cada ativação, os dados da visão são acessados diretamente, sem a necessidade de re-computação do comando de consulta. [SJGP90] utiliza, também, *caching* de regras de produção de acordo com a implementação escolhida para cada regra, baseando-se no provável número de tuplas envolvidas em sua execução.

Além da materialização, o particionamento e a forma de implementação do mecanismo de visões favorecem o desempenho do sistema como um todo. A estrutura das visões em [PMSL94] replica a estrutura relacional do banco de dados. Na ativação da visão os dados das relações reais são particionados pelo comando de consulta. As relações da visão, menores que as relações do banco de dados, são convertidas para representações em memória. Como o sistema é desenvolvido para uma plataforma *cliente-servidor*, a visão é enviada para a estação de trabalho. Assim, o mecanismo de visões diminui o tráfego pela rede e aumenta o desempenho na estação. [ZGMHW95] propõe algoritmos para manutenção consistente de visões materializadas.

Em sistemas orientados a objetos, [Wie86] propõe que uma nova interface seja criada no mecanismo de visões para que a recuperação de informações em bancos de dados utilize

conceitos de operações orientadas a conjuntos existentes nas linguagens de programação. [dSAD93] possibilita a materialização dos dados das visões.

2.1.5 Segurança e Integridade do Banco de Dados

Visões estão ligadas à limitação e à proteção das informações. A proteção caracteriza-se por impedir acesso a usuários que não têm autorização para fazer acesso a certos dados. A garantia de proteção contribui para a manutenção da integridade do banco de dados. Muitos autores garantem a proteção dos dados juntamente com a limitação de informações [HZ88, HZ90, AB91, Ber91, MM91, dSAD93].

Em [Sto75] restrições de integridade são armazenadas no banco de dados. Os comandos de consulta são re-escritos de forma a permitir que as restrições sejam verificadas. Caso alguma restrição seja violada, o comando de consulta é desabilitado. Já em [BH88], a segurança é garantida através da implementação de estratégias de controle de acesso e fluxo de dados. O *controle de acesso* discutido é classificado pelos autores como *dependente de dados* e tem forte influência no mecanismo de visões. Ele é implementado através de *regras de acesso*, que indicam o nível de acesso que um usuário tem em relação a um objeto, de acordo com um predicado. As regras de acesso podem ser armazenadas em um catálogo global ou juntamente com os dados sobre os quais se aplicam. Como o sistema permite que relações migrem de localidade transparentemente (mantendo uma tabela de referência no local original de criação), as regras de acesso podem migrar de localidade. O estabelecimento dos direitos de acesso pode ser permitido somente na localidade onde a tabela está fisicamente armazenada ou podem ser feitos em qualquer localidade.

[SJGP90] garante a limitação de dados e a integridade do banco de dados subjacente através de regras. [PMSL94] garante que somente os objetos complexos solicitados no comando de consulta sejam mapeados para representação em memória e, portanto, disponíveis ao usuário da visão.

Em [ZGMHW95] a integridade entre o banco de dados e as visões materializadas é mantida através de algoritmos especializados para ambientes *warehousing*, onde os dados base encontram-se dissociados dos dados das visões, em um ambiente distribuído. Os algoritmos de manutenção de visões baseiam-se na proposta de [BLT86]. As alterações nos dados base são notificadas ao ambiente *warehousing* por meio de mensagens, que desencadeiam operações de manutenção. Os algoritmos supõem a existência de ordenação de mensagens e de operações atômicas. A possibilidade de processamento local e a existência de chaves-primárias nas atualizações são exploradas pelos algoritmos.

[Wie86] preocupa-se em manter a integridade do banco de dados subjacente através do controle das atualizações em visões. Ele define uma camada de atualização de objetos, invocada quando um *commit* é solicitado. Quando o mapeamento para o banco de dados apresenta ambigüidades, uma das opções de mapeamento é escolhida durante a definição

da camada de atualização de objetos.

No contexto de segurança, [SS89] propõe um ambiente em que objetos são criados no contexto de visões das quais serão componentes. Cada propriedade de um objeto possui associada a si uma lista dos métodos aos quais ela é visível. Assim, somente os métodos que constam da lista podem ter acesso aos dados da visão.

O mecanismo de autorização escolhido por [Ber92] baseia-se na inclusão das regras de autorização nos métodos do objeto. Quando o cliente executa o método, além da autorização para tal, ele deve ter autorização para ter acesso a todos as propriedades às quais o método faz acesso. O mecanismo de segurança utiliza encapsulamento, diferenciando o encapsulamento que garante a correção na manipulação do objeto do encapsulamento que garante o controle de acesso. A política de controle de autorização é descentralizada, baseada no conceito de que o criador do objeto concede autorizações a usuários. Os usuários com autorização podem conceder autorizações do mesmo nível que possuem a outros usuários.

A integridade do banco de dados é garantida em [Bra92] por meio dos relacionamentos de consistência entre atributos (*ACR*), formados por uma tabela de mapeamento entre representações distintas de dados compartilhados por versões distintas do mesmo objeto. A verificação de consistência é feita através dos relacionamentos de consistência de operações entre as propriedades dos objetos que se sobrepõem, garantindo uma espécie de dependência funcional entre os objetos.

2.1.6 Evolução do Esquema

A evolução do estado dos dados é inevitável. Tal evolução pode significar a necessidade de incluir novas informações ou a mudança de perspectiva sobre entidades do mundo real. Muitos autores propuseram que a evolução de esquemas fosse controlada por meio de visões.

Em [Wie86] a evolução do esquema é obtida por meio de um mecanismo de herança, através do qual novos atributos podem ser incorporados a tipos. Esta solução visa resolver o problema decorrente da especialização de objetos compartilhados durante sua utilização.

A evolução do esquema apresentada em [Ber91] permite tanto a adição de informações e métodos quanto a re-estruturação da hierarquia de classes. A adição de atributos e métodos na definição das visões permite simular mudanças no esquema, como por exemplo, mudanças em definições de classes e em hierarquias de heranças e agregações.

[Bra92] entende a evolução do banco de dados como a evolução das classes e dos objetos que o compõem. A evolução das classes pode ser aditiva (especialização, generalização, projeção e versionamento) ou destrutiva (generalização) e a evolução dos objetos, isto é, a instanciação de objetos para as novas classes, ocorre através de sua criação ou adição à classe.

[dSAD93] propõe a utilização de visões genéricas para implementar a evolução do esquema do banco de dados. Novos dados seriam armazenados em bases distintas, integrados pelo mecanismo de visões genéricas e manipulados por meio de visões, de forma transparente para o usuário.

2.1.7 Resumo da aplicação de visões

A tabela 2.1 resume os artigos com relação aos tópicos discutidos nesta seção. Como pode ser notado, visões são muito utilizadas para re-estruturação de informações e para garantir a integridade e a segurança do dados.

Artigo	*Re-estruturação de Dados*			*Integração Bases Heterogêneas*	*Desempenho*	*Segurança/ Integridade*	*Evolução do Esquema*
	Projeção	*Particionamento*	*Ger. de Inform.*				
[Sto75]	X	X	X			X	
[FSdS79]	X	X				X	
[BH88]	X	X		X	X	X	
[SJGP90]	X	X			X	X	
[AKK94]	X	X		X			
[PMSL94]	X	X		X	X	X	
[Wie86]	X				X	X	X
[HZ88]	X	X				X	
[SS89]	X	X				X	
[HZ90]	X	X		X		X	
[AB91]	X	X	X			X	
[Ber91]	X	X	X			X	X
[MM91]	X	X				X	
[Ber92]						X	
[Bra92]	X	X	X			X	X
[dSAD93]	X	X	X	X	X	X	X

Tabela 2.1: Aplicações de visões

2.2 Problemas da especificação e da implementação de visões

Esta seção discute aspectos relativos à implementação e especificação de arquiteturas e mecanismos de visões. Muitas abordagens preocupam-se em solucionar problemas de aplicações de visões, como segurança, integração de bases heterogêneas e outras. A utilização de visões materializadas para melhorar o desempenho introduz o problema de manter os dados materializados da visão consistentes com o banco de dados original. O mesmo problema acontece no contexto de versões, com o acréscimo da necessidade de um controle especializado de recuperação de dados, que consiga combinar dados de versões diferentes sem comprometer o desempenho do banco de dados e garantir transparência ao usuário. A seguir são analisadas soluções para alguns desses problemas.

2.2.1 Atualização consistente do banco de dados

O mapeamento das atualizações em visões para o banco de dados subjacente pode não ser direto e, muitas vezes, é ambíguo. Estes motivos fazem com que muitos mecanismos de visões propostos simplesmente não permitam atualizações em visões, comprometendo a flexibilidade do mecanismo.

Em [Sto75] é permitida a atualização somente de visões que projetam atributos de uma relação do banco de dados subjacente, desde que os dados a serem atualizados não estejam envolvidos em nenhuma restrição definida para a relação.

[FSdS79] propõe um mecanismo de atualização de visões para um ambiente relacional. O mapeamento para o banco de dados subjacente depende da composição dos dados na visão. Quando o mapeamento de dados é direto, o mapeamento de atualizações também o é. Quando os dados da visão são compostos (por exemplo, através de junções) o mapeamento deve observar as restrições (políticas, de consistência, entre outras) impostas ao banco de dados e os relacionamentos entre atributos (como, por exemplo, dependências funcionais). Visões são compostas pela aplicação incremental de operadores da linguagem de consulta sobre uma base de dados, criando visões intermediárias. A alteração na visão é mapeada para a visão intermediária que a gerou. O processo repete-se recursivamente, até a alteração do banco de dados real e a re-computação da visão. Em cada atualização intermediária, as restrições são avaliadas para garantir a correção da atualização. Quando as visões são formadas pela combinação de dados de relações distintas podem surgir casos onde o mapeamento ou a garantia de preservação das restrições não seja possível.

[SJGP90] propõe que as atualizações nas visões sejam garantidas por um sistema de regras. Como a definição da visão é uma coleção de projeções de atributos de relações do banco de dados subjacente, o sistema poderia gerar automaticamente regras que garantissem o mapeamento correto das atualizações.

Em [PMSL94] a visão mantém a mesma estrutura do banco de dados. Qualquer manipulação da visão é mapeada diretamente para os dados reais do banco de dados subjacente. Em [ZGMHW95] a manutenção de consistência ocorre na ordem inversa. Os dados base, quando alterados, notificam o ambiente *warehousing*, que promove as alterações necessárias para garantir a consistência com o banco de dados original.

No caso de sistemas orientados a objeto, [Wie86] propõe enumerar as ambigüidades que as atualizações podem apresentar no mapeamento para o banco de dados subjacente, solucionando-as por meio de uma camada responsável pela atualização de visões. Esta camada procura tirar vantagem da semântica disponível no banco de dados como, por exemplo, as informações sobre níveis de autorização.

Em [Ber92] são introduzidos controles de níveis de autorização tanto no esquema quanto na implementação dos métodos, garantindo que clientes não-autorizados não atualizem o banco de dados. Em [SS89] a atualização baseia-se na utilização dos próprios objetos reais para povoar a visão.

2.2.2 Visão como unidade de reutilização

Uma forma de se obter melhoria em desempenho é a reutilização de visões. A forma de reutilização da visão depende do mecanismo proposto, seu modelo e implementação.

Reutilização de Esquema

Assim como o banco de dados, uma visão possui esquema e conteúdo. O esquema de uma visão é obtido pela projeção do esquema do banco de dados sobre o qual a visão é definida e, às vezes, pelo acréscimo de informações, deduzidas dos dados já existentes. Além disso, a definição de uma visão pode conter diversas informações de manipulação, como por exemplo forma de apresentação, regras de acesso, entre outras. Um mecanismo de visões deveria possibilitar a definição de visões sobre outras visões (*aninhamento de visões*), abreviando o trabalho de definição, teste e validação.

O esquema de uma visão pode ser usado para a definição de outras visões [Sto75, FSdS79, BH88, AKK94, Wie86, HZ88, HZ90, MM91, dSAD93]. Em [HZ88] a reutilização de esquema ocorre na definição de novos contextos de associações entre objetos e métodos. Em [Ber91] é introduzido o conceito de super-visões, isto é, visões utilizadas como base para a criação de novas visões. O conceito de visões é utilizado em [Bra92] como modelo para o relacionamento entre objeto e classe. Neste contexto, a reutilização de esquema ocorre por meio de mecanismos de herança.

[dSAD93] propõe o uso de *visões aninhadas*, isto é, visões que têm como base outras visões. Contudo, a implementação do mecanismo de controle de ativação de visões é feita através de uma estrutura de pilha, onde a base é sempre um banco de dados real e o topo corresponde à visão ativa (à qual o usuário tem acesso). Em virtude dessa implementação, é impossível ativar duas visões simultaneamente, comprometendo a utilização do mecanismo em ambientes multi-usuário.

Materialização

A materialização de visões consiste em tornar permanentes os dados de uma visão. Uma visão materializada pode ser afetada por atualizações sobre ela mesma ou atualizações do banco de dados subjacente. Segundo [GM95], visões materializadas permitem acesso rápido aos dados de uma visão, podem ser utilizadas para implementar *data warehousing* e são alternativas viáveis para implementação de sistemas de tempo real, visualização de dados, verificação de restrições de integridade e otimização de consultas.

O esforço de manutenção de consistência entre a visão e o banco de dados subjacente é chamado de *problema de atualização de visões*. Quando a alteração ocorre no banco de dados e deve ser propagada para a visão materializada o problema de manutenção de consistência é chamado de *manutenção de visões* [LMSS95].

As atualizações de visões podem ser *periódicas* ou *eventuais* ou ainda *completas* ou *parciais* [AGMK95]. Para [SJGP90, LMSS95, GM95], a materialização de visões é uma alternativa para obter melhoria de desempenho. Em [SJGP90] o mecanismo de visões permite a materialização, invalidando-as quando ocorre alguma alteração no banco de dados subjacente.

Já em [ZGMHW95] visões são mantidas materializadas e são gerenciadas através da técnica de *warehousing*. Esta técnica é utilizada para recuperação e integração de dados de fontes distribuídas, autônomas e possivelmente heterogêneas.

Para [GM95] o problema de manutenção de visões pode ser estudado observando-se a quantidade de informações disponíveis para realizar a manutenção da visão (*dimensão de informação*), a abrangência do algoritmo de manutenção (*dimensão de modificação*), a capacidade de expressão da linguagem de definição da visão (*dimensão de linguagem*) e o domínio de dados sobre o qual o algoritmo de manutenção atua (*dimensão de instância*). [GM95] descreve técnicas de manutenção de visões, classificando-as de acordo com a taxonomia apresentada.

A materialização aumenta a redundância e, em algumas propostas, os dados virtuais tornam-se reais. A materialização também pode ser usada para simular evolução de esquema e versionamento de dados [dSAD93].

Versões

Para [Ber92] versões são o resultado de visões materializadas do mesmo esquema. Os conceitos de versões de bancos de dados e visões materializadas são muitas vezes confundidos, tornando sua diferenciação nebulosa. Alguns autores as diferenciam de acordo com as estruturas apresentadas. Uma visão materializada apresenta a mesma estrutura que o banco de dados, enquanto versões apresentam estruturas diferentes. Contudo, uma visão que possibilita a geração de informações pode ser materializada. O resultado da materialização, neste caso, seria uma visão ou uma versão?

Uma diferença básica entre visões materializadas e versões está na sua manipulação. *A priori*, uma visão é materializada quando um usuário deseja tornar persistente aquela instanciação da sua abstração de trabalho. Neste caso, pode-se considerar que sua visão materializada não seria combinada com outras visões ou com o banco de dados, uma vez que isto poderia corresponder a uma outra visão. Esta impossibilidade não é uma proibição formal, é apenas filosófica.

Versões não deveriam duplicar o banco de dados, mas sim garantir que atualizações futuras permitissem a duplicação incremental dos dados. Um mecanismo de duplicação inteligente deveria duplicar e converter os dados somente sob demanda e quando realmente necessário. [dSAD93] propõe que versões sejam obtidas a partir do mecanismo de visões como visões materializadas incrementais.

2.2.3 População das Visões

Uma visão pode ser definida de diversas maneiras, dependendo dos serviços que pretende implementar. Algumas propostas incluem na definição da visão o comando da linguagem de consulta que selecionará o seu conjunto de dados [Sto75, SJGP90, PMSL94, HZ90, AB91, Ber91, dSAD93].

Uma vez que o comando de consulta é processado, um conjunto de dados é selecionado. Este conjunto de dados pode ser replicado virtualmente ou não. Quando os dados não são replicados, o conjunto de dados acessíveis pela visão é real, isto é, através da visão se tem acesso aos dados realmente armazenados no banco de dados. Grande parte das propostas estudadas parte do pressuposto que visões possuem dados virtuais, como pode ser visto na seção 2.2.7. A ausência de dados virtuais pode ser vista em [FSdS79, SS89, Ber92].

Nos mecanismos propostos por [AB91, dSAD93] as visões são povoadas virtualmente através de *objetos imaginários*. Eles são criados durante a ativação da visão e destruídos em sua desativação. Contudo, nenhuma dessas propostas trata da questão de visões atualizáveis.

2.2.4 Interfaces de definição e manipulação

Visões podem ser utilizadas no modo do programador (procedural) e no modo interativo [MM91]. Embora o modo de especificação possa ser declarativo, o aspecto fundamental considerado diz respeito à ausência de uma interface que permita ao usuário interagir com o mecanismo de visões de forma direta, quer seja na especificação da visão, quer seja em sua manipulação.

A definição de visões em modo procedural pode transformá-las em unidades de reutilização, além dos demais serviços já oferecidos. A manipulação procedural é feita na aplicação, onde declarações de visões podem compor o código da aplicação. A definição de visões em modo interativo permitiria ao usuário criar visões sob demanda, sem interferência de administradores ou programadores de aplicação. Da mesma forma, critérios de projeção e particionamento poderiam ser especificados dinamicamente, fornecendo mais flexibilidade ao mecanismo. A manipulação interativa de visões pode ser bastante útil em aplicações que envolvam projetos, prototipações ou planejamento visuais.

O modelo de mapas dinâmicos apresentado em [AKK94] permite a interação com os dados da visão de forma direta. A manipulação dos mapas apresentados é feita através de janelas de parametrização dos dados exibidos, permitindo que o usuário estabeleça as prioridades de sua visão de forma *ad hoc*.

[MM91] apresenta as *hiper-visões* onde à visão é acoplado o mecanismo de visualização que permite ao usuário interagir com a visão. Através da interação com a visão, o usuário pode ter acesso ao banco de dados através da execução de métodos, da navegação e atualização.

2.2.5 Suporte a Orientação a Objetos

Os mecanismos de visões em sistemas orientados a objetos podem ou não estender os conceitos do paradigma, como em [AB91, dSAD93, HZ88, HZ90, SS89, Ber91]. Contudo, as extensões feitas ao paradigma e possíveis efeitos colaterais de definição de visões devem observar e manter características do modelo, como encapsulamento, grafos de composição e herança.

Controle de Hierarquia

As classes de um modelo orientado a objetos são estruturadas hierarquicamente. A hierarquia de classes pode ser representada através de um grafo, onde cada vértice é uma classe e arestas representam a relação de especialização ou generalização entre duas classes. Uma das necessidades em se manter o controle sobre a estrutura de uma hierarquia de classes em sistemas orientados a objetos é a resolução de ativação de métodos. Dependendo do controle de hierarquia utilizado, a localização da implementação de um método em uma classe pode ser ascendente (*upwards resolution*) ou descendente (*downwards resolution*).

Em [HZ88] o mecanismo de herança é implementado através da inclusão da declaração de especialização na definição da visão. Em [Bra92] a hierarquia de herança não permite herança múltipla e é representada por um grafo orientado acíclico. Conseqüentemente, o modelo não prevê a ocorrência de conflitos.

[Ber91] propõe visões que são derivadas de uma classe e nas quais podem ser ocultados atributos e adicionados métodos. Logo, uma visão não pode ser considerada uma subclasse de sua classe base. Assim, a hierarquia de heranças não é afetada pela definição de visões.

As mudanças na hierarquia de classes causadas pela criação de classes virtuais em [AB91] são inferidas pelo sistema através de regras padrão de inferência de tipos. Esta solução é válida quando a hierarquia de herança é representada por um grafo acíclico, isto é, não aceita herança múltipla.

Resolução de Conflitos

Sistemas orientados a objetos permitem que subclasses sejam criadas através do mecanismo de herança. Em ocorrendo herança múltipla, pode acontecer que uma classe herde métodos com nomes iguais, provenientes de superclasses diferentes. Quando um objeto dessa subclasse recebe uma mensagem invocando este nome de método, o objeto não consegue decidir qual método será executado, ocorrendo um conflito chamado por [AB91] de *esquizofrenia*.

Em [HZ88] conflitos decorrentes de herança múltipla são eliminados porque os métodos são também objetos. Assim, cada método possui seu próprio identificador, evitando que conflitos de nomes ocorram.

Objetos com múltiplos papéis

Em sistemas orientados a objetos, um mesmo objeto pode apresentar comportamentos diferentes, dependendo do estado em que se encontra. O suporte a comportamentos diferentes é chamado de suporte a múltiplos papéis.

Em [SS89] múltiplos papéis são suportados por meio de uma extensão ao paradigma de orientação a objetos, onde podem ser definidas múltiplas interfaces para um mesmo objeto. A possibilidade de objetos participarem de múltiplas instanciações de visões de uma mesma classe de visões permite que variáveis sejam instanciadas mais de uma vez. Isso pode ser encarado como a possibilidade de um objeto executar múltiplos papéis simultaneamente.

No modelo de dados FUGUE [HZ88, HZ90] o suporte a múltiplos papéis é feito através de visões. A cada visão definida, um novo conjunto de métodos pode ser associado ao objeto, permitindo que ele apresente comportamentos diferentes.

[Bra92] apresenta um mecanismo de habilitação de múltiplos papéis (ou seja, visões) através do uso do conceito de versões. Neste caso, diferentes *versões* do mesmo objeto podem apresentar comportamento e propriedades diferentes.

2.2.6 As funcionalidades de visões

Algumas funcionalidades dos mecanismos de visões estão diretamente ligadas ao conceito de visões, como a re-estruturação de dados baseada em seleção e projeção e a utilização de população virtual. Outras funcionalidades, como a reutilização de esquema e a materialização de dados podem ser adicionadas ao mecanismo sem utilizar as funcionalidades já existentes para sua obtenção. Estas funcionalidades são chamadas nesta dissertação de *funcionalidades originais*, as quais podem, se combinadas, auxiliar o mecanismo de visões a oferecer outras funcionalidades, aqui chamadas de *funcionalidades adicionadas* (figura 2.1).

As funcionalidades presentes em um mecanismo de visões podem ser influenciadas pela presença de outras funcionalidades (relacionamento *influenciado por*) ou podem ser imprescindíveis para a obtenção de uma outra funcionalidade (relacionamento *necessário para*).

Visões tradicionalmente permitem a *re-estruturação de dados* e utilizam dados virtuais (*população virtual*). Estas funcionalidades são necessárias em um mecanismo de visões que permita a *derivação de informações*. Permitindo a *reutilização de esquema*, o mesmo mecanismo pode ser utilizado para *integrar bases de dados heterogêneas*[1], o qual é o primeiro passo para permitir a criação de *centros de dados* (*data warehousing*).

As mesmas funcionalidades que permitem a integração de bases heterogêneas podem ser utilizadas para simular a *evolução de esquema* através de visões, virtualmente ou não.

[1] Estas funcionalidades são necessárias mas não suficientes para permitir a integração.

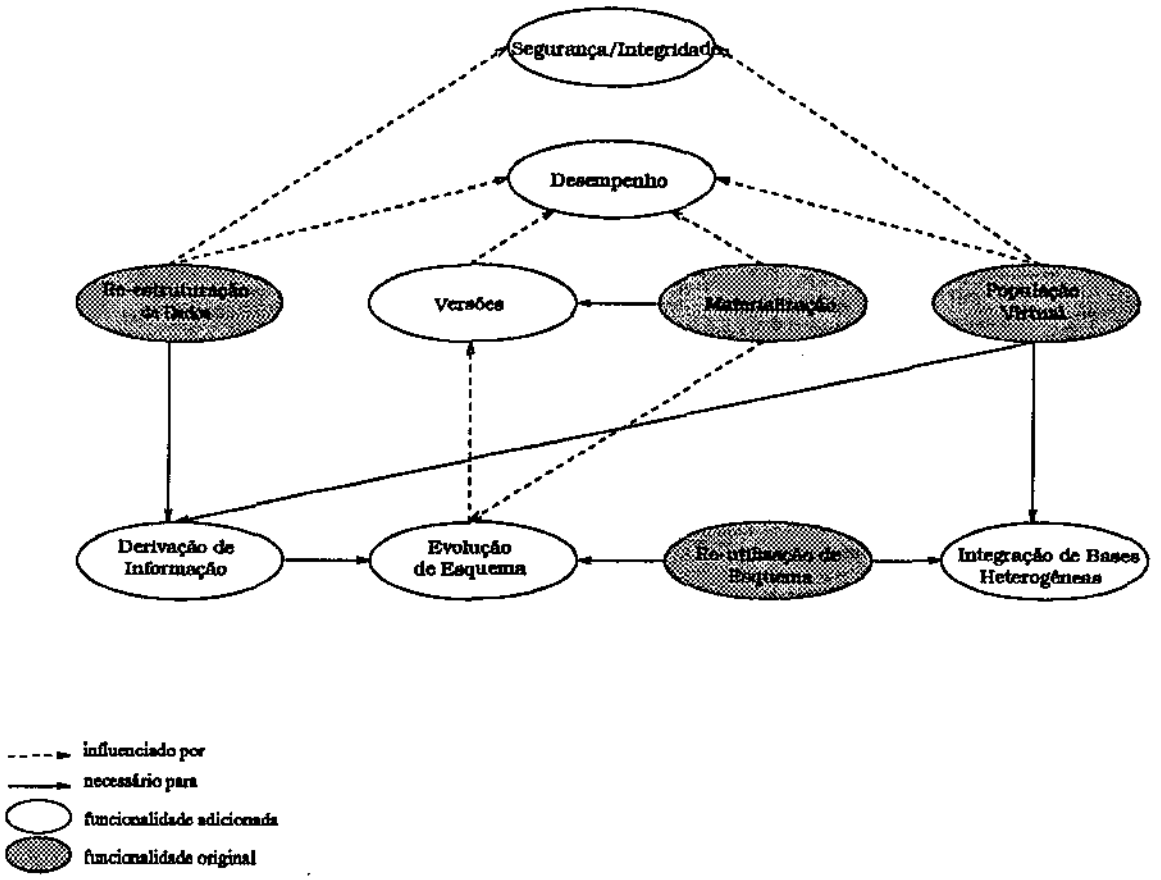


Figura 2.1: O relacionamento entre as funcionalidades oferecidas em visões

Quando utiliza a *materialização*, a evolução de esquema torna-se uma questão semântica, pois a mesma visão materializada que permite tal evolução de esquema pode ser utilizada para criar *versões* do banco de dados.

A manutenção da *segurança* e da *integridade* do banco de dados, assim como o desempenho das aplicações que utilizam visões, são afetadas pela presença de algumas funcionalidades, como população virtual, re-estruturação de dados e materialização. Contudo, nenhuma destas funcionalidades pode necessariamente garantir a segurança, a integridade ou o desempenho.

2.2.7 Classificação dos Artigos

A tabela 2.2 resume os artigos segundo os aspectos apresentados nesta seção. Muitos mecanismos propõem meios para atualização de visões, ainda que com restrições. A reutilização de esquema demonstra a utilização de visões como blocos de construção e torna mais evidente o seu papel como um *banco de dados* de propósito específico. O

string NA siginifica **Não Aplicável.**

Artigo	*Atualiz*	*Reutilização*			*População*		*Interfaces*				*Suporte Orient. Objetos*		
	Consist	*Esquema*	*Materia-*	*Versão*			*Definição*		*Manipulação*		*Mult*	*Resol*	*Contro*
	BD Subj		*lização*		*Real*	*Virt*	*Inter*	*Proc*	*Inter*	*Proc*	*Papéis*	*Conft*	*Hierarq*
[Sto75]	X	X				X		X		X	NA	NA	NA
[FSdS79]	X	X			X			X		X	NA	NA	NA
[BH88]		X				X		X		X	NA	NA	NA
[SJGP90]	X		X			X		X		X	NA	NA	NA
[AKK94]		X				X		X	X		NA	NA	NA
[PMSL94]	X	X				X		X	X		NA	NA	NA
[Wie86]	X	X				X		X		X			
[HZ88]		X			X			X		X	X	X	X
[SS89]	X				X			X		X	X		
[HZ90]		X			X			X		X	X		
[AB91]						X		X		X			X
[Ber91]		X	X	X		X		X		X		X	X
[MM91]	X	X				X		X	X				X
[Ber92]	X				X			X		X			
[Bra92]		X			X			X		X	X	X	X
[dSAD93]		X	X	X		X		X		X			

Tabela 2.2: Caracterização dos mecanismos de visões segundo problemas de implementação

2.3 Visões orientadas a objetos

Tanto a definição do esquema quanto a forma de povoamento de uma visão podem ser obtidas de diversas maneiras. Estas não são mutuamente excludentes e estabelecem relações de compromisso que afetam tanto o poder do modelo quanto o desempenho do mecanismo que o implementa.

A proposta de um modelo de visão orientado a objetos deve levar em consideração uma série de fatores que influenciam o projeto e implementação do mecanismo que dá suporte ao modelo proposto. Em SGBDR o papel de uma visão é agregar as informações semanticamente relacionadas sob o ponto de vista de uma determinada aplicação, pois tais informações estão pulverizadas nas relações do banco de dados. Em SGBDOO as entidades do mundo real podem ser diretamente mapeadas para o modelo de dados do banco de dados, semanticamente mais rico. Mesmo assim, cada aplicação poderá criar esquemas distintos para um mesmo aspecto do mundo real, tendo em vista que a modelagem reflete as necessidades do usuário. Os dados são armazenados segundo tal modelagem e, portanto, é necessário re-estruturá-los caso outra aplicação (com outra visão do mundo) deseje ter acesso a eles.

Alguns autores definem visões orientadas a objetos como *classes* [Bra92, SS89, Wie86, Ber91]. Embora haja diferenças no controle de hierarquia de herança de classes, a idéia

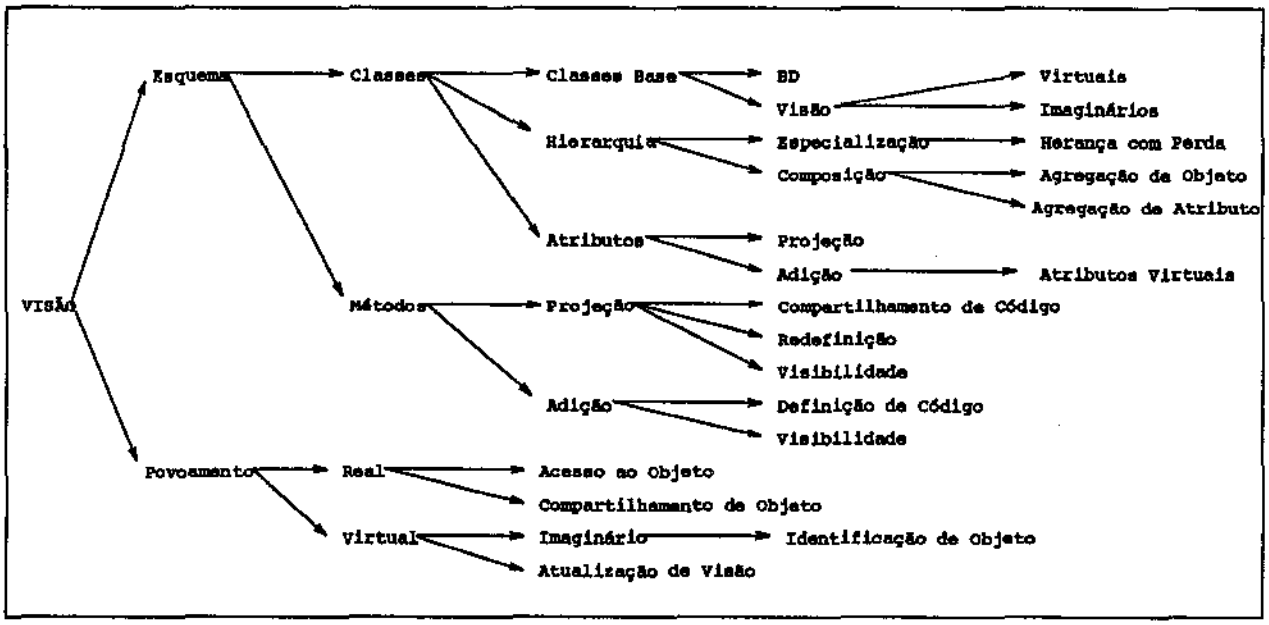


Figura 2.2: Modelo de Visões: fatores de decisão

fundamental compartilhada por estas propostas é que um objeto do banco de dados agrega o máximo de informações inter-relacionadas desejável, possuindo os dados suficientes e necessários para descrever a entidade correspondente do mundo real. Outros autores usam analogia com SGBDR e consideram que o esquema de uma visão é definido da mesma forma que o esquema de um banco de dados [dSAD93, dS95]. Por esta perspectiva, uma visão é um conjunto de classes inter-relacionadas que, juntas, compõem o conjunto de dados necessários à aplicação.

Todavia, quer seja em SGBDR quer em SGBDOO, uma visão é composta por um esquema e um comando de povoamento. A figura 2.2 mostra os fatores de decisão envolvidos no projeto de um modelo de visões, discutidos nas seções seguintes.

Assim como num SGBDR, onde uma visão é constituída a partir de relações do BD, uma visão OO é constituída a partir de classes do banco de dados. Na descrição que se segue, são utilizados os seguintes conceitos:

- uma visão OO é um banco de dados OO (virtual), sendo definida através do par $< esquema, povoamento >$;
- a definição do esquema de uma visão é resultado do seguinte conjunto de operações sobre o esquema do BD subjacente:
  - *seleção*: são selecionadas uma ou mais classes;

- *projeção*: são escolhidos um ou mais atributos de uma classe (que podem, por sua vez, referenciar outras classes);
- *herança*: são criadas novas classes na visão, a partir de herança de classe do BD;
- *composição*: são criadas novas classes na visão, a partir da composição de atributos/classes do BD;
- *definição de atributos virtuais*: são definidos novos atributos para a visão, que não correspondem a atributos do BD;

2.3.1 Esquema de visão orientada a objetos

A definição do esquema de uma visão orientada a objetos depende da definição das classes e dos métodos a elas associados. Como uma visão é criada a partir de um banco de dados, o conjunto de classes utilizado para a definição da visão é o conjunto de classes que compõem o esquema do banco de dados e, ainda, outras visões. O conjunto de classes que é utilizado para a definição do esquema da visão é chamado de *conjunto de classes base* da visão. As classes base podem ser *reais*, *virtuais* ou *imaginárias* [AB91, dSAD93]. A principal característica de uma classe *real* é o fato de que os objetos que a povoam estão fisicamente armazenados. As classes *virtuais* são povoadas por objetos virtuais[2]. As classes *imaginárias* são povoadas por objetos imaginários[3].

Uma visão em SGBDR é uma relação. Nesta dissertação, uma visão em SGBDOO é um banco de dados orientado a objetos. O esquema de uma visão pode ser considerado como o esquema de um banco de dados à parte (como em [dSAD93]) ou pode ser incorporado ao esquema do banco de dados base.

Uma visão também é uma unidade de reutilização, podendo ser utilizada como um bloco de construção para a definição de visões mais especializadas (seção 2.2.2). Se o esquema da visão é considerado como um esquema à parte, o mecanismo de visões deve oferecer meios para que as classes de visão possam ser utilizadas na definição de outras visões. Como se trata de esquemas distintos, a disponibilização de classes de visão ocorre através de algum mecanismo de integração de esquemas. Existem diferentes formas de obter tal integração [dA95a]. No caso de orientação a objetos, um mecanismo recente é a importação/exportação de classes entre esquemas, como em [dSAD93]. As características de um mecanismo de importação/exportação de classes entre esquemas são:

- Duplicação: as classes são duplicadas a cada exportação, sendo copiadas para o banco de dados que as importa;

[2]Objetos que só existem virtualmente durante a ativação da visão.

[3]Não estão fisicamente armazenados. Possuem identificadores determinados a partir de um conjunto de atributos de sua estrutura (chamados *core attributes*).

- Autonomia: com a duplicação, cada esquema é independente dos esquemas que contribuíram para sua definição;
- Consistência: é quebrada. As alterações nas classes importadas/exportadas não produzem efeitos colaterais imediatos pois os esquemas são autônomos. Assim sendo, a manutenção da integridade semântica entre as definições de uma classe em diferentes esquemas exige controle adicional;

Pode ser necessário propagar para os demais esquemas as mudanças ocorridas nas definições das classes importadas, criando um relacionamento de *inter-dependência* entre tais esquemas. Esta dependência gera uma carga adicional no gerenciamento e impõe restrições à evolução dos esquemas. A solução para a manutenção da consistência entre esquemas inter-dependentes em presença de alterações de esquema pode ser encarada como um problema de manutenção de visões. Uma diferença básica entre a manutenção de visões e a manutenção de integridade entre esquemas é que, em visões, na ausência de atualização de visões, a cópia primária é definida como sendo o banco de dados original. No caso de esquemas, pode não ser viável a utilização do conceito de *cópia primária* para as classes. Assim, para cada classe devem ser mantidas informações que indiquem as classes e esquemas dependentes de sua definição.

A integração de esquemas é o primeiro passo para integração de dados, cujos benefícios são claros. Por outro lado, a criação de dependências entre esquemas abre caminho para o surgimento de inconsistências, decorrentes principalmente de sua evolução.

As Classes de Visão

As classes que compõem uma visão podem ser *virtuais* ou *imaginárias*. A utilização de classes virtuais e imaginárias possibilita a re-estruturação de uma classe base e a geração (derivação) de informações na visão, através da projeção dos atributos e da adição de atributos virtuais.

A definição de uma classe de visão consiste em definir como sua estrutura é formada e quais as características assumidas pela classe em virtude de sua formação. Classes de visão são derivadas de classes base predefinidas no banco de dados ou em outras visões. A derivação de visões pode utilizar o mecanismo de herança, de composição ou ambos.

Um mecanismo de herança convencional implementa os relacionamentos de generalização e especialização entre as classes e representa tais relacionamentos por meio de grafos orientados acíclicos. Para atender as necessidades de especificação de classes de visão, o mecanismo de herança deve permitir o *ocultamento* de atributos e métodos da superclasse. Este tipo de herança é conhecido como *herança com perda* ou *ocultamento descendente* [Mey96].

Propostas que utilizam classes virtuais incluem tais classes na hierarquia de classes do próprio banco de dados [AB91, dSAD93, Bra92]. Outras propostas utilizam o mecanismo

de herança com perda para permitir a re-estruturação de uma classe base [Ber91, Ber92, AB91, dSAD93, dS95].

Quando a derivação não se apóia no mecanismo de herança, ela é obtida por composição, isto é, atributos das classes base são utilizados na definição da classe de visão (*projeção de atributos* para a visão). Neste caso, uma forma de representar a estrutura da classe de visão são os *grafos de composição*, os quais armazenam informações sobre os atributos projetados. Classes de visão descritas puramente por grafos de composição não criam relacionamentos diretos do tipo generalização e especialização.

Atributos

Uma das principais utilizações de visões é a re-estruturação da informação armazenada no banco de dados. Logo, é fundamental que seja possível projetar e combinar atributos de classes base em uma classe de visão. Enquanto a projeção é dependente apenas da enumeração dos atributos base desejados, a combinação de atributos exige que seja possível adicionar novos atributos a uma classe de visão. Estes atributos, derivados dos atributos preexistentes, são chamados de *atributos virtuais*.

A inclusão de atributos virtuais em uma classe de visão implica na determinação da sua forma de valoração, isto é, como será atribuído um valor ao atributo que está sendo criado virtualmente. Existem duas soluções básicas para a valoração de atributos virtuais: a avaliação de uma expressão ou a execução de um método ou função que retorne um valor para o atributo. Ambas as soluções são viáveis, embora a primeira (avaliação de uma expressão) possa facilmente ser simulada pela segunda. Em ambos os casos o valor retornado ao atributo é dependente de valores de atributos que podem não fazer parte da visão[4]. Logo, todos os atributos das classes base de uma determinada classe de visão devem poder ser utilizados para a determinação do valor de atributos virtuais.

Métodos

A definição dos métodos pertencentes a uma classe de visão passa pelos mesmos fatores que a definição dos atributos componentes da classe. Aqui chamamos *projeção de métodos* a utilização de métodos das classes base na definição do comportamento das classes de visão.

O compartilhamento de corpos de métodos entre as classes base e de visão é um recurso desejável em um modelo de visões. A decisão de compartilhamento ou não de corpos de métodos depende do sentido de *compartilhamento*. Se no compartilhamento não houver re-definição de métodos, a duplicação do corpo de um método é uma solução aceitável.

[4]Como, por exemplo, no caso da derivação de coordenadas polares a partir de coordenadas cartesianas que estão armazenadas no banco de dados e que não precisam estar na visão.

Se o compartilhamento for real, isto é, apenas um corpo servindo a todas as assinaturas do método, então torna-se necessário a integração de esquemas.

Da mesma forma que atributos, os métodos das classes base podem ser projetados ou não para a visão [dSAD93]. A projeção de métodos das classes base deve permitir que apenas os métodos relacionados aos atributos projetados sejam também projetados na classe de visão, de modo a garantir sua execução. Além dos métodos projetados, a adição de atributos virtuais a uma classe de visão torna necessária a adição de novos métodos à classe, utilizados para a valoração de tais atributos.

2.3.2 Povoamento

A especificação do esquema de uma visão OO consiste na definição da estrutura das classes de visão, isto é, seu grafo de composição e métodos associados aos seus atributos.

O povoamento de uma visão é resultado da avaliação de um comando de consulta que contém um predicado de seleção de objetos. Quando se considera uma visão como uma especialização de uma classe base, toda a *extensão* da classe base compõe a extensão da visão. Isso pode ser encarado como a avaliação de um comando de consulta cujo predicado é verdadeiro para todos os objetos da classe. O povoamento de uma visão composta por um conjunto de classes é dado através da avaliação conjunta dos comandos de consulta para cada uma das classes da visão.

As propostas que encaram visões como classes especiais adicionadas ao banco de dados [Bra92, SS89, Ber92] obrigam que o povoamento da visão seja real, isto é, ao se fazer acesso a um objeto da visão faz-se acesso ao objeto real, que povoa a classe base. As propostas que permitem a utilização de uma população virtual aproximam-se mais do modelo de povoamento de visões em SGBDR, isto é, o povoamento é feito por comandos de consulta. Os comandos de consulta não são necessariamente expressos na linguagem de consulta do banco de dados. Exemplos são a *herança parametrizada* e *invariantes de classe* [AB91, dSAD93].

A especificação dos comandos de consulta consiste na declaração dos comandos de consulta ao banco de dados juntamente com a especificação das fontes de informação, isto é, os bancos de dados[5] sobre o quais a visão será definida. A avaliação dos comandos de consulta seleciona o conjunto de dados da visão. Contudo, a forma como os dados serão acessíveis à visão constitui um problema ortogonal.

População real

Os efeitos colaterais decorrentes de atualizações, do compartilhamento de objetos, do controle de concorrência e, principalmente, da re-estruturação de dados apresentam problemas quando da utilização de população real para o povoamento de uma visão.

[5] Raízes de persistência, no caso de orientação a objetos.

Quando uma visão é definida como uma restrição à extensão da classe base a visão pode fazer acesso aos objetos reais, pois o esquema da visão é igual ao esquema da base. Contudo, isto pode trazer problemas à re-estruturação da informação, o que contraria o próprio propósito de utilizar uma visão. Como seriam adicionados aos objetos reais os novos atributos e métodos inerentes à visões particulares? Como pode ser feita a composição de atributos, ou mesmo de objetos, de maneira a criar uma nova unidade semântica que deve ser disponibilizada à aplicação? A resposta a estas perguntas passa pela criação de meta-informação específica para cada visão. Dessa forma, um objeto de visão seria um objeto virtual com referências a objetos reais, contendo meta-informação necessária para gerenciar o acesso a tais objetos reais.

A possibilidade de povoamento de visões com dados reais permite a *materialização* de visões pois, uma vez que os dados da visão estão materializados não se torna necessário replicá-los virtualmente, com a mesma estrutura. Além disso, a possibilidade de materialização permite ao mecanismo de visões gerar *versões* do banco de dados (seção 4.1.2).

População virtual

O povoamento da visão através de objetos virtuais resolve os problemas de povoamento com objetos reais. Objetos virtuais são aqueles que agregam as informações obtidas junto aos objetos reais. Desta maneira, tanto a agregação de objetos quanto a agregação de atributos são utilizadas para compor o objeto virtual durante a ativação da visão.

O mapeamento de atualizações sobre objetos virtuais para os objetos reais pode causar efeitos colaterais indesejáveis. Contudo, as atualizações feitas nas visões podem, de acordo com a semântica da aplicação, ser mapeadas para as classes base, levando ao problema conhecido como *atualização de visões* [FSdS79, GMS92, LL92, Tom93]. Da mesma forma, os efeitos colaterais de atualizações em objetos reais devem poder ser refletidos nos dados da visão[6], quer ela seja materializada ou não. Este problema é conhecido como *manutenção de visões* [BLT86, GM95, LMSS95, OR92, ZGMHW95].

População imaginária

Com a utilização de população virtual surgem novos fatores a serem observados. A criação de novos objetos, ainda que virtuais, exige a criação de identificadores para tais objetos. Como estes objetos modelam entidades, é importante que a cada ativação da visão os seus identificadores sejam os mesmos de ativações anteriores [HR95, KK95]. [AB91] propõe a utilização de um conjunto de atributos do objeto como argumento para uma função de

[6]Como no caso de visões utilizadas para o monitoramento de desempenho de equipamentos *on-line*, como gerenciamento de carga de trabalho de processadores ou fluxo de mensagens em um ambiente de rede.

criação de identificadores (*core attributes*), criando assim uma espécie de *chave* para o objeto virtual, denominado *objeto imaginário*.

Capítulo 3

Visões em SIG

Este capítulo faz uma revisão de SIG, enfatizando o modelo de dados geográfico, que cria os conceitos de campos e objetos, e as operações suportadas pelo modelo. Além disso, faz a revisão de alguns mecanismos de visões propostos para SIG e identifica quais as necessidades de SIG que serão atendidas através de visões.

3.1 Sistemas Geográficos

Do ponto de vista da comunidade de bancos de dados [MP94], Sistemas de Informações Geográficas (SIG) são sistemas de informação dependentes de bancos de dados que manipulam dados sobre fenômenos geográficos, associados à sua localização física e relacionamentos espaciais [OM95]. Os dados geo-referenciados manipulados por SIG são obtidos de diversas fontes e são capturados por dispositivos diferentes, em formatos distintos. Eles ocupam uma quantidade de espaço de armazenamento considerável e requerem funções de análise e apresentação especializadas, não presentes em sistemas de banco de dados comerciais [MP94].

SIG comportam aplicações urbanas e ambientais [Cif95]. Na primeira, as aplicações de SIG podem ser agrupadas em 2 conjuntos: as que se ocupam do planejamento e as que se ocupam do gerenciamento de algum tipo de rede. No primeiro caso encontram-se a manutenção do cadastro imobiliário, mapeamento urbano básico (MUB), saúde pública, educação, habitação, segurança, zoneamento e crescimento demográfico, entre outras. Dentre as aplicações que se ocupam do gerenciamento de redes estão saneamento básico, energia elétrica, telecomunicações, gasodutos e sistema viário. As aplicações ambientais estão diretamente ligadas ao gerenciamento de recursos hídricos, recurso minerais, fauna e flora e o monitoramento de catástrofes.

[MP94] divide as aplicações de SIG em sócio-econômicas, ambientais e aplicações de gerenciamento (aplicada às outras duas primeiras). Uma outra divisão apresentada baseia-se na escala geográfica com as quais os produtos SIG trabalham: AM/FM (*Automated*

Mapping/Facilities Management) trabalham com escalas na faixa de 1:500 a 1:20.000; SIG para aplicações ambientais trabalham com escalas menores, abaixo de 1:5.000.000.

Quando se utiliza sistemas geográficos depara-se com o conceito de entidade geográfica. Entidades geográficas são entidades do mundo real que possuem características espaciais e se relacionam espacialmente com outros objetos ou entidades geográficas.

Os dados geográficos são caracterizados por sua posição geográfica, seus atributos, suas relações topológicas e seu aspecto temporal [MP94]. Além dos dados convencionais tradicionalmente manipulados por sistemas de bancos de dados comerciais, SIG manipulam dois outros tipos de dados especiais, necessários para se modelar o mundo real. Os dados *espaciais* modelam a geometria e os relacionamentos topológicos das entidades, associando valores à superfície terrestre. Os dados *pictóricos* são utilizados para armazenar a imagem do objeto.

3.1.1 Arquiteturas de SIG

O conjunto de dados não convencionais dos SIG não é suportado por sistemas convencionais de bancos de dados. Por isso, SIG são definidos segundo diversas abordagens [Cif95]. A arquitetura *Toolbox* não diferencia SIG de aplicações estritamente espaciais. Esta arquitetura propõe um conjunto de algoritmos para manipulação de objetos espaciais, não necessariamente geo-referenciados.

A abordagem baseada em *operações* encara um SIG como um sistema de informações tradicional e enfoca as fases de manipulação dos dados geográficos. As operações são divididas em coleta, pré-processamento, gerenciamento, análise e geração de saídas [Cif95]. Existe um esforço por parte da comunidade científica em fazer com que SIG suportem operações não convencionais, como por exemplo análise espacial e topológica. Devido à grande quantidade de dados que SIG manipulam, o mecanismo de processamento de consulta deve ser estendido com índices para otimizar a localização de dados espaciais.

A abordagem baseada em *aplicações* é um refinamento da abordagem orientada a processos e classifica os SIG de acordo com sua aplicação, como por exemplo *sistema de informações ambientais.*

A arquitetura baseada em *banco de dados* considera que SIG são constituídos por bancos de dados não convencionais, capazes de armazenar e manipular dados convencionais, espaciais e imagens. O banco de dados deve possuir mecanismos para suportar eficientemente a manipulação de grandes quantidades de dados espaciais e gráficos e fornecer tipos especiais para representação destes dados. SIG são caracterizados por esta abordagem como uma camada colocada sobre um sistema de banco de dados convencional.

[CCH+96] apresenta uma arquitetura genérica para SIG e três estratégias para implementá-la. A arquitetura é composta de quatro camadas, sobre as quais são desenvolvidas as aplicações e a interface com o usuário. A primeira camada separa a visualização da ma-

nipulação de dados geográficos. A segunda separa as componentes espacial e convencional dos dados geográficos. A terceira camada é responsável pelo armazenamento persistente e pela manipulação elementar das representações de dados geográficos. A quarta camada implementa as estruturas de armazenamento matricial, vetorial e convencional, separando, ainda, as representações matricial e vetorial das estruturas físicas de armazenamento. A primeira estratégia de implementação para esta arquitetura é a estratégia DUAL, onde os dados convencionais são armazenados em um SGBD, tipicamente relacional, e os dados espaciais e pictóricos são armazenados em arquivos separados, relacionados aos dados convencionais. A duas outras estratégias são baseadas na idéia de integração de armazenamento dos dados geográficos em SGBD não convencionais. A primeira delas é baseada na utilização de SGBD com suporte a *campos longos* e a segunda, baseada na utilização de SGBDR *extensíveis*.

Sistemas geográficos podem ser implementados sobre bancos de dados distintos: relacionais, relacionais estendidos e orientados a objetos. Acredita-se que o paradigma de orientação a objetos ofereça um ambiente mais propício para SIG devido, principalmente, à possibilidade representar as entidades do mundo real diretamente no modelo conceitual. Além disso, a forte característica de reutilização e extensibilidade contribuem para atender aos requisitos dos usuários de SIG, cujas necessidades são distintas e carecem de definições mais precisas. Contudo, um SIG deve possuir um modelo de dados capaz de representar dados convencionais e geográficos e ser capaz de expressar os relacionamentos entre estes dados.

3.1.2 Modelos de Dados Geográficos

Modelos de dados são abstrações conceituais que permitem a representação de entidades do mundo real. Em SIG, dois tipos básicos de modelos de dados são utilizados: o modelo de *campos* e o modelo de *objetos*.

No *modelo de campos* o mundo real é visto como um substrato contínuo, sobre o qual são mapeados fenômenos geográficos, isto é, um *campo* pode ser encarado como uma função contínua cujo domínio é geo-referenciado e o contra-domínio corresponde ao mapeamento do domínio a valores que representam o fenômeno geográfico, cujas fronteiras são nebulosas.

No modelo de campos, cada localização está relacionada a apenas um valor descritivo do fenômeno geográfico, formando um conjunto de regiões disjuntas cuja união determina o campo. Esta característica é chamada de restrição de *preenchimento do plano* [CCH+96]. Um campo que representa as relações topológicas e espaciais de um determinado fenômeno de uma região é chamado de *camada temática* do fenômeno sobre a região. O modelo de campos é um modelo mono-temático normalmente empregado para representar fenômenos como tipos de solos, temperatura, altitude e vegetação, entre outros.

Tipicamente, os dados do modelo de campos são representados em formato *varredura* (representados matricialmente). A cada célula da matriz é atribuído um único valor, correspondente ao *tema* dominante na célula. Os formatos de célula mais comumente utilizados são retângulo e quadrado, sendo este último conhecido como formato *raster*.

O *modelo de objetos* enfoca os objetos geográficos, suas características e suas relações topológicas com as demais entidades geográficas. Neste modelo, o mundo é encarado como um conjunto de entidades geo-referenciadas bem definidas, cuja existência independe do fenômeno geográfico em estudo. Uma característica de objetos é que sua identificação é anterior ao conhecimento de sua localização geo-referenciada[1]. Objetos geográficos distintos podem possuir a mesma localização e não obedecem à restrição de preenchimento do plano. *Objetos* são utilizados para representar entidades geo-referenciadas tais como construções humanas (cidades, pontes, monumentos) e abstrações que tenham limitação espacial bem definida (divisão política, prédios, ruas, redes hidrográficas).

O modelo de objetos representa os dados através de pontos, linhas e polígonos, num formato denominado *vetorial*, que utiliza listas de pares ordenados para determinar a localização geográfica do dado representado. Neste formato, a geometria dos objetos pode ser representada mais precisamente e diversos atributos podem ser associados a cada objeto, caracterizando o modelo de objetos como *multi-temático*, em contraste com o modelo de campos. Exemplos deste tipo representação são o modelo total, o modelo *spaghetti*, o modelo topológico e o modelo DIME.

O modelo *total* é utilizado para representar dados vetoriais. Sua unidade básica de representação é o polígono. Cada polígono é representado por um conjunto de coordenadas e os dados convencionais referentes a cada polígono são armazenados separadamente. Coordenadas de polígono adjacentes são armazenados redundantemente e relacionamentos topológicos não são representados. O modelo *spaghetti* é semelhante ao modelo total, mas armazena dados de ponto, linha e polígono. O modelo *topológico* armazena uma matriz de ponto e uma matriz de linhas. Na matriz de pontos são armazenadas suas coordenadas. Na matriz de linhas são armazenados seus pontos extremos e os polígonos à esquerda e à direita. Assim, podem ser derivados relacionamentos topológicos direto da representação.

A estrutura de armazenamento do modelo DIME (*Dual Independent Map Enconding*) é bastante semelhante à do modelo topológico, mas apresenta uma composição hierárquica de representação. Na matriz de linhas podem ser armazenados dados convencionais adicionais. Se os dados convencionais são comuns a outras linhas eles podem ser armazenados em um estrutura separada, para evitar redundância. O modelo *Arc-Node* utiliza uma estrutura para armazenar as coordenadas dos pontos, uma outra para armazenar as linhas (arcos) através de seus pontos extremos. Uma terceira estrutura é utilizada para armazenar os polígonos, os quais são definidos por uma lista de arcos. Novamente, dados convencionais podem ser armazenados na estrutura, como por exemplo área, perímetro,

[1] A Lagoa dos Patos é identificável ainda que não se precise sua geometria e sua localização geográfica.

entre outros. O *modelo de objetos relacional* normaliza o modelo Arc-Node para sua aplicação em bancos de dados relacionais [Cif95].

Grande parte da complexidade de SIG advém da manipulação de dados espaciais, isto é, dados associados a uma geometria. Como visto, a forma de representação geométrica em SIG é não-consensual, fazendo com que diferentes implementações utilizem modelos de representação diferentes. Usuários de SIG enfrentam grandes dificuldades, pois são, normalmente, profissionais de áreas sem afinidade com ciência da computação, e mesmo bancos de dados, que se deparam com a necessidade de manipular representações não convencionais de geometria, geradas a partir de limitações tecnológicas do suporte computacional a SIG.

Uma maneira de amenizar este quadro é possibilitar que o usuário abstraia a representação da informação geográfica que ele está utilizando. [CFS+94, CCH+96] apresentam um modelo onde os dados geográficos são tratados em 4 níveis que guardam independência entre si:

- REALIDADE: os elementos do mundo real, sujeitos à diferentes interpretações;
- CONCEITUAL: oferece ferramentas para modelar formalmente as entidades do mundo real em campos e objetos geográficos, como classes, operações e linguagem de manipulação;
- REPRESENTAÇÃO: associa as entidades modeladas conceituais a formas distintas (e, possivelmente, complementares) de representar campos e objetos geográficos face a fatores relativos à diferentes visões da realidade, como representação em diferentes escalas e projeções cartográficas;
- FÍSICO: define padrões, formas de armazenamento e estruturas de dados para implementar as diferentes representações;

SIG devem possibilitar aos usuários trabalharem no nível conceitual, abstraindo o nível de representação e, principalmente, o nível físico.

3.1.3 Tipos de dados de um SIG genérico

Usuários de SIG enxergam o mundo através das abstrações de campos e de objetos (seção 3.1.2), as quais generalizam categorias de dados geográficos mais específicas.

A figura 3.1 mostra alguns tipos de dados geográficos generalizados pela abstração de campos. A característica comum a estes tipos de dados é que eles representam a variação de um fenômeno geográfico contínuo sobre uma região [Cam95, CCH+96].

Um *mapa temático* é uma sub-divisão geométrica de uma região em áreas homogêneas com respeito a valores de um conjunto de atributos, associando a cada área homogênea um dado conjunto destes valores [AABT94]. *Dados temáticos* modelam fenômenos geográficos

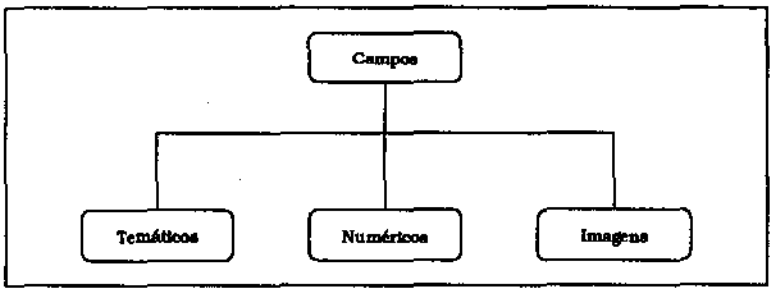


Figura 3.1: Tipos de dados do modelo de campos

que têm como característica particionar a área-alvo do campo geográfico em sub-áreas onde o tema de estudo pertence a uma categoria distinta. O particionamento é total, isto é, a área alvo do campo é igual à união das sub-áreas que compõem o dado temático.

Dados numéricos são discretizações da variação contínua de fenômenos geográficos sobre uma área-alvo. Eles representam os fenômenos geográficos em três dimensões, registrando o valor absoluto do fenômeno geográfico observado para cada localização da área-alvo. Campos numéricos são representados como grades regulares ou triangulares [Cam95, CCH+96]. São normalmente utilizados para representar altimetria, mas podem também ser utilizados para modelar unidades geológicas ou propriedades do solo [CCH+96].

Dados de imagens são matrizes de *pixels* cujos valores são inteiros que representam variações de tons. Imagens são captadas por sistemas de imageamento (radares, satélites).

Para [Cam95], os tipos de dados do modelo de campos são distinguidos por seus domínios. O domínio de um campo *temático* é um conjunto de inteiros enumerável, onde cada elemento do conjunto discrimina uma categoria do tema representado no mapa. O domínio de um campo numérico é o conjunto dos reais, os quais representam o valor do mapeamento do fenômeno geográfico sobre o campo. O domínio de um campo *imagem* é um conjunto discreto de inteiros, cujos valores estão relacionados aos tons que compõem uma imagem.

No modelo de objetos, os principais tipos de dados apontados por [Cam95, CCH+96] são objeto simples e objetos complexos (figura 3.2), os quais podem ser utilizados para modelar Redes. Um *objeto* mantém relacionamentos espaciais (topológicos) com outros objetos geográficos.

Uma *rede* é a representada conceitualmente por um grafo onde cada vértice é um objeto geográfico. As redes acrescentam aos grafos a restrição de *conectividade*. Exemplos deste tipo de dado são redes hidrográficas, redes rodoviárias e redes elétricas.

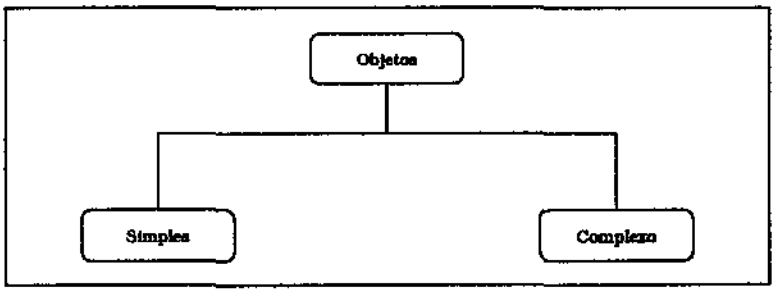


Figura 3.2: Tipos de dados do modelo de objetos

3.1.4 Operações sobre dados geográficos

As aplicações SIG apresentam diferenças entre si desde o nível conceitual devido à grande diversidade de usuários e domínios de aplicação. Em consequência, não há consenso sobre um conjunto base de operações que devem ser tornadas disponíveis para a implementação de aplicações.

Existem diferentes taxonomias de operações em SIG. [Cif95] enumera um conjunto mínimo de operações primitivas (transações), as quais podem ser utilizadas para construir aplicações em SIG genéricos. A descrição das operações primitivas é feita em função das dimensões dos objetos geográficos envolvidos na operação. Tais operações podem ser aplicadas a campos e/ou objetos, como mostra a tabela 3.1.

Transações Primitivas	Variações	Objetos	Campos
Reclassificação			X
Superposição	Convencional		X
	Numérica		X
	Booleana		X
	Análise de Ponderação		X
	Transações baseadas em conjunto	X	
Decomposição	Fronteira	X	
	Interior	X	
Transações	Topológicas booleanas	X	
	Topológicas Escalares	X	
	Busca Topológica	X	
	Escalares	X	X
Buffer (análise de proximidade)		X	X

Tabela 3.1: Transações primitivas sobre dados geográficos

A *reclassificação* permite que categorias de um tema sejam agrupadas, gerando novas categorias. A *superposição* permite a criação de temas formados pela superposição de

outros temas. No novo tema, cada categoria é gerada pela combinação das categorias dos temas superpostos. A operação de superposição possui algumas variantes, dependendo do tipo de algoritmo utilizado.

A superposição *convencional* cria uma nova categoria a partir da sobreposição de duas categorias. A superposição *numérica* é uma sobreposição convencional acrescida de operações numéricas. A superposição *booleana* aplica operações booleanas às categorias superpostas.

A *análise de ponderação* é uma variante da operação de superposição que combina aspectos das superposições convencional e numérica. As categorias do tema superposto são geradas por superposição convencional e seus pesos são obtidos a partir da aplicação de operadores numéricos (por exemplo, adição) às categorias dos temas originais. Na análise de ponderação *simples* as categorias são ponderadas com pesos distintos, normalmente potências de 2. Na *tabelada* os relacionamentos entre as categorias dos temas originais são ponderados e dispostos em uma tabela (categorias tema 1 x categorias tema 2). O peso de cada categoria resultante é obtido buscando a entrada correspondente na tabela de pesos.

Transações baseadas em conjunto são generalizações da superposição booleana. Neste tipo de transação, são aplicadas operações matemáticas de conjunto sobre os objetos geográficos e o resultado é também um conjunto de objetos geográficos.

A *decomposição* aplica-se a polígonos e consiste em separar o interior e a fronteira de um objeto. Esta operação é usada para se determinar a pertinência ou não de um ponto ao interior de um polígono.

Os relacionamentos topológicos entre objetos geográficos são obtidos através das *transações topológicas*. Os relacionamentos topológicos identificados em [Cif95] são cruzamento, interseção, disjunção, adjacência, inclusão e igualdade geométrica. As *transações topológicas booleanas* verificam se um determinado relacionamento topológico existe entre dois objetos geográficos. As *transações de busca topológica* têm como resultado objetos geográficos que apresentam um determinado relacionamento topológico em relação ao objeto topológico especificado. As *transações topológicas escalares* determinam as coordenadas geográficas da ocorrência de um relacionamento topológico entre dois objetos geográficos.

Transações escalares retornam valores escalares referentes às características de um objeto geográfico ou de relacionamentos topológicos entre objetos geográficos, como por exemplo área, comprimento e distância entre dois objetos.

A *análise de proximidade* é efetuada a partir da criação de uma região de *buffer* em torno de uma área analisada, que pode ser interna ou externa à sua fronteira. A aplicação sucessiva desta operação permite a criação de regiões de *buffer* sobre regiões de *buffer*.

Uma outra classificação, proposta por [YA95], aponta 6 conjuntos de operações como blocos de construção básicos para qualquer tipo de tarefa em SIG. Estas operações são

consideradas etapas no processamento de aplicações SIG, independentes do modelo de dados e de domínio (tabela 3.2).

Objetivo das Operações	Operações
Pesquisa / (Re)classificação	Pesquisa temática Pesquisa por Região (Re)-Classificação
Análise de Localização	Buffer Corredor Sobreposição Thiessen/Voronoi
Análise de Terreno	Relevo e Aspecto Catchment/Bacias Rede de drenagem Análise de Visibilidade
Distribuição / Vizinhança	Custos, difusão, espalhamento Proximidade Vizinho mais próximo
Análise Espacial / Estatísticas	Padrões/Dispersão Centralidade Análise Multivariada
Medidas	distância, área

Tabela 3.2: Operações genéricas (blocos de construção)

Enquanto [Cif95] tenta classificar as operações em termos de suas funções, [YA95] mistura operações e aplicações. Assim, na tabela 3.2, a Análise de Localização pode ser obtida através de operações de {*buffer*, corredor, sobreposição, Thiessen/Voronoi}. No entanto, operações de *buffer* generalizam operações de corredor; as operações de sobreposição são as mesmas de [Cif95]; e Thiessen/Voronoi são operações de *buffer* sobre pontos (ao invés de regiões), a partir dos quais são traçados polígonos de Voronoi. Na linha seguinte, Análise de Terreno, a conotação da segunda coluna é outra: trata-se de identificar possíveis objetivos de uma análise de terreno, ao invés de determinar operações. O mesmo tipo de confusão se aplica a todas as entradas da tabela de [YA95], exceto às duas últimas, onde apenas a penúltima linha não é contemplada diretamente por [Cif95].

As operações realizadas sobre dados geográficos podem ser classificadas de maneiras distintas. [Cam95, CCH+96] agrupam as operações de duas maneiras: de acordo com o tipo de resultado e de acordo com o tipo de dado sobre o qual se aplicam. De acordo com o *tipo de resultado*, as operações podem ser:

- de construção: resultam na criação de novos objetos geográficos;
- de atualização: modificam valores dos atributos de objetos geográficos já existentes;

- escalares: retornam valores escalares referentes a propriedades e relacionamentos entre objetos geográficos;
- *booleanos*: retornam valores lógicos referentes a relacionamentos entre objetos geográficos (relacionamentos espaciais);

Para [SE90], a classificação por tipo de resultado é um conceito remanescente de SIG com arquitetura *toolbox* e não condiz com a perspectiva de banco de dados.

Do ponto de vista de *tipo de dado sobre o qual se aplicam*, as operações podem atuar sobre campos, objetos ou ambos [Cam95, CCH+96, Wor95].

A classificação de [Cam95, CCH+96] separa operações sobre campos daquelas sobre objetos, cujas implementações dependem da forma de representação de cada tipo de dado. As operações sobre campos (temáticos, numéricos e imagem) são divididas em operações pontuais, de vizinhança e zonais. Uma operação *pontual* é aquela cujo valor mapeado para um ponto do campo resultante é obtido através da análise de apenas um ponto no campo original. Em uma operação *de vizinhança* o valor mapeado para um ponto do campo resultante depende da análise de um conjunto de pontos do campo original, normalmente adjacentes. Operações *zonais* são casos especiais de operações de vizinhança onde a vizinhança de um ponto é determinada por uma região pré-definida, como uma categoria de um tema. A tabela 3.3 mostra as operações primitivas definidas por [Cif95] e que se aplicam a campos *temáticos*.

Operações	Tipo
Reclassificação	Pontual
Superposição	Pontual
Buffer	Pontual

Tabela 3.3: Operações sobre campos temáticos

A tabela 3.4 apresenta algumas operações genéricas sobre campos numéricos [CCH+96]. Estas operações são dependentes do fenômeno geográfico modelado. As operações de mínimo, máximo, médio e modal de [Cam95] combinam as transações de *buffer* e escalares de [Cif95]. Operações de declividade, exposição e visibilidade são do tipo Análise de Terreno de [YA95]. Estas operações, e a interpolação espacial, são resultado da combinação das transações topológicas e *buffer* de [Cif95], aliadas a funções de interpolação e triangulação de terrenos.

Imagens normalmente são pré-processadas antes de serem utilizadas em aplicações geográficas. As operações sobre imagens são divididas em dois grupos [CCH+96][2]:

[2] A comparação das tabelas 3.1, 3.3 e 3.4 mostra que o processamento de imagens não é tratado por [Cif95]. O mesmo acontece em [Cam95].

Operações	Tipo
Fatiamento	Pontual
Volume	Vizinhança
Superfície	Vizinhança
Mínimo	Vizinhança
Máximo	Vizinhança
Médio	Vizinhança
Modal	Vizinhança
Interpolação Espacial	Vizinhança
Declividade	Vizinhança
Exposição	Vizinhança
Visibilidade	Vizinhança
Interpolação	Vizinhança
Triangulação	Vizinhança
Agregação	Vizinhança

Tabela 3.4: Operações sobre campos numéricos

- Transformações radiométricas: os valores dos *pixels* da imagem são alterados sem modificar a geometria da imagem;
- Transformações geométricas: a geometria da imagem é alterada;

A tabela 3.5 resume as principais operações realizadas sobre imagens [CCH+96]. A natureza destas operações é heterogênea. Realce e filtragem são operações que servem tanto para identificar determinadas características quanto para pré-processar uma imagem. Classificação e segmentação permitem identificar objetos a partir de campos (imagens).

Transformação	Operações	Tipo
Radiométrica	Realce	Vizinhança
	Filtragem	Vizinhança
	Segmentação	Vizinhança
	Classificação	Vizinhança

Tabela 3.5: Operações sobre imagens

No modelo de objetos, as classes de objetos básicas de [CCH+96] são os objetos *simples* e *complexos*, como mostra a figura 3.2. As operações que atuam sobre *objetos* são distintas das operações sobre campos, referindo-se em geral à manipulação de sua geometria. Além

disso, por terem identidade própria, objetos permitem um conjunto de operações referentes a relacionamentos espaciais, definidos em termos de sua geometria e topologia.

Alguns autores propõem álgebras espaciais baseadas na utilização de operações referentes a *relacionamentos espaciais* [Gut94]. As operações de relacionamento espacial podem ser divididas em *métricas*, de *orientação* e *relacionamentos topológicos*. A tabela 3.6 mostra exemplos de relacionamentos métricos e a tabela 3.7 relacionamentos de orientação.

Operações *métricas* operam sobre dados cuja semântica está relacionada a algum tipo de medição (referindo-se às transações escalares de [Cif95]). O resultado das operações depende do sistema métrico utilizado na implementação (por exemplo, a distância euclidiana pode diferir da distância no sistema métrico MANHATTAN [Gat91]).

Tipo Relacionamento	Relacionamento	Referência
Métrico	distância	[Cif95, Gut94]
	perímetro	[Cif95]
	comprimento	[Cif95]
	área	[Cif95, Gut94]

Tabela 3.6: Relacionamentos métricos entre objetos geográficos

Operações de *orientação* são aquelas relacionadas ao posicionamento geográfico relativo de dois ou mais objetos. [FD91] enumera um conjunto de 13 relacionamentos espaciais (propostos por [Fre75]) através dos quais seria possível descrever todas as ocorrências de relacionamentos espaciais em duas dimensões. O conjunto engloba relacionamentos métricos e topológicos, além dos relacionamentos de orientação apresentados na tabela 3.7.

Tipo Relacionamento	Relacionamento	Referência
Orientação	ao norte de	[Gut94]
	ao sul de	[Gut94]
	acima de	[Gut94, Fre75]
	abaixo de	[Gut94, Fre75]
	à direita de	[Fre75]
	à esquerda de	[Fre75]
	atrás de (3D)	[Fre75]
	em frente a (3D)	[Fre75]
	entre	[Fre75]

Tabela 3.7: Relacionamentos de orientação entre objetos geográficos

Trabalhos na área de *raciocínio espacial* têm abordado o problema de formalização de relacionamentos espaciais de orientação. [Vie93] propõe o uso da *Mereological Theory* na definição de um formalismo relacional para representar o espaço e raciocinar a seu respeito. [Her93] define um pequeno conjunto de relacionamentos de orientação que, juntamente com os relacionamentos topológicos de [EF91], são utilizados para oferecer um modelo de orientação. A orientação relativa de um objeto ocorre em função de um referencial. [Her93] argumenta que os relacionamentos de orientação mais comumente utilizados são:

- em frente;
- atrás;
- à direita;
- à esquerda;
- à direita-atrás;
- à esquerda-atrás;
- à direita-em frente;
- à esquerda-em frente;

É difícil obter formalizações e interpretações consensuais para os conceitos utilizados na representação de relacionamentos de orientação . Para [Vie93], formalismos relacionais utilizados para descrever o espaço são concebidos como camadas sobre a representação numérica do espaço, na qual estão implícitos muitos relacionamentos espaciais de difícil captura pelas técnicas de raciocínio espacial. Contudo, a utilização dos conceitos de relacionamentos de orientação, ainda que parcialmente interpretados, pode auxiliar usuários de SIG na elaboração de consultas e, consequentemente, na construção e manipulação de cenários.

Relacionamentos topológicos descrevem o relacionamento entre as fronteiras dos objetos [Her93] e são imunes a transformações topológicas como translação, rotação e mudança de escala [Gut94]. Os relacionamentos topológicos são representados em [Cif95] pelas transações topológicas booleanas. [CdFvO93] determina o conjunto mínimo de relacionamentos topológicos necessários para representar a topologia entre objetos geográficos: {*cross, in, touch, disjoint e overlap*}. A determinação de tais relacionamentos é auxiliada por operações para separar a fronteira e o interior das entidades geográficas. A tabela 3.8 mostra os relacionamentos topológicos entre objetos geográficos.

Enquanto as operações de [CdFvO93] são binárias, as transações topológicas booleanas de [Cif95] são orientadas a conjuntos, tornando necessário definir operações distintas para verificar a ocorrência e a não ocorrência de relacionamentos topológicos entre os objetos

Tipo Relacionamento	Relacionamento	Referência
Topológico	*cross*	[CdFvO93]
	in	[CdFvO93]
	touch	[CdFvO93]
	disjoint	[CdFvO93]
	overlap	[CdFvO93]

Tabela 3.8: Relacionamentos topológicos entre objetos geográficos

geográficos envolvidos na transação. As operações para separar a fronteira e interior de objetos geográficos propostas por [CdFvO93] são as variantes da *decomposição* de [Cif95].

As operações que atuam sobre *redes* são operações que atuam sobre grafos, como buscas em largura e profundidade, determinação do menor caminho, do caminho crítico, do fluxo máximo, dentre outras. Outros tipos de operações, como determinação de circuitos eulerianos ou operações sobre redes com arestas ponderadas, são utilizadas para otimização das entidades reais modeladas como redes. A tabela 3.9 apresenta as operações sobre redes [SE90, Wor95, CCH+96].

Classe de Operações	Operações	Tipo
Espaciais	Vizinhança	construção
Conectividade	caminho ótimo	construção
	caminho crítico	construção
	segmentação dinâmica	construção
	caminho mais curto	construção
	Fecho transitivo	construção

Tabela 3.9: Operações sobre redes

Operações *mistas* integram os modelos de objetos e campos, isto é, são operações que utilizam campos e objetos para serem executadas. [CCH+96] mostra exemplos de operações mistas como sobreposição, reclassificação, operações zonais induzidas e geração de mapas de distâncias. Operações de *conversão* atuam sobre um tipo de dado e têm como resultado dados de outro tipo (tabela 3.10).

3.1.5 Classificação das Operações

A definição de visões passa obrigatoriamente pela definição de operações sobre dados geográficos. Nesta dissertação, propomos uma classificação de operações que combina

Operações	Tipo	Entrada	Saída
Fatiamento	Pontual	Numérico	Temático
Agregação	Pontual	Numérico	Temático
Classificação	Pontual	Imagem	Temático
Ponderação	Pontual	Temático	Numérico
Identificação	Vizinhança	Temático	Objetos
Interseção Espacial	Vizinhança	Temático	Objetos
Mapa de distâncias	Vizinhança	Objetos	Temático
Reclassificação por Atributo	Vizinhança	Objetos	Temático

Tabela 3.10: Operações de conversão de tipos

as classificações discutidas aqui, tendo como base a proposta de [Cif95]. O conjunto mínimo de operadores binários topológicos de [CdFvO93] foi acrescentado às *transações topológicas booleanas.*

A classificação aqui proposta segue a divisão de [CCH+96] em operações sobre objetos, campos e mistas mostradas, respectivamente nas tabelas 3.11, 3.12 e 3.13.

Esta classificação, como se verá posteriormente, foi utilizada para definir o modelo de visão proposto nesta dissertação.

3.2 Visões em SIG

Esta seção traz uma revisão bibliográfica de propostas de utilização de mecanismos de visões em SIG e a identificação de necessidades de SIG que podem ser atendidas através de visões.

SIG são fortemente dependentes de SGBD. Em aplicações convencionais de bancos de dados as visões permitem ao usuário abstrair os modelos lógico e físico do banco de dados, concentrando-se nas entidades conceituais que compõem sua aplicação. Esta necessidade é ainda mais forte na presença de dados geográficos, devido à necessidade de poupar os usuários da complexidade e diversidade de representações dos dados.

3.2.1 Revisão Bibliográfica

Pouco existe na literatura sobre mecanismos de visões para SIG. [AKK94] propõe que, através dos *mapas dinâmicos*, bancos de dados geográficos heterogêneos sejam integrados, embora o processamento das visões ocorra em um ambiente homogêneo. [GPST94] propõe um modelo de gerenciamento de grafos persistentes implementados através de bancos de dados. Os chamados *grafos de visões* são aplicados no gerenciamento de dados complexos

Objeto	Operações	Variações	Tipo	Referência
Objetos	Topológicas Booleanas (Binárias)	*cross*	booleana	[CdFvO93]
		in	booleana	[CdFvO93]
		touch	booleana	[CdFvO93]
		disjoint	booleana	[CdFvO93]
		overlap	booleana	[CdFvO93]
	Topológicas Booleanas (Orientadas a Conjuntos)	Adjacência	booleana	[Cif95]
		Não Adjacência	booleana	[Cif95]
		Inclusão	booleana	[Cif95]
		Não Inclusão	booleana	[Cif95]
		Igualdade	booleana	[Cif95]
		Não Igualdade	booleana	[Cif95]
		Inclusão	booleana	[Cif95]
		Interseção	booleana	[Cif95]
		Disjunção	booleana	[Cif95]
		Cruzamento	booleana	[Cif95]
		Não Cruzamento	booleana	[Cif95]
	Transações Escalares (Métricas)	Distância Mínima	escalar	[Cif95, Gut94]
		Área	escalar	[Cif95, Gut94]
		Perímetro	escalar	[Cif95]
		Comprimento	escalar	[Cif95]
		Localização	escalar	[Cif95]
	Orientação	ao norte de	booleana	[Gut94]
		ao sul de	booleana	[Gut94]
		acima de	booleana	[Gut94, Fre75]
		abaixo de	booleana	[Gut94, Fre75]
		à direita de	booleana	[Fre75]
		à esquerda de	booleana	[Fre75]
		atrás de (3D)	booleana	[Fre75]
		em frente a (3D)	booleana	[Fre75]
		entre	booleana	[Fre75]
	Transações Baseadas em Conjuntos	União	construção	[Cif95]
		Interseção	construção	[Cif95]
		Diferença	construção	[Cif95]
	Busca Topológica	Proximidade máxima	construção	[Cif95]
		Adjacência	construção	[Cif95]
		Não Adjacência	construção	[Cif95]
		Contingência	construção	[Cif95]
		Não Contingência	construção	[Cif95]
		Inclusão	construção	[Cif95]
		Não Inclusão	construção	[Cif95]
		Interseção	construção	[Cif95]
		Disjunção	construção	[Cif95]
		Cruzamento	construção	[Cif95]
		Não Cruzamento	construção	[Cif95]
	Topológicas Escalares	Interseção Escalar	escalar	[Cif95]
	Decomposição	Interior	construção	[CdFvO93, Gut94, Cif95]
		Fronteira	construção	[CdFvO93, Gut94, Cif95]
	Geométricas	Translação	construção	[SE90]
		Rotação	construção	[SE90]
		Centróide	escalar	[SE90, Cif95]
	Conversão	Mapa de Distâncias	construção	[Cam95]
		Reclassificação/Atributo	construção	[Cam95]
Redes	Espaciais	Vizinhança	construção	[SE90]
	Conectividade	caminho ótimo	construção	[SE90]
		caminho crítico	construção	[CCH+96]
		segmentação dinâmica	construção	[CCH+96]
		caminho mais curto	construção	[Wor95]
		Fecho transitivo	construção	[Wor95]

Tabela 3.11: Operações sobre objetos

Campo	Operações	Variações	Tipo de Operação	Referência
Temático	Reclassificação	—	construção	[Cif95]
	Superposição	Convencional	construção	[Cif95]
		Numérica	construção	[Cif95]
		Booleana	construção	[Cif95]
		Análise de Ponderação	construção	[Cif95]
	Conversão	Ponderação	construção	[Cam95]
		Identificação	construção	[Cam95]
		Interseção Espacial	construção	[Cif95]
Numérico	Estatísticas	Mínimo	escalar	[Cif95, SE90]
		Máximo	escalar	[Cif95, SE90]
		Média	escalar	[Cif95, SE90]
		Mediana	escalar	[SE90]
		Variância	escalar	[SE90]
		Regressão	escalar	[SE90]
		Correlação	escalar	[SE90]
		Tabulação cruzada	escalar	[SE90]
		Padrões	escalar	[YA95]
		Dispersão	escalar	[YA95]
		Análise *Multivariada*	escalar	[YA95]
	Escalares	Interpolação	construção	[CCH+96]
		Triangulação	construção	[CCH+96]
		Superfície	escalar	[Wor95]
		Volume	escalar	[Wor95]
	Conversão	Agregação	construção	[Cif95]
		Fatiamento	construção	[CCH+96]
Imagem	Transformação Radiométrica	Realce	atualização	[CCH+96]
		Filtragem	atualização	[CCH+96]
		Segmentação	atualização	[CCH+96]
	Conversão	Classificação	construção	[Cif95]

Tabela 3.12: Operações sobre campos

que podem ser modelados por grafos (redes), tais como rede viária e ferroviária.

[Med94] apresenta um mecanismo de visões estendido, chamado *visões geo-referenciadas*, adaptado às necessidades de sistemas de informação geográficas, com funcionalidades não suportadas pelos mecanismos de visão tradicionais oferecidos pelos sistemas geográficos atuais. O mecanismo de visões é apontado como solução para a manipulação de dados geográficos e se concentra em resolver os problemas de integração, derivação de camadas de dados, exibição dos dados existentes em diferentes níveis de abstrações, definição de regiões geográficas virtuais e versões de dados para planejamento.

O modelo de visão estendido proposto em [Med94] é baseado no paradigma de orientação a objetos e considera que visões são compostas de dados virtuais. O mecanismo consiste em definir uma função de generalização espacial aplicada ao resultado da consulta, que gera os dados da visão. Após a conversão dos dados, estes são limitados em uma região, através de operadores especiais. Este mecanismo de visões não trata diretamente da geração de novas informações a partir dos dados já existentes. Ele oferece uma interface procedural de definição e interface interativa de manipulação, sem abordar os aspectos relativos a possíveis conflitos na composição de classes.

[Med94] aborda as particularidades de atualização de visões que contêm dados espaciais

Operações	Variações	Tipo	Referência
Análise de Proximidade	Simples	construção	[Cif95]
	Múltiplo	construção	[Cif95]
	Multi-nível simples	construção	[Cif95]
	Multi-nível múltiplo	construção	[Cif95]
Diversas	Seleção convencional	construção	[Cif95]
	Mudança resolução	construção	[Cif95]

Tabela 3.13: Operações aplicáveis a campos e objetos

e textuais. A atualização de um dado espacial (por exemplo, uma fronteira) de um objeto pode implicar na atualização de outros objetos. Por isso, somente são permitidas atualizações em objetos da visão que possam ser mapeados diretamente para um objeto real, em que os métodos de manipulação do dado sejam visíveis, isto é, estejam disponíveis na visão.

Uma outra utilização de visões é descrita em [AABT94], que propõe uma arquitetura para um sistema de apoio a tomada de decisão para atender às necessidades de usuários de aplicações geográficas ligadas a atividades de planejamento. Segundo o autor, os usuários desses tipos de aplicações têm necessidades que extrapolam a simples geração de mapas. Tipicamente, tais aplicações envolvem a construção de cenários, baseados em consultas geográficas complexas, e a avaliação do impacto de eventos externos sobre cenários. A construção de cenários pode ser vista como a geração de visões e corresponde à seleção de um conjunto de objetos espaciais relacionados à aplicação. A avaliação do impacto de eventos externos é feita através da customização do cenário, isto é, através da aplicação de restrições e transformações (modelos matemáticos) aos objetos geográficos, culminando com a criação de novos objetos espaciais. O objetivo do sistema é a criação e manipulação de novos objetos espaciais que atendam às restrições da aplicação e preservem seus relacionamentos espaciais, que eram implícitos aos objetos originais.

A arquitetura do sistema é composta por um gerenciador de visões. Este gerenciador de visões possui um processador de transformações matemáticas, ligado a uma base de métodos, e um processador de restrições que, acoplado a um módulo geométrico, é responsável pelas consultas espaciais. Segundo o autor, a junção espacial é a operação fundamental empregada pelo módulo e, aliada a um conjunto pequeno de funções geométricas (como interseção e composição) constitui uma plataforma adequada para a definições de operações para as aplicações.

O esquema das visões mantidas pelo gerenciador é uma instanciação de um meta-esquema que modela uma entidade geográfica através de componentes geométricas e descritivas. As entidades geográficas são associadas a um conjunto de relacionamentos topológicos e às ocorrências de tais relacionamentos com outras entidades geográficas. A

modelagem e atividades de banco de dados são expressas em MDBL (Modeling and Database Language) e as restrições em CDL (Constraint Definition Language).

[AABT94] afirma que as restrições aplicadas aos objetos espaciais causam alterações no banco de dados. Contudo, não fica claro a forma de criação das visões. Além disso, o artigo não relaciona as operações geométricas oferecidas pelo módulo geométrico, utilizadas em conjunto com a junção espacial. Os relacionamentos topológicos não são apresentados.

3.2.2 Aporte de visões para SIG

SIG são fortemente dependente de SGBD e visões são utilizadas para criar bancos de dados virtuais. Tendo em vista as necessidades de aplicações geográficas, as sub-seções seguintes enumeram algumas necessidades de SIG que podem ser atendidas através de um mecanismo de visões.

Modelo de campos e objetos

Usuários de SIG enxergam o mundo por meio de duas abstrações : *campos* e *objetos*, conforme a visão contínua ou discreta da realidade. Campos são, em geral, armazenados em estruturas tesserais e objetos em estruturas vetoriais (seção 3.1.2). Embora tais estruturas possam ser armazenadas em BD, SGBD convencionais não permitem a utilização de conceitos de campos e objetos.

Estes conceitos permitem aos usuários abstrair a modelagem conceitual e representações dos dados geográficos no BD. Além disso, campos e objetos comportam operações diferentes e, por isso, a distinção entre os conceitos facilita a manutenção da integridade semântica das aplicações.

Uma forma de oferecer tais abstrações aos usuários de SIG é criá-las através de visões. Em SGBD cuja modelagem conceitual inclui campos e objetos, novos campos e objetos podem ser derivados daqueles existentes, devido à capacidade de re-estruturação de dados oferecida por visões (capítulo 2). Em SGBD que não implementam tais abstrações, novas classes podem ser criadas na visão, permitindo a modelagem de tais abstrações e, ainda, a sua re-estruturação, de forma a adequar a realidade modelada no SGBDOO subjacente às necessidades das aplicações, como por exemplo através de generalizações.

Cenários

Aplicações geográficas limitam as informações do mundo de três maneiras: ou limitam a região geográfica de estudo, ou limitam o conjunto de fenômenos geográficos estudados, ou ambos. Na prática, ambos são limitados e novos campos e objetos podem ser criados (derivados) para atender às restrições impostas por tais limitações.

A limitação da região geográfica de estudo é definida em SIG como a *definição de uma área-alvo* de estudo. Dentro da área-alvo delimitada, as aplicações manipulam o conjunto de dados relevante ao estudo do fenômeno geográfico em questão. A composição dos dados sobre uma área-alvo é chamada de *plano de informação*. Esta dissertação denomina *cenário* o par (planos de informação,área-alvo).

Originalmente ligado à perspectiva de simulação, o conceito de cenário pode ser traduzido, em terminologia de banco de dados, como a especificação de uma visão. Desta forma, o conceito de cenário diz respeito a um conjunto de dados relevante a uma aplicação geográfica e em relação a uma determinada área-alvo, onde podem ser combinados campos, objetos e dados não espaciais.

As aplicações geográficas normalmente combinam diferentes tipos de dados para gerar informações semanticamente relacionadas às aplicações. A criação de um cenário passa necessariamente pela aplicação de funções de manipulação sobre os dados que compõem os planos de informação subjacentes. Com a utilização de um mecanismo de visões convencional para criar cenários, a aplicação das funções de manipulação constitui uma etapa posterior à criação da visão.

Uma maneira de facilitar a criação de cenários é a aplicação das funções de análise sobre os planos de informação antes da composição do cenário, permitindo que as aplicações concentrem-se em sua semântica e não nos esforços de preparação dos dados. Uma forma de se obter tal funcionalidade é um mecanismo de visões para SIG, que possa executar funções de análise sobre os planos de informação antes de utilizá-los para a composição do cenário.

Para representar o conceito de cenários através de uma visão, o mecanismo de visões deve, além de limitar os dados do banco de dados subjacente e re-estruturar os dados geográficos em questão, fornecer informações adicionais (principalmente espaciais) ao usuário da visão, permitindo que os dados presentes no cenário e os resultados das análises realizadas sobre o mesmo sejam corretamente interpretados. Este conjunto de informações adicionais é chamado, aqui, de *contexto*.

A necessidade de contextualizar um cenário é bastante clara em aplicações geográficas onde a forma de *visualização* do cenário ocorre, em geral, por meio de mapas cartográficos. Como exemplo, considere a visão resultante da consulta como "*Quais são as cidades paulistas que possuem um campus de uma universidade estadual e que distam, no máximo, 100 km da capital?*". Se a visão retornar apenas os dados referentes à consulta, o máximo que o mecanismo de visualização pode mostrar é um conjunto de pontos representando as cidades que atendem ao predicado do comando de consulta. Assim sendo, é necessário agregar à resposta informações (espaciais) referentes ao estado e a capital, permitindo ao mecanismo de visualização situar as cidades em um mapa que apresenta o estado (polígono) e a capital (ponto) como referências.

Do ponto de vista de visões, a noção de contexto significa estender o esquema e o

conjunto de dados de povoamento, fornecendo um conjunto extra de informações, as quais não são necessariamente manipuladas pelo usuário da visão.

Consultas exploratórias

Cenários combinam informações de diferentes fenômenos geográficos, criando um conjunto de informações semântica e estruturalmente favoráveis à aplicação de funções de análise geográfica.

Em aplicações de planejamento, simulação e, principalmente, na análise de um cenário, muitas vezes o usuário deseja restringir a área-alvo ou trabalhar com apenas um subconjunto dos fenômenos geográficos. Este procedimento permite o estudo mais detalhado de pequenas regiões ou mesmo a análise incremental da combinação dos fenômenos geográficos que atuam sobre a região. Além disso, estas aplicações constituem um processo iterativo de análise e transformação dos dados do cenários. Esta re-estruturação e limitação de informações que permite a *exploração* do cenário em níveis variados é chamada aqui de *consulta exploratória*.

Visões podem ser utilizadas como blocos de construção para a definição de outras visões, as chamadas *visões aninhadas*, as quais são especializações da visão original. Visões aninhadas podem ser utilizadas para suportar consultas exploratórias em SIG. Em aplicações geográficas a definição de visões aninhadas pode significar o aprofundamento na análise de determinado fenômeno ou na redução da região geográfica em estudo, ou ambos.

Múltiplas representações

Os dados geográficos podem ser tratados de diversas maneiras. A modelagem determina o conjunto de operações às quais os dados podem ser submetidos.

Diferentes aplicações têm necessidades distintas de representação de dados. Para satisfazer tais necessidades o SIG tem duas opções: armazenar todas as possíveis representações dos dados geográficos, ou armazenar apenas uma representação *base* ou *canônica* e derivar as demais representações quando necessário. A escolha entre as soluções é a escolha entre custo de armazenamento (representação base) versus a rapidez de recuperação (armazenamento de todas as representações). Ambas as soluções são difíceis. A primeira, pela dificuldade de se determinar todas as representações necessárias, obter os dados em tais representações e pela complexidade da manutenção da integridade entre elas. A segunda opção esbarra no fato de nem sempre ser possível derivar uma representação a partir de outra.

Do ponto de vista de bancos de dados pode-se considerar diferentes representações como visões da mesma entidade geográfica. Neste caso, o dado base para cada visão é a representação canônica da entidade geográfica, do qual são derivadas as novas visões

(representações) do dado. Contudo, a derivação de representações exige a existência de uma função de derivação, o que nem sempre é possível.

A derivação de representações, apesar dos inconvenientes, não traz o problema de obter os dados nas diversas representações. Além disso, reduz drasticamente o volume de dados armazenados e, a *priori*, simplifica o processo de manutenção de integridade entre versões de representações, uma vez que uma única representação é armazenada.

O conceito de múltiplas representações é sobrecarregado no contexto de SIG. Considerando o modelo de 4 níveis de [CFS+94], este conceito aplica-se ao nível de representação (terceiro nível). Uma entidade conceitual *ocupação de solos*, modelada como campo geográfico, pode ser representada tanto como um campo temático quanto como uma imagem. Este tipo de múltipla representação (no terceiro nível) não oferece mapeamento entre todas as representações, pois é impossível derivar uma imagem a partir de um campo temático. Além disso, as duas representações são complementares e devem poder ser manipuladas simultaneamente. Por outro lado, a representação de um campo temático em formato vetorial ou varredura (nível físico) também pode ser considerado como representações múltiplas do mesmo dado geográfico e reflete uma situação onde a derivação é possível.

Versões

Versões são utilizadas, entre outras aplicações, no gerenciamento de configurações para ambientes de projeto e para gerenciar a evolução temporal de SGBD.

Versões podem ser encaradas como visões povoadas com dados reais sobre as quais não é feita manutenção, ou seja, os dados da versão não precisam estar atualizados em relação ao banco de dados base (seção 2.2.2).

A materialização de visões pode ser utilizada na criação de versões, principalmente em SGBD que não oferecem mecanismos de versões. A criação de versões através da materialização de visões não oferece suporte à evolução incremental do BD, como feito por mecanismos de versionamento. Além disso, mecanismos de visões não oferecem funcionalidades para o gerenciamento de dados versionados.

Capítulo 4

Modelo e Mecanismo de Visões para SIG

Como visto na seção 3.2, um mecanismo de visões para SIG exige flexibilidade, vários níveis de construção (*visões aninhadas*), muitas funções de manipulação de dados virtuais e muitas situações de simulação. Este capítulo apresenta as propostas de um modelo, um mecanismo e uma linguagem de definição de visões que possam ser utilizados em SIG para atender às necessidades apontadas no capítulo 3.

A arquitetura do modelo de visões é orientada a objetos porque orientação a objetos oferece maior flexibilidade, permite a especificação incremental de aplicações e reutilização. Além disso, é semanticamente mais rica, permitindo a captura das características do mundo real diretamente no modelo.

4.1 Funcionalidades de mecanismos de visões

SGDB suportam visões de duas formas: podem ter o conceito de visões embutido ou podem ser estendidos. A figura 4.1 mostra de forma simplificada estas diferenças. No caso de SGBD estendidos, o mecanismo de visões é executado como uma aplicação sobre o SGBD, servindo de interface entre as aplicações que usam visões e o banco de dados [AB91, dSAD93, GPST94, HR95, MMC94, Med94, PMSL94, dS95, TAPR95].

A figura 4.2 mostra a arquitetura de um SGBD genérico onde o mecanismo de visões (embutido) é mostrado à parte. O esquema do banco de dados é descrito através de declarações em DDL[1]. As declarações e especificações escritas em DDL são compiladas e o resultado é armazenado no *dicionário de dados*, que descreve o *esquema* do banco de dados. O acesso às informações é feito através de comandos DML[2], os quais são compilados

[1] Linguagem de Definição de Dados.
[2] Linguagem de Manipulação de Dados.

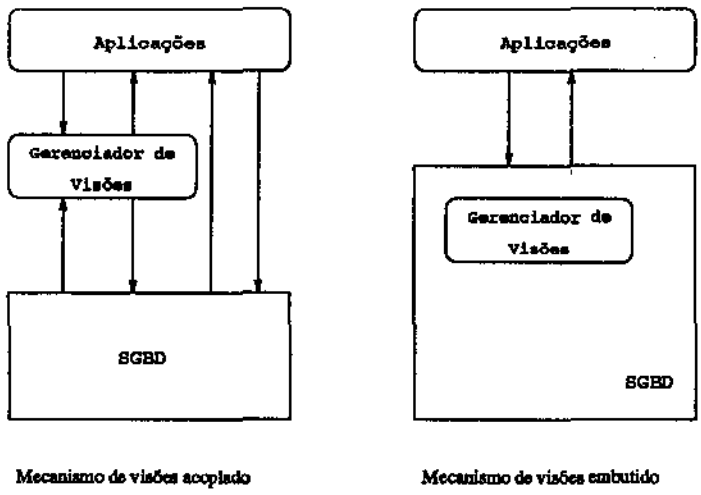


Figura 4.1: SGBD com mecanismo de visões acoplado e embutido

e o resultado da compilação é interpretado pelo processador de consultas[3]. O resultado da operação de consulta é disponibilizado à aplicação (usuário) pelo *gerenciador de buffer*, que gerencia os dados em memória.

No caso de visões, a DDL e a DML são estendidas com comandos de definição e manipulação de visões, respectivamente. O mecanismo de visões utiliza as funcionalidades do SGBD, tais como gerenciamento de *buffer* e controle de concorrência. Do ponto de vista do usuário, a utilização de visões não acrescenta nenhuma complexidade às aplicações. Uma visão é construída através de declarações DDL que permitem a especificação de seu esquema e seus dados. Em SGBDR, a definição de uma visão é feita através de uma consulta, usando uma cláusula especial: por exemplo, CREATE VIEW, seguida de uma cláusula SELECT-FROM-WHERE convencional.

O mecanismo de visões é responsável pela criação do banco de dados oferecido pela visão a partir de um banco de dados base e pela manutenção da consistência entre ambos. Além disso, cabe ao mecanismo de visões oferecer meios para a especificação e manutenção das dependências entre o banco de dados e as visões. A consistência entre o banco de dados base e as visões depende da semântica das aplicações que as utilizam.

A arquitetura de um mecanismo de visões depende da arquitetura do SGBD sobre o qual ele é implementado. A figura 4.3 mostra uma arquitetura para um mecanismo de visões, composta de um repositório de dados, um dicionário de dados e um gerenciador de visões. O *dicionário de dados da visão* é um banco de dados auxiliar que armazena meta-informações relativas à definição das visões, como nome, esquema e comando de consulta.

[3]Comandos DML podem estar embutidos na linguagem de desenvolvimento de aplicações.

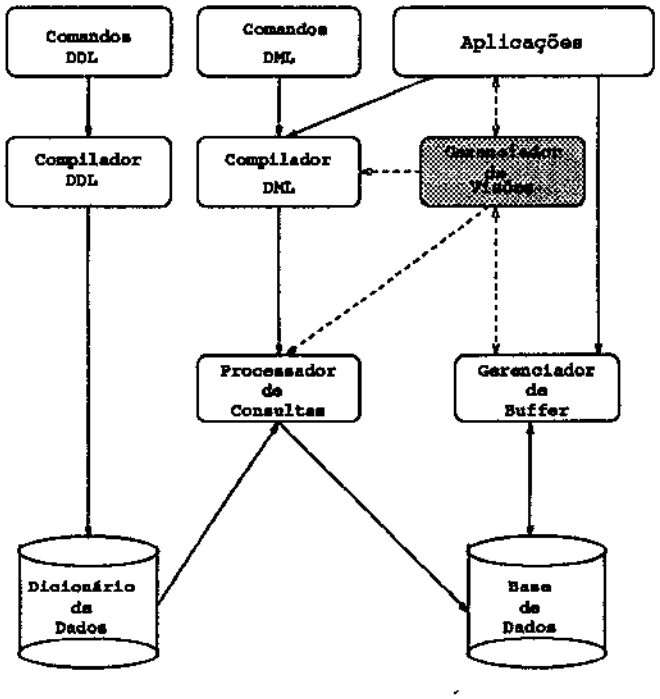


Figura 4.2: SGBD com suporte a visões

O *repositório de dados da visão* é responsável pelo armazenamento temporário dos dados virtuais das visões. O *gerenciador de visões* é responsável pela ativação e desativação da visão, além de gerenciar a interação usuário/visão e garantir a consistência entre os dados das visões e do banco de dados.

A *ativação* de uma visão consiste em criar e disponibilizar aos usuários o banco de dados correspondente à visão, com base nas meta-informações armazenadas no dicionário de dados da visão. Quando a visão não é mais necessária à aplicação, cabe ao gerenciador de visões garantir sua *desativação*, isto é, promover possíveis atualizações no dicionário de dados de visões e eliminar resquícios oriundos da utilização da visão (*garbage-collection*).

A manutenção da consistência entre o banco de dados e as visões depende da propagação para a visão das alterações ocorridas no banco de dados (*manutenção* de visões) e vice-versa, quando a atualização de dados na visão é permitida (*atualização* de visões).

4.1.1 Atualização e Manutenção de Visões

Atualização e manutenção de visões são problemas decorrentes do desacoplamento entre banco de dados e visões de usuário. Ambos os problemas podem ser generalizados como o problema de manutenção da consistência entre visões e banco de dados.

Nem todas as visões podem ser atualizadas. Atualizações em visões podem ou não ser

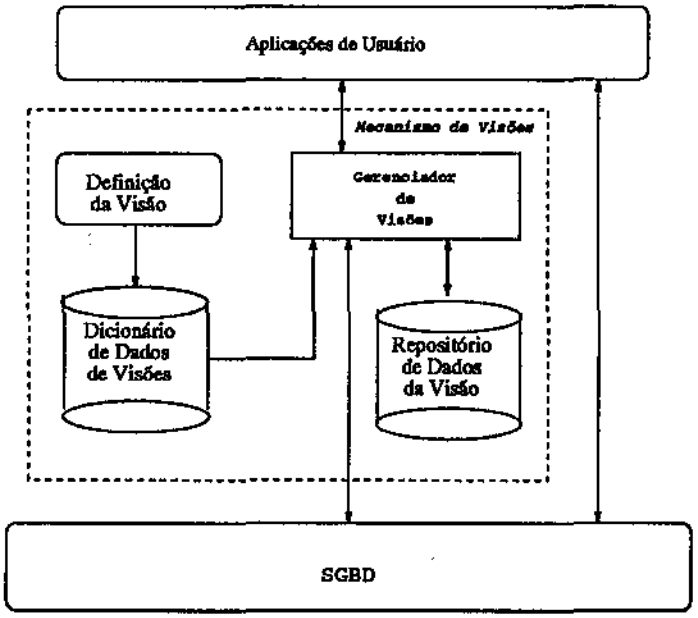


Figura 4.3: Arquitetura de um mecanismo de visões

mapeadas para o banco de dados subjacente, dependendo da semântica da aplicação e da possibilidade de se mapear as atualizações para os objetos reais, que servem de base para definir os componentes dos objetos virtuais.

Em sistemas OO, a atualização de visões só pode ser feita através de métodos disponíveis nas visões. Estes métodos, por sua vez, ou atualizam diretamente os atributos base ou invocam métodos do banco de dados base para atualizar os atributos base.

A manutenção de visões (*refresh*) pode ser feita através da re-avaliação dos comandos de povoamento que criaram a visão. Esta solução, bastante trivial, não oferece bons resultados em relação ao desempenho. Como várias visões podem ser dependentes do dado atualizado, a dependência deve ser registrada, aumentando a quantidade de meta-informação que precisa ser gerenciada pelo SGBD. Além disso, a re-avaliação dos comandos de povoamento envolve dados que não foram afetados pelas atualizações ocorridas. Assim, alguns autores propõem a manutenção apenas dos dados afetados pelas mudanças do banco de dados [BLT86, GMS92, GM95, LMSS95, OR92, ZGMHW95].

As atualizações no banco de dados devem ser monitoradas pelo mecanismo de visões para que possam ser propagadas para as visões que exigem dados atualizados. A garantia de consistência depende da propagação de atualizações, que por sua vez levanta as questões de *quando* e *como* propagá-las. A determinação de *quando* propagar atualizações depende do monitoramento das operações de atualização realizadas no banco de dados e nas visões. A determinação de *como* propagar depende das soluções adotadas pelos mecanismos de atualização e manutenção de visões, um problema que depende semanticamente

das aplicações da visão.

4.1.2 Versões

Versões podem ser encaradas como visões povoadas com dados reais, materializadas, e sobre as quais não é feita manutenção, ou seja, os dados da versão não precisam estar atualizados em relação ao banco de dados base.

Desta maneira, um mecanismo de visões que permite a materialização de dados e o povoamento real também está apto a gerar versões do banco de dados sem, contudo, oferecer funcionalidades do gerenciamento de versões (como manutenção de integridade entre versões).

4.2 O modelo de visões OO proposto - GeoVisões

Como mostrado na seção 2.3, a definição do modelo de visões depende da escolha das soluções para os problemas decorrentes das características de visões. A figura 4.4 mostra as soluções escolhidas nesta dissertação para um modelo de visões orientado a objetos. Em comparação com os fatores de decisão de projeto de modelo de visões orientadas a objeto, apresentados na figura 2.2, foram excluídas as soluções *população real* e *redefinição de métodos*. A primeira, pela necessidade de re-estruturação de dados, conforme discutido na seção 2.3.2, e a segunda por decisão de projeto, já que a definição de métodos na visão supre a funcionalidade oferecida pela re-definição.

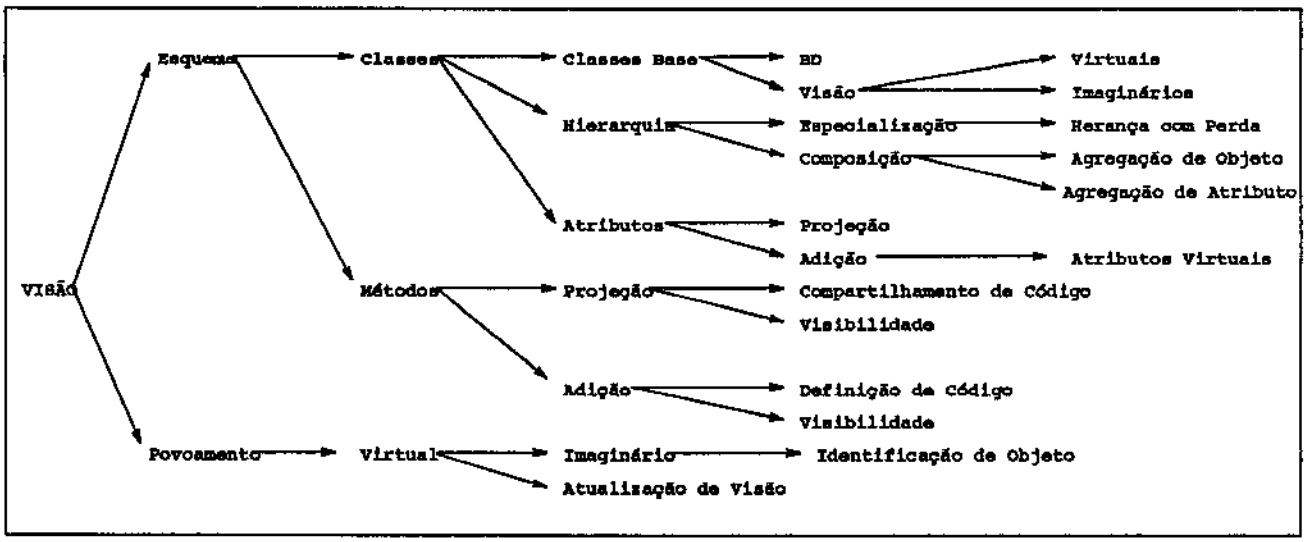


Figura 4.4: O modelo proposto de visões orientadas a objetos

Um banco de dados OO geográfico (BD) é um banco de dados real que modela a espacialidade e o georeferenciamento das entidades e fenômenos do mundo real. Um BD é constituído de um esquema ($\mathcal{E}$), uma extensão (χ) e funções sobre a extensão ($\mathcal{F}$) (BD = $< \mathcal{E}, \chi, \mathcal{F} >$). O *esquema* de BD reflete o mapeamento de uma parcela do mundo real para o modelo de dados de BD e é constituído por um conjunto de classes ($\mathcal{K}$) definidas sobre o conjunto de tipos de BD ($\mathcal{T}$) e organizadas em uma hierarquia ($\mathcal{H}$), a qual é baseada no conceito de sub-tipos e de especialização de comportamento ($\mathcal{E} =< \mathcal{T}, \mathcal{K}, \mathcal{H} >$). Uma classe k ($k \in \mathcal{K}$) é definida através de um *nome*, único no esquema, um *tipo* ($\mathcal{T}_k$), que descreve sua estrutura e um *conjunto de métodos* ($\mathcal{M}_k$), que descreve seu comportamento, e possui uma *extensão* (χ_k), isto é, um conjunto de instâncias (objetos). As classes de BD podem ou não modelar as abstrações de campos e objetos geográficos (seção 3.1.2). O tipo de k ($\mathcal{T}_k \in \mathcal{T}$) é definido por um conjunto de atributos ($a_1 : t_{a1}, a_2 : t_{a2}, ..., a_n : t_{an}$), os quais são formados por um *nome* (a_i), único no tipo, e descritos por um tipo t_{ai}, pertencente ao conjunto de tipos de BD ($t_{ai} \in \mathcal{T}$). Um método m (m $\in \mathcal{M}_k$) é definido por um *nome*, único na classe, uma *assinatura*, que descreve seus parâmetros, e um *corpo*, isto é, um código escrito em uma linguagem de desenvolvimento de aplicações sobre BD.

Um *objeto* é descrito por um *identificador*, único em BD, associado a um *valor*, pertencente ao domínio de $\mathcal{T}_k$. Objetos mantêm dois tipos de relacionamentos com as classes de BD:

- *pertence a*: um objeto pertence a uma única classe k, sobre a qual ele foi instanciado. Objetos que são instâncias de k são ditos *pertencentes* a k;
- *é instância de*: se k_0 é super-classe de k então todos os objetos pertencentes a k são também instâncias de k_0. Contudo, objetos pertences a k_0 não são instâncias de k;

A *extensão* de k (χ_k) é formada pelo conjunto de objetos que são instâncias de k, isto é, objetos pertencentes a k e objetos instâncias de suas sub-classes. A *extensão* de BD (χ) é um conjunto de objetos formado pela união das extensões das classes base de BD.

O conjunto de funções ($\mathcal{F}$) é constituído por funções que não são associadas às classes do BD, mas são aplicáveis aos objetos que constituem sua extensão (χ). No caso de bancos de dados geográficos, χ contém objetos georeferenciados e $\mathcal{F}$ inclui funções espaciais.

Em resumo,

$$\text{BD} = < \mathcal{E}, \chi, \mathcal{F} >$$
$$\mathcal{E} =< \mathcal{T}, \mathcal{K}, \mathcal{H} >$$

Uma visão OO para SIG, ou GeoVisão ($\mathcal{GV}$), é um BDOO geográfico, sendo assim um BDOO geográfico virtual que oferece manipulação de dados geográficos. No caso mais geral, o SGBDOO subjacente não oferece funcionalidades para gerenciamento de dados georeferenciados. Neste caso, a GeoVisão tem a responsabilidade adicional de prover ao usuário as classes virtuais geo-referenciadas (capítulo 6).

Uma GeoVisão é definida através de um esquema ($\mathcal{E}'$), uma extensão (χ'), um conjunto de funções ($\mathcal{F}'$) e um comando de povoamento ($\mathcal{P}$):

$$\mathcal{GV} =< \mathcal{E}', \chi', \mathcal{F}', \mathcal{P} >.$$

Esquema, extensão e funções têm definição análoga à do BDOO subjacente. O comando de povoamento é um conjunto de consultas/programas que, quando executado, *povoa* $\mathcal{E}'$, criando χ'.

Um esquema de uma GeoVisão é um esquema de um BDOO geográfico (definido através de nome, tipos $\mathcal{T}'$, classes $\mathcal{K}'$ e hierarquia $\mathcal{H}'$) :

$$\mathcal{E}' =< \mathcal{T}', \mathcal{K}', \mathcal{H}' >.$$

As classes da visão c ($c \in \mathcal{K}'$) são definidas por um tipo $\mathcal{T}_c$ ($\mathcal{T}_c \in \mathcal{T}'$) e um conjunto de métodos ($\mathcal{M}_c$). e são criadas a partir de um conjunto de classes importado do BD, chamadas *classes base* da GeoVisão. Atributos podem ser definidos na própria GeoVisão (*atributos virtuais*) ou extraídos diretamente das classes base (atributos projetados).

Um atributo *projetado* é um atributo de uma classe base k utilizado na composição da classe de visão c. O valor de um atributo *projetado* é a projeção do valor do atributo original, pertencente à k. O valor de um atributo *virtual* é obtido através da execução de métodos ou funções que retornem um valor do seu tipo. A *extensão* de uma classe de visão (χ_c) é, como em uma classe base, formada pelas instâncias de c.

Um método m de uma classe de visão c (m $\in \mathcal{M}_c$) é definido por um *nome* (único, na classe), uma *assinatura* e um *corpo*. A *assinatura* de um método de visão é composta por um conjunto de tipos $(t_1, ..., t_n, t_0)$, definidos no conjunto de tipos da visão ($t_i \in \mathcal{T}'$), que descrevem os seus parâmetros e o resultado (t_0) de sua execução. O *corpo* de um método é um código escrito em uma linguagem de programação de aplicações que interopere com BD. Como os atributos, métodos podem ter sido *projetados* das classes base ou definidos na própria visão (*métodos da visão*).

Atributos de visão podem ser *atualizáveis* e precisam estar associados a um conjunto de *métodos de propagação de atualizações*, os quais são invocados sobre os objetos reais, pertencentes à extensão de BD, para propagar para o banco de dados as atualizações ocorridas na visão. *Métodos de propagação de atualizações* são métodos das classes base projetados na visão ou métodos da visão definidos em função de métodos da classes base.

Em BD relacionais, o esquema de uma visão é um esquema relacional, com apenas uma relação e o relacionamento entre esta relação e as relações base é definido a partir da expressão de consulta (povoamento da visão). Aqui, GeoVisões podem ter várias classes. Assim, é preciso definir uma forma de especificar o relacionamento entre as classes de visão e as classes base. Denominamos esta forma de *relacionamento de mapeamento* (RM). Cada classe c de visão tem seu RM, que é um conjunto de dados que define, para cada atributo em c, o atributo correspondente em $k \in \mathcal{K}$ do BD subjacente e, para cada método projetado em c, o método correspondente em k.

Os dados do RM de uma classe são utilizados para realizar o seu povoamento, isto é, criar sua extensão, e para gerenciar a propagação de atualizações para BD. A integração do esquema da visão com o esquema do banco de dados depende de um mecanismo de importação/exportação de informações.

Uma GeoVisão é um BD virtual, definido sobre o BD base, que reflete, portanto, o modelo de dados do BD geográfico subjacente. Por ser virtual, a hierarquia de classes ($\mathcal{H}'$) tem como raiz uma classe de visão, chamada GEOVIEW CLASS, que implementa funcionalidades de uma classe virtual (como em [dS95]). Além disso, as classes de uma GeoVisão modelam informações *manipuláveis* e informações de *controle* (figura 4.5). As informações manipuláveis de uma GeoVisão são baseadas no modelo de [CCH+96].

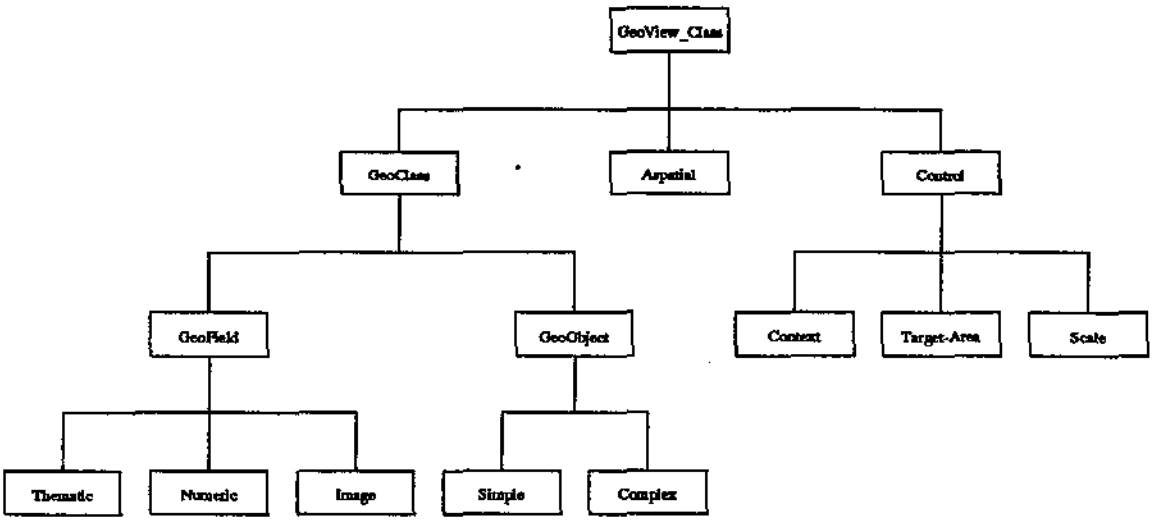


Figura 4.5: Hierarquia de classes de uma GeoVisão

As classes manipuláveis de uma GeoVisão são dispostas nas sub-árvores das *classes geográficas* (GEOCLASS) e das classes *convencionais* (ASPATIAL). As *classes geográficas* modelam campos e objetos geográficos. A classe de geo-campos é especializada em *campos temáticos*, *campos numéricos* e *imagens*. A classe de geo-objetos é especializada em *objetos simples* e *objetos complexos*.

As classes de controle de uma GeoVisão fornecem informações necessárias à composição de cenários e à manipulação espacial. A *área-alvo* da GeoVisão (ω) é definida como um conjunto de coordenadas geográficas utilizado para delimitar a região de estudo. A *escala* geográfica (ρ) determina a escala dos dados utilizados para a composição do cenário e pode ou não ser utilizada como restrição na criação da extensão de uma GeoVisão. O *contexto* de uma GeoVisão (γ) é um conjunto de informações adicionais (principalmente espaciais) não-manipuláveis utilizadas para auxiliar a interpretação das operações realizadas sobre a visão. As informações de γ são obtidas junto ao BD através de comandos de consulta,

restritos pela área-alvo e pela escala geográfica da GeoVisão.

O conjunto de funções de uma GeoVisão ($\mathcal{F}'$) é um mapeamento de $\mathcal{F}$ sobre as classes modeladas na $\mathcal{GV}$ e inclui funções de manipulação de dados geográficos, utilizadas na composição dos cenários. Uma função f ($f \in \mathcal{F}'$) pode ser executada sobre os objetos da GeoVisão e pode ser encapsulada em um método de visão.

Um esquema de visão pode estar associado a diversos comandos de povoamento, os quais são os responsáveis pela obtenção dos dados da GeoVisão, gerando extensões diferentes para um mesmo esquema de GeoVisão. Um *comando de povoamento* ($\mathcal{P}$) está associado a um único esquema e pode ser considerado como uma função que leva a extensão do BD à extensão da $\mathcal{GV}$ ($\mathcal{P}(\chi) \rightsquigarrow \chi'$) e é definido por um nome (único entre os comandos de povoamento), um conjunto de comandos de consulta ($\mathcal{Q}$), referências para as extensões de BD e de $\mathcal{GV}$ e está associado a um único esquema ($\mathcal{P} =< \mathcal{Q}, \chi, \chi', \mathcal{E}' >$).

Um *comando de consulta* q ($q \in \mathcal{Q}$) é composto por uma referência a uma *classe de visão*, uma referência a um conjunto de *classes base* e um conjunto de predicados. Comandos de consulta são utilizados para povoar as classes de visão e para recuperar os dados de contexto referentes à área-alvo. As referências às extensões do BD e da $\mathcal{GV}$ associam as classes, base e de visão, às suas raízes de persistência e são utilizadas para determinar a origem e o destino dos dados resultantes da avaliação do comando de povoamento. A referência ao esquema da GeoVisão ($\mathcal{E}'$) fornece as meta-informações (RM) necessárias para compor sua extensão.

A extensão de uma GeoVisão (χ') é composta por objetos virtuais e é obtida através da avaliação dos comandos de consulta definidos em $\mathcal{P}$ sobre χ. GeoVisões devem manter seus dados consistentes com o BD, através da propagação das atualizaçãoes na $\mathcal{GV}$ para BD e vice-versa. A extensão de uma GeoVisão pode ser materializada, criando ou não versões (seção 4.1.2). A materialização de GeoVisões não será abordada aqui.

4.3 Um mecanismo para GeoVisões

Esta seção apresenta a arquitetura de um mecanismo para GeoVisões e discute as implicações, restrições e vantagens desta arquitetura.

A figura 4.6 mostra a arquitetura proposta para um mecanismo de GeoVisões, baseado na noção de extensão de um SGBDOO. A arquitetura do mecanismo considera que o SGBDOO subjacente não oferece inicialmente funcionalidades espaciais ou de geoprocessamento e, por isso, exige o acoplamento de uma biblioteca espacial para fornecer funções de manipulação de dados geográficos ao mecanismo. Além disso, como o modelo de GeoVisões define uma visão como um BDOO, a arquitetura prevê a utilização, por parte do mecanismo, de todas as funcionalidades do SGBDOO subjacente comumente oferecidas por SGBD, como gerenciamento de *buffer*, controle de transações e mecanismos de recuperação de falhas.

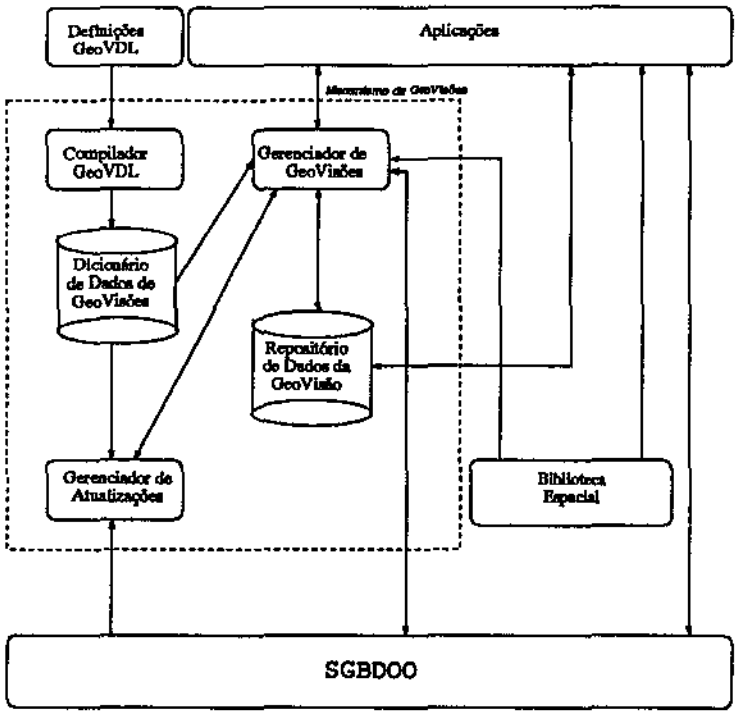


Figura 4.6: Arquitetura para um mecanismo de GeoVisões

A arquitetura do mecanismo de GeoVisões é composta por 3 módulos básicos: o *compilador* GeoVDL, o *gerenciador* de GeoVisões e o *gerenciador de atualizações*. Estes módulos oferecem as funcionalidades de visões apoiados em meta-informações armazenadas no *dicionário de dados de visões*, as quais permitem criar o conjunto de dados das GeoVisões, armazenados no *repositório de dados da GeoVisão* (extensão da GeoVisão). O *dicionário de dados da GeoVisão* contém a descrição do esquema, a especificação dos comandos de povoamento e das funções da GeoVisão.

As seções seguintes descrevem cada um dos componentes desta arquitetura.

4.3.1 O compilador de definições de GeoVisões

O *compilador de definição de GeoVisões* executa papel semelhante ao compilador DDL de um SGBD. Ele é responsável por traduzir a definição de uma GeoVisão para a DDL e a DML nativas do SGBDOO, que devem descrever o esquema e os comandos de povoamento da visão ($\mathcal{GV} =< \mathcal{E}', \chi', \mathcal{F}', \mathcal{P} >$). Neste trabalho supomos que GeoVisões podem ser construídas em cima de SGBDOO que não oferece inicialmente facilidades para manipulação de dados geográficos. Neste caso, uma das atribuições das GeoVisões é oferecer ao usuário facilidades de geo-classes, mesmo que estas não estejam disponíveis no SGBDOO subjacente. Assim, o mecanismo de GeoVisões precisa considerar dois componentes:

- acrescentar ao BD facilidades para gerenciamento de dados geográficos; e
- gerenciamento de dados virtuais.

A linguagem de definição de GeoVisões

O esquema de uma GeoVisão é idêntico ao esquema de um BDOO geográfico. Contudo, a descrição do esquema de um banco de dados OO não se depara com a necessidade de re-estruturar classes predefinidas e as peculiaridades de especificação dos componentes de uma GeoVisão (como área-alvo, contexto e funções de geoprocessamento) não são atendidas pela DDL de um SGBDOO. O compilador DDL não está preparado para gerar as meta-informações necessárias ao gerenciamento de GeoVisões. Os requisitos de especificação de GeoVisões tornam-se ainda mais divergentes em relação à DDL quando se trata do comando de povoamento de uma GeoVisão. Por isso, é necessário a utilização de uma linguagem de definição de GeoVisões que possibilite registrar sua semântica e as diferenças conceituais de seus componentes. No mecanismo de GeoVisões proposto, a especificação de visões é feita através de uma linguagem específica, chamada GeoVDL (*GeoViews Definition Language*).

A GeoVDL oferece meios para a especificação das dependências entre visões e bancos de dados, utilizadas para a manutenção de consistência entre BD e GeoVisões. A especificação do esquema consiste na definição das classes e métodos. A especificação do comando de povoamento consiste na declaração dos comandos de consulta ao banco de dados juntamente com a especificação das fontes de informação, isto é, os bancos de dados (raízes de persistência) sobre os quais a GeoVisão será definida. O mecanismo de GeoVisão deve oferecer meios para a definição, consulta e alterações de especificações em GeoVDL.

O compilador GeoVDL

A definição de uma GeoVisão é feita em GeoVDL. Todavia, uma GeoVisão é um BDOO e é manipulado pelo SGBDOO sobre o qual o mecanismo de GeoVisões foi implementado. Isso torna necessário traduzir as definições em GeoVDL para o modelo de dados e linguagens (DDL e DML) do SGBDOO subjacente.

A análise sintática das declarações GeoVDL é baseada nas definições sintáticas da linguagem. A análise semântica da definição de uma GeoVisão está diretamente relacionada às peculiaridades de geoprocessamento, como por exemplo a definição de uma área-alvo como uma região (polígono fechado). Além disso, outras particularidades são encontradas no ambiente de geoprocessamento como, por exemplo, restrições de escala.

O resultado da compilação de declarações GeoVDL são definições de classes e métodos na DDL do SGBD subjacente. Essas definições são armazenadas no *dicionário de dados de GeoVisões*, juntamente como outras informações extraídas do processo de compilação,

como os relacionamentos de mapeamento (RM) das classes de visões. Os comandos de consulta e manipulação de dados são traduzidos para a DML nativa do SGBD.

4.3.2 O gerenciador de GeoVisões

Esta seção descreve o gerenciador de GeoVisões do ponto de vista das funcionalidades que ele implementa. O *gerenciador de GeoVisões* é responsável pela ativação e desativação de GeoVisões, além da manutenção de consistência entre as GeoVisões e o SGBDOO subjacente. A figura 4.7 mostra as etapas de gerenciamento de uma visão (ciclo de vida de uma visão). Este ciclo de vida se refere às instâncias de GeoVisões enquanto estiverem em estado *ativo*. O gerenciador de GeoVisões divide a responsabilidade da manipulação com o gerenciador de atualizações.

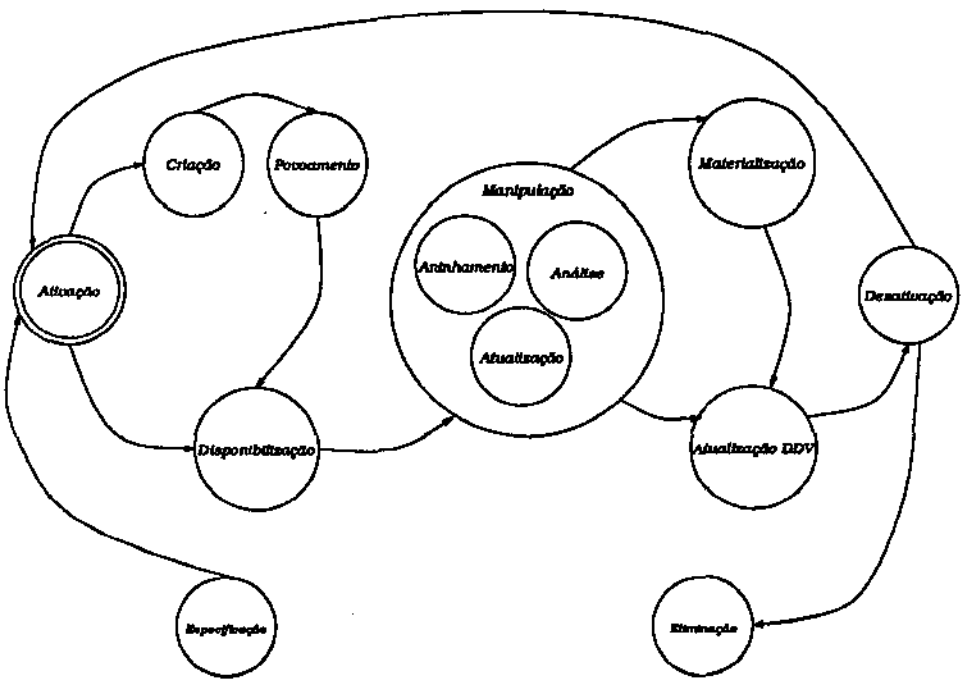


Figura 4.7: Ciclo de vida de uma visão

A ativação de GeoVisões

Uma aplicação ativa uma GeoVisão através da execução de seu comando de povoamento. *Ativar* uma GeoVisão significa criá-la e torná-la disponível às aplicações, isto é, executar um comando de povoamento e criar o BDOO virtual (GeoVisão) correspondente.

Quando a ativação de uma GeoVisão é solicitada, o gerenciador de GeoVisões busca junto ao dicionário de dados de GeoVisões as meta-informações que descrevem o comando

de povoamento e o esquema ao qual ele está associado. No dicionário de dados também estão informações gerenciais, como o estado de sua extensão, isto é, se ela foi materializada ou não. Para uma GeoVisão materializada, o processo de criação se encerra com a localização do nome do repositório que armazena seus dados.

Para uma GeoVisão não-materializada, o gerenciador de GeoVisões utiliza as informações do dicionário de dados da GeoVisão para criar um repositório de dados cujo esquema corresponda ao esquema da GeoVisão e a seguir avalia o comando de povoamento. O comando de povoamento corresponde à execução de consultas às classes base da GeoVisão e o resultado é armazenado no repositório.

Uma vez criada, a GeoVisão é disponibilizada à aplicação. A ativação de uma GeoVisão obedece aos seguintes passos:

1. recuperar o esquema da GeoVisão e seu comando de povoamento, armazenados no dicionário de dados de GeoVisões;
2. determinar a forma de povoamento da GeoVisão, pois GeoVisões que foram materializadas não necessitam mais da execução dos comandos de povoamento;
3. se a GeoVisão não foi previamente materializada então:
   (a) criar um repositório para os dados virtuais da GeoVisão (*repositório de dados da GeoVisão*)
   (b) selecionar as classes base;
   (c) criar o esquema e executar os comandos de povoamento da GeoVisão;
4. se a GeoVisão foi materializada então os passos são:
   (a) garantir sua consistência com o BD base
   (b) disponibilizar o esquema e os dados da GeoVisão
5. tornar o repositório de dados da GeoVisão acessível à aplicação
6. direcionar a aplicação a trabalhar com os dados da GeoVisão

A manipulação de GeoVisões

Durante sua manipulação, uma GeoVisão pode ser *atualizada* ou pode ser utilizada como base para a ativação de uma outra GeoVisão (aninhamento de visões). A *manutenção da consistência* entre o BD subjacente e a GeoVisão, em presença de atualizações, depende do monitoramento de atualizações, realizado pelo gerenciador de atualizações (seção 4.3.3). Como uma GeoVisão é um BD, o esquema de monitoramento utilizado para o BD e as GeoVisões pode ser o mesmo. O gerenciador de atualizações notifica o gerenciador de

GeoVisões sobre quais métodos devem ser aplicados a que objetos para que a consistência entre BD e GeoVisões seja mantida. Uma vez notificado, o gerenciador de GeoVisões deve decidir *quando* realizar a propagação da atualização. Para a propagação das atualizações da GeoVisão para o BD duas alternativas não mutuamente excludentes são:

- periodicamente ou
- durante a desativação

e para propagar as atualizações do BD para a GeoVisão existem três opções:

- periodicamente
- sob solicitação da aplicação (*request for refresh*) ou
- durante sua ativação

O momento de propagação de atualizações deve ser decidido de acordo com semântica das aplicações suportadas pelo BD e pelas GeoVisões e notificadas ao gerenciador de GeoVisões no momento da ativação. Para realizar a propagação de atualizações, cabe ao gerenciador de GeoVisões recuperar o objeto base e executar a atualização sobre ele. Em um ambiente em que ocorram atualizações frequentes, o processo de propagação de atualizações pode ganhar grandes proporções. Para amenizar o impacto causado pela propagação de atualizações, o gerenciador de GeoVisões pode disparar a propagação apenas a determinadas GeoVisões.

A solicitação da ativação de uma GeoVisão durante a utilização de outra GeoVisão é encarada como uma solicitação de aninhamento de visões. Quando esta situação ocorre, a GeoVisão ativa é utilizada como base para ativação da nova GeoVisão. É responsabilidade do programador de aplicações garantir que o esquema e a extensão da GeoVisão base suportam a execução do comando de povoamento da GeoVisão aninhada.

A desativação de GeoVisões

Desativar uma GeoVisão corresponde a eliminar o seu repositório, criado pela ativação. Quando a GeoVisão é desativada, o banco de dados virtual da GeoVisão é destruído ou materializado, conforme solicitação da aplicação. No caso de materialização, o gerenciador de GeoVisões deve atualizar o nome do banco de dados base para o povoamento da GeoVisão (que passa a ser o próprio repositório de dados da GeoVisão materializada).

4.3.3 O gerenciador de atualizações

O gerenciador de atualizações realiza o monitoramento de atualizações junto ao BD e à GeoVisão. Para cada atualização detectada, o gerenciador de atualizações obtém informações sobre os objetos que sofreram alterações, como as classes às quais eles pertencem e os métodos responsáveis por tais atualizações e verifica, no dicionário de dados da GeoVisão, se tais classes (e métodos de atualização) são utilizadas na definição da GeoVisão ativa, e se os objetos satisfazem os predicados dos comandos de consulta. Caso satisfaçam, o gerenciador de atualizações avalia se é necessário re-povoar tais classes e, se for, notifica o gerenciador de GeoVisões.

4.4 GeoVDL - a linguagem de definição de GeoVisões

Esta seção especifica a linguagem GeoVDL, a linguagem de definição de visões para SIG. Ela deve suportar dois contextos relativos à definição de GeoVisões: a definição do esquema e a definição do conjunto de dados que povoará tal esquema (comando de povoamento).

A linguagem GeoVDL é utilizada para traduzir para o mecanismo de GeoVisões as características definidas no modelo de GeoVisões. Visões para SIG (GeoVisões) são utilizadas para criar as abstrações de campos e objetos geográficos, para a definição de cenários e consultas exploratórias (seção 3.2.2). A sua utilização na criação de múltiplas representações, assim como versões, depende das funcionalidades do mecanismo de GeoVisões (seção 4.4.1).

4.4.1 Requisitos da GeoVDL

As funcionalidades dos mecanismos de visões são interdependentes. A GeoVDL deve permitir a expressão de algumas funcionalidades de mecanismos de visões, necessárias para atender às necessidades de SIG, identificados na seção 3.2.2. A tabela 4.1 mostra as funcionalidades de visão necessárias para a criação de GeoVisões.

	Campos e Objetos	Cenários	Múlt. Representações	Cons. Exploratórias	Versões
Re-estruturação	X	X	X		
Ger. de Informações	X	X	X		
Reutilização	X	X		X	
Integração	X	X			
População Virtual	X	X	X	X	
Materialização					X
Atualização	X	X	X		
Funções	X	X	X		

Tabela 4.1: Funcionalidades de visões necessárias à GeoVisões

Como mostrado na seção 2.2.6, a combinação da re-estruturação de dados com reutilização de esquema permite ao mecanismo de GeoVisões simular a *evolução do esquema* do banco de dados. Utilizando a evolução de esquema em conjunto com a materialização, outra funcionalidade pode ser obtida: a criação de versões. Ainda, a re-estruturação de dados em conjunto com a utilização de população virtual favorece a manutenção da segurança e integridade dos dados da GeoVisão (figura 2.1).

Uma vez que o esquema define a estrutura dos dados da GeoVisão, mudanças nos predicados do comando de povoamento podem não comprometer a capacidade do esquema em representar os dados resultantes da avaliação do comando de povoamento. Como exemplo, suponhamos uma GeoVisão que em uma determinada aplicação utiliza como predicado de particionamento a expressão "$altitude > X$". A mudança da expressão para "$altitude > X + Y$" altera o predicado de particionamento e, a rigor, constitui um novo comando de povoamento. Contudo, tal alteração não exige mudanças no esquema da GeoVisão, permitindo que o mesmo esquema possa receber dados de ambos os comando de povoamento. Assim sendo, a separação entre esquema de dados e comando de povoamento permite a reutilização do esquema da GeoVisão (a estrutura do cenário) com comandos de povoamento distintos, acrescentando uma funcionalidade alternativa ao mecanismo de GeoVisões, comparável à *herança parametrizada* [dSAD93].

Cenários são uma agregação de abstrações de geoprocessamento, isto é, campos e objetos geográficos (planos de informação). Para a criação das abstrações de geoprocessamento, a linguagem de definições de GeoVisões deve permitir a re-estruturação dos dados das classes base (seções 2.1.1), importadas de esquemas diferentes (seção 2.1.3), a definição de atributos virtuais (seção 2.1.2) e deve utilizar povoamento virtual (seção 3.2.2), o qual também permite a utilização de GeoVisões na criação de representações alternativas para as entidades geográficas. A geração de tais representações alternativas depende da existência de funções de mapeamento entre representações (seção 3.2.2). A utilização de uma representação em uma aplicação ocorre através da ativação da GeoVisão que a cria.

A utilização de múltiplas representações em um banco de dados leva ao debate *armazenamento versus derivação*. A escolha da forma de manutenção das representações é diretamente dependente da semântica das aplicações que as utilizam e, por isso, deve ser considerada como uma decisão de projeto, tanto das aplicações quanto do banco de dados. Assim, enquanto a população virtual permite a derivação de representações, a materialização (seção 2.2.2) permite que as representações sejam mantidas independentemente no banco de dados.

A renomeação de atributos e métodos confere maior poder de representação semântica da GeoVisão, bem como a determinação do escopo de visibilidade, que também colabora para a manutenção de sua segurança e integridade (seção 2.1.5).

Consultas exploratórias refletem especializações nas abstrações de trabalho dos usuários e são obtidas através da especificação de GeoVisões aninhadas (seção 3.2.2). Para permitir

tal funcionalidade, a linguagem GeoVDL deve possibilitar a reutilização de esquema e dos dados das GeoVisões na definição de novas GeoVisões (seção 2.2.2). Esta funcionalidade permite também a criação de hierarquias de representações, as quais podem ser entendidas como visões aninhadas, em uma granularidade maior.

Aplicações têm diferentes requisitos para a combinação das abstrações de geoprocessamento na composição de um cenário. Tais combinações podem ser obtidas através da aplicação de funções de análise geográfica (seção 3.1.4), na ativação da GeoVisão. Logo, a linguagem GeoVDL deve permitir a expressão de tais funções, as quais podem ser utilizadas para reorganizar os dados do cenário, antes de sua disponibilização ao usuário.

4.4.2 Definição de GeoVisões em GeoVDL

Esta seção apresenta os requisitos necessários para que a linguagem GeoVDL reflita o modelo de visões para SIG (GeoVisões).

Definição de esquema

O esquema de uma GeoVisão ($\mathcal{E}' =< \mathcal{T}', \mathcal{K}', \mathcal{H}' >$) é uma 3-tupla composta de um conjunto de tipos, um conjunto de classes e uma hierarquia de classes, identificados pelo *nome* do esquema de GeoVisão. O conjunto de classes da GeoVisão inclui informações de controle, isto é, contexto (γ), área-alvo (ω) e escala (ρ) (seção 4.2).

O nome do esquema de uma GeoVisão é um *string* de caracteres. O conjunto de tipos da GeoVisão ($\mathcal{T}'$) não precisa ser definido explicitamente através da linguagem, pois é a união do conjunto de tipos do banco de dados base com o conjunto de classes da GeoVisão.

A hierarquia das classes da GeoVisão é gerenciada pelo SGBDOO subjacente. Contudo, cabe ao gerenciador de GeoVisões (seção 4.3.2) fornecer ao SGBDOO informações para a correta determinação da hierarquia.

Uma GeoVisão tem classes geográficas para modelar dados manipuláveis (campos, objetos, dados não espaciais) e classes de controle (figura 4.5). Uma classe é definida por seu tipo e seu comportamento. A estrutura do tipo das classes é definida através da *projeção* de atributos das classes base e da definição de *atributos virtuais*, agrupados em estruturas complexas através da utilização de construtores de conjuntos (*tuple, array, set, list* e *bag*). Os atributos projetados podem ser renomeados e ter seu escopo de visibilidade alterado, permitindo melhor expressão da semântica da GeoVisão. Atributos virtuais são associados a um tipo ou a métodos de GeoVisão, a partir dos quais os tipos dos atributos virtuais podem ser inferidos.

As classes base podem ser agregadas para a definição de classes de visões. Além disso, as classes base podem sofrer o ocultamento de atributos e métodos e a adição de atributos e métodos, simultaneamente. [Ber91] argumenta que a adição de ocultamento de propriedades e comportamento, simultaneamente, descaracterizam os relacionamentos

de especialização e generalização entre as classes envolvidas na re-estruturação. Logo, o mecanismo de herança do SGBDOO subjacente deve ser capaz de suportar, além dos relacionamentos de especialização e generalização, o relacionamento de *herança por visão* [Mey96].

A projeção de atributos depende diretamente da determinação das classes base da classe de visão. Atributos projetados podem ser de tipos distintos: tipos primitivos atômicos, tipos primitivos complexos ou tipos definidos por usuários. Os *tipos primitivos atômicos* são *integer, real, boolean* e *string* e os *tipos primitivos complexos* são *tuple, array, list, set* e *bag*. *Tipos definidos por usuários* combinam tipos primitivos atômicos, tipos primitivos complexos e, possivelmente, outros tipos definidos por usuários e, portanto, são complexos. A importação de classes base é explicitamente feita através de uma lista de nomes de classes, relacionadas a um esquema.

A definição do conjunto de métodos (comportamento) da classe de GeoVisão é semelhante à definição da estrutura de seu tipo. Métodos podem ser projetados das classes base ou definidos na GeoVisão. Os métodos projetados podem ser renomeados, mas mantêm sua assinatura e corpo. A definição de métodos da GeoVisão inclui seu escopo de visibilidade (**PRIVATE** ou **PUBLIC**), seu nome, sua assinatura e seu tipo. A definição dos corpos dos métodos de visão é feita através da inclusão de código escrito em alguma linguagem de programação de aplicações utilizada pelo banco de dados.

Área-alvo (ω) e escala (ρ) estão sempre presentes no esquema de uma GeoVisão e não são re-estruturáveis. Logo, podem ser automaticamente incluídas no esquema da GeoVisão pelo compilador de GeoVisões. Por isso, estas classes somente são referenciadas no comando de povoamento.

Aspectos sintáticos A definição do esquema de uma GeoVisão é dividida em 3 etapas. A primeira é a definição do *cabeçalho* da GeoVisão. A segunda etapa consiste na definição das classes componentes do esquema e a terceira é a definição dos corpos dos métodos de visão, quando cabível. A figura 4.9 apresenta a sintaxe de definição dos corpos do métodos de visão. A figura 4.8 apresenta a sintaxe de definição do cabeçalho e das classes componentes da GeoVisão:

- **nome:** a cláusula **DEFINE VIEW SCHEMA** <View_Schema_Name> permite a definição do nome do esquema. A unicidade do nome deve ser garantida pelo gerenciador de GeoVisões, através das meta-informações armazenadas no dicionário de dados de GeoVisões;

- **integração de esquemas:** a cláusula **IMPORT FROM SCHEMA** <schema_name> **CLASS** <Class_Name_list> permite que as definições de classes base, contidas no esquema de BD possam ser utilizadas na definição das classes da GeoVisão. Devido ao encapsulamento e à existência de tipos complexos (primitivos e construídos por usuários),

```
<View_Schema_Definition> ::=  DEFINE VIEW SCHEMA <View_Schema_Name>
                              <Schema_Importation>
                              <Schema_Structure>
                              [OPERATIONS<Operation_Enumeration>]

<Schema_Importation> ::=      IMPORT FROM SCHEMA <schema_name> CLASS <Class_Name_list>
                              [<Schema_Importation>]

<Schema_Structure> ::=        CLASS <class_name>
                              ISA {[THEMATIC | NUMERIC | IMAGE] FIELD | OBJECT | ASPATIAL | CONTEXT}
                              [GENERALIZES <Class_Name_list>]
                              [INHERITS <Class_Name_list>]
                              <Class_Structuration>
                              [<Schema_Structure>]

<Class_Structuration> ::=     {{PUBLIC | PRIVATE} TYPE {<collections_constructors> | (<type>)}
                              [METHODS <Methods_Projection>]
                              [<View_Methods>]
                              END;}

<Method_Projection> ::=       {[PUBLIC|PRIVATE|]
                              <View_method_name>: <Schema_Name>.<Class_Name>.<Base_Method_Name>
                              [;<Method_Projection>]}

<View_Methods> ::=            {[PUBLIC|PRIVATE] VIEW <View_Method_Name>[(<argument_list>)]:<type>
                              [;<View_Methods>]}

<type> ::=                    {<nonvirtual_type> | <vitual_type>}

<nonvirtual_type> ::=         {<collections_constructors> | <primitive_types> | <class_name>}

<virtual_type> ::=            [<Attributes_Projection>]
                              [<View_Attributes>]

<Attribute_Projection> ::=    {[PUBLIC | PRIVATE | PROTECT]
                              <View_attribute_name>: <Schema_Name>.<Class_Name>.<Base_Attribute_Name>
                              [UPDATABLE BY {<Method_Name>]}
                              [;<Attribute_Projection>]}

<View_Attributes> ::=         {[PUBLIC|PRIVATE] VIEW <attribute_name>:{<Method_Name> | <nonvirtual_type>}
                              [UPDATABLE BY {<Method_Name>]}
                              [;<View_Attributes>]}

<collections_constructors> ::= {tuple|array|list|set|bag}(<type>)

<primitive_types> ::=         {integer|real|boolean|string}

<Operation_Enumeration> ::=   {[<function_name>]
                              [<view_function>]
                              [<Operation_Enumeration>]}

<view_function> ::=           VIEW FUNCTION <function_name>[(<argument_list>)]: <nonvirtual_type>
```

Figura 4.8: GeoVDL- Sintaxe de definição de esquema

o mecanismo deve importar para o esquema da GeoVisão todos os tipos necessários à modelagem da classe base, isto é, todos os tipos que compõem os tipos complexos possivelmente presentes na classe base, os tipos complexos utilizados na modelagem destes tipos complexos, e assim sucessivamente;

- **re-estruturação de classes:** a re-estruturação de classes é resultado da combinação de outros requistos do modelo de GeoVisões, a saber, a *projeção de métodos e atributos*, a *definição de métodos e atributos virtuais* e a *definição do escopo de visibilidade* de atributos e métodos;

- **escopos de visibilidade:** a GeoVDL oferece três níveis de escopo de visibilidade aplicáveis a atributos e dois níveis aplicáveis a métodos:

  - PUBLIC : acesso irrestrito ao atributo ou método;
  - PRIVATE : o acesso ao atributo é permitido somente aos métodos da classe. Os métodos PRIVATE só podem ser utilizados como módulos na definição de métodos da visão (PUBLIC ou PRIVATE);
  - PROTECT : aplicável somente a atributos, os quais são tratados como atributos somente de leitura (READ-ONLY);

- **projeção de atributos:** a cláusula **<Base_Attributes_Projection>** permite que os atributos da classe de visão sejam associados a atributos das classes base. Atributos base de tipos complexos podem ser desmembrados em seus tipos componentes durante a projeção. A ausência desta diretiva pode ser encarada como a utilização na visão de todos os atributos da classe-base em questão;

- **projeção de métodos:** a cláusula **<Base_Methods_Projection>** permite a projeção de métodos, que é feita através da associação de seu nome ao nome pelo qual ele será invocado na classe de visão. A assinatura e o corpo do método não são alterados. A garantia de execução do método é responsabilidade do projetista da GeoVisão;

- **definição de atributos virtuais:** a cláusula **<View_Attributes>** permite a associação de um atributo de visão a um tipo ou a um método de visão. Atributos **atualizáveis** devem ser associados aos métodos responsáveis pela propagação de atualizações, através da cláusula **[UPDATABLE BY <Base_Method_Name>]** . Esta cláusula inclui uma referência ao objeto real no tipo da classe de visão;

- **definição de métodos virtuais:** a cláusula **<View_Methods>** permite a definição de sua assinatura e a cláusula **<View_Method_Body_Specification>** permite a definição dos corpos dos métodos de visão (figura 4.9);

```
<View_Method_Body_Specification> ::=  VIEW METHOD BODY OF <View_Method_Name>
                                      IN VIEW CLASS <View_Class_Name>
                                      OF VIEW SCHEMA <View_Schema_Name> AS
                                      BEGIN BODY
                                      <native_code>
                                      END BODY
```

Figura 4.9: GeoVDL- Sintaxe de definição de corpo de método de visão

- **definição de classes**: é suportada pela re-estruturação das classes base e permite a definição de três tipos de classe:
  - *de controle*: as classes CONTEXT . Atributos de classes de contexto são automaticamente definidos com escopo de visibilidade PROTECT , evitando a ocorrência de efeitos colaterais durante a manipulação da GeoVisão;
  - *geo-classes* que modelam geo-campos (THEMATIC, NUMERIC e IMAGE FIELD) e geo-objetos simples e complexos (OBJECT);
  - *convencionais*, através da cláusula ASPATIAL ;

  As classes de visão podem utilizar várias classes base na sua composição. Tal combinação é suportada por três formas distintas de combinação das extensões base na composição da extensão da classe de visão. Na primeira forma, os valores de um objeto de cada classe base são combinados para a valoração de um objeto virtual, que compõe a extensão da classe de visão. Na segunda forma, indicada pela cláusula GENERALIZES , os objetos das classes base que atendem ao predicado do comando de povoamento da classe de visão são utilizados na valoração de objetos virtuais distintos, fazendo com que as extensões das classes base sejam unidas para formar a extensão da classe de visão. A terceira forma é utilizando especialização (cláusula INHERITS).

  A possibilidade de combinar a extensão de duas classes base em uma mesma classe de visão é um processo normalmente decorrente da generalização conceitual da modelagem do BD. Este processo exige que atributos de classes base diferentes sejam mapeados para o mesmo atributo virtual da classe de visão. Tendo em vista a garantir, tanto quanto possível, a simplicidade da linguagem, a solução escolhida para este tipo de modelagem é permitir, na presença da clásula GENERALIZES , que atributos da GeoVisão sejam multiplamente definidos. Cada definição associa o atributo de visão a um atributo de cada uma das diferentes classes base que estão sendo generalizadas. O gerenciador de GeoVisões é responsável pela execução da política de combinação de extensões;

```
<Function_Body_Specification> ::=  VIEW FUNCTION BODY OF <function_name>
                                   IN SCHEMA <View_Schema_Name> AS
                                   BODY
                                   <native_code>
                                   END BODY
```

Figura 4.10: GeoVDL- Sintaxe de definição de corpo de função

- **definição de funções:** funções são definidas a partir das funções base ou de forma independente. Funções devem ser especificadas junto com a definição do esquema, através da cláusula [**OPERATIONS <Operation_Specification>**] . A GeoVDL contempla a inclusão das funções de geoprocessamento de acordo com a taxonomia de operações sobre abstrações geográficas (seção 3.1.4), separando as funções de acordo com as abstrações às quais são aplicáveis (figura 4.11). A aplicação de funções da visão é suportada pela linguagem através da cláusula **VIEW FUNCTION** . A cláusula **<Function_Body_Specification>** permite a definição dos corpos das funções da GeoVisão (figura 4.10);

```
<GeOperations> ::=  {<Thematic_Field_Operations>|<Numeric_Field_Operations> |
                     <Image_Field_Operations>|<Object_Operations> |
                     <Network_Operations> | <Buffer_Operations> |
                     <GeOperations>}

<Thematic_Field_Operations> ::=  {<Reclassify_Operation>|<Overlay_Operation>|<TF_Conversion_Operations>}

<Numeric_Field_Operation> ::=  {<Statistical_Operations>|<Scalar_Operations>|<NF_Conversion_Operations>}

<Image_Field_Operations> ::=  {<Radiometric_Operations>|<IF_Conversion_Operations>}

<Object_Operations> ::=  {<Spatial_Relationships>|<Set_Based_Operations> |
                          <Topological_Search_Operations> | <Scalar_Topological_Operations> |
                          <Decomposition_Operations> | <Geometric_Operations> |
                          <OM_Conversion_Operations>}

<Network_Operations> ::=  {<Spacial_Operations> | <Conectivity_Operations>}

<Reclassify_Operation> ::=  RECLASSIFY <thematic_field>
                            BY <table>
                            ONTO <thematic_field>

<Overlay_Operation> ::=  OVERLAY <thematic_field>,<thematic_field>
                         ONTO <thematic_field>
                         {ADDING | SUBTRACTING | MULTIPLYING | DIVIDING | WEIGHTING}
```

Figura 4.11: GeoVDL- Operações (funções) de geoprocessamento

Definição do comando de povoamento

Comandos de povoamento são associados a um esquema de uma GeoVisão. Sua definição consiste em um nome, um conjunto de comandos de consulta ($\mathcal{Q}$) e referências para às extensões de BD e de $\mathcal{GV}$ ($\mathcal{P} = \mathcal{Q}, \chi, \chi', \mathcal{E}'$). Além disso, o comando de povoamento deve povoar as componentes da GeoVisão não definidos explicitamente no esquema, como a área-alvo (ω) e escala (ρ).

Um comando de consulta q ($q \in \mathcal{Q}$) de uma classe de visão c deve conter referências às suas classes base e respectivas extensões combinando seleção convencional com seleção espacial. No modelo de campos, as operações possíveis de serem utilizadas para o particionamento de extensão são as operações *estatísticas* e *escalares* do tipo *escalar*[4], aplicáveis somente a campos *numéricos*. Estas operações são, no entanto, operações escalares e não caracterizam um predicado como *espacial*. Para o modelo de objetos, a seleção espacial utiliza as operações referentes a relacionamentos espaciais, às operações definidas nas transações de busca topológica e topológicas escalares. A operações de decomposição, por serem unárias, podem ser aplicadas aos objetos geográficos durante a avaliação do predicado (tabela 3.11). O povoamento permite a construção de abstrações e, consequentemente, do cenário.

O povoamento das classes de uma GeoVisão é definido em 2 etapas:

1. a avaliação de comandos de consultas sobre as classes base de c;
2. a avaliação de expressões, métodos e funções que povoam os atributos virtuais de c;

Aspectos sintáticos A definição de um comando de povoamento para o esquema da GeoVisão é feita em 3 etapas. A primeira etapa é a definição do povoamento dos dados do *cabeçalho* da GeoVisão, o que inclui o povoamento da área-alvo e da escala, não explicitamente definidos em $\mathcal{E}'$. A segunda é a definição dos comandos de povoamento das classes da GeoVisão, compostos pelos comandos de consulta às classes base. A terceira etapa é a definição das funções a serem aplicadas aos dados da GeoVisão. A figura 4.12 apresenta a sintaxe de definição de um comando de povoamento para GeoVisões.

O aparato sintático oferecido pela GeoVDL deve permitir ao projetista da GeoVisão determinar corretamente o conjunto de dados que comporá a extensão da GeoVisão. Os ítens abaixo expõem a forma como cada um dos requisitos de dados da GeoVisão é atendido pelo comando de povoamento:

- **nome:** a cláusula **DEFINE VIEW <View_Extension_Definition_Name>** permite a definição do nome do esquema de povoamento. A unicidade do nome deve ser garantida pelo gerenciador de GeoVisões;

[4]Referente à coluna TIPO DE OPERAÇÃO (tabela 3.12).

```
<View_Extension_Definition> ::=
        <View_Header_Extension_Definition>
        <View_Classes_Extension_Definition>
        [OPERATIONS <Operations>]

<View_Header_Extension_Definition> ::=
        DEFINE VIEW <View_Extension_Definition_Name>
        ON VIEW SCHEMA <View_Schema_Name>
        BASE EXTENSION <Extension_Name>
        [EXTENSION <View_Extension_Name>]
        TARGET AREA = {<Query_Command>|<Coordinates>|<expression>}
        ON SCALE = <scale>

<View_Classes_Extension_Definition> ::=
        VIEW CLASS <class_name> [NAME <persistence_name>]
        [INCLUDES <Query_Command>]
        [<View_Attribute_Population>]
        [OPERATION <Operation_Specification>]
        [<View_Class_Extension_Definition>]

<View_Attribute_Population> ::=
        {ATTRIBUTE <attribute_name> = {<expression>|<view_method>}}
        [<View_Attribute_Population>]

<Query_Command> ::=
        BEGIN QUERY [SCALING]
        <native_query_command>
        END QUERY

<Operation_Specification> ::=
        {[<GeOperations>]
        [<View_Operations>]
        [<Operation_Specification>]}

<View_Operations> ::=
        VIEW OPERATION <function_invocation>
```

Figura 4.12: GeoVDL- Sintaxe de definição de comando de povoamento

- **esquema ($\mathcal{E}'$)**: a associação do comando de povoamento $\mathcal{P}$ com o esquema $\mathcal{E}'$ é feita através da cláusula **ON VIEW SCHEMA <View_Schema_Name>** . Cabe ao gerenciador de GeoVisões garantir a integridade entre as definições de $\mathcal{P}$ e $\mathcal{E}'$;

- **extensão base (χ)**: a referência ao repositório que contém as raízes de persistência das classes base é incluída na definição do comando de povoamento através da cláusula **BASE EXTENSION <Extension_Name>** ;

- **extensão da GeoVisão (χ')**: a cláusula **[EXTENSION <View_Extension_Name>]** determina o nome do repositório de dados que armazenará os dados da GeoVisão. Cabe ao gerenciador de GeoVisões garantir a integridade entre as definições da GeoVisão e os repositórios, começando pela garantia de unicidade de nome do repositório;

- **área-alvo (ω)**: o povoamento da ω de uma GeoVisão é definido através da cláusula **TARGET AREA = {<Query_Command> | <Coordinates> | <expression>}** , no cabeçalho do comando de povoamento. A definição da área-alvo através de *buffers* é obtida através de expressões. A definição da área alvo em linguagens de consulta geográficas é, normalmente, incluída nos comandos de consulta, como em GML (através da cláusula

WITHIN) e em LEGAL [Cam95] (através da cláusula ON MAP). No caso específico de LEGAL, a inclusão da cláusula ON MAP está diretamente ligada à estrutura de armazenamento de dados no sistema SPRING [CCH+96], basicamente imagens. Em GeoVDL, a área-alvo é definida como uma restrição global à composição do cenário, evitando que os comandos de consulta às classes base precisem incluí-la nos predicados.

- **escala** (ρ): assim como ω, a definição da escala no cabeçalho do comando de povoamento evita sua inclusão em todos os predicados dos comandos de consulta. A sua definição é feita através da cláusula ON SCALE = <scale> . A escala não deve constituir uma restrição à recuperação de dados para a construção do cenário, pois o SGBDOO subjacente pode não conter dados na escala desejada pela GeoVisão que, por sua vez, pode dispor de funções de conversão de escala. Contudo, a utilização da escala como restrição aos comandos de consulta pode ser forçada através da cláusula [SCALING] anexada aos comandos de consulta;

- **comando de consulta** : os comandos de consulta são executados na sequência em que foram definidos em $\mathcal{P}$. Isto garante que ω, quando povoada através de um comando de consulta, estará disponível para a execução dos demais comandos de consulta de $\mathcal{P}$. Comandos de consulta são expressos na linguagem de consulta do SGBDOO subjacente, dentro do contexto determinado por BEGIN QUERY [SCALING]–END QUERY . Aqui, novamente, consideramos que a linguagem de consultas permite a definição de predicados espaciais;

- **definição de funções**: funções são especificadas junto com a definição do esquema, através da cláusula [OPERATIONS <Operation_Specification>] e [OPERATION <Operation_Specification>] . A aplicação de funções não espaciais é suportada pela linguagem através da cláusula VIEW FUNCTION .

- **a forma de povoamento de:**

  - **classes**: o povoamento das classes de visão é definido através da cláusula <View_Classes_Extension_Definition> , que associa uma raiz de persistência, um comando de consulta e um conjunto de operações à classe de visão. O resultado da avaliação do comando de consulta pode ser submetido às funções e é utilizado para criar a extensão da classe de visão. As classes de uma GeoVisão apresentam particularidades no seu povoamento:

    * **classes de controle** : a área-alvo ω é a primeira classe a ser povoada. O resultado da avaliação do comando de consulta das classes de contexto γ é restrito pela área-alvo ω e, possivelmente, pela escala. Dados de contexto podem ser submetidos a funções espaciais e não espaciais. A definição de

atributos atualizáveis em classes de contexto constitui um erro semântico, que deve ser detectado pelo compilador GeoVDL;

  * **geo-classes:** os dados são restritos espacialmente pela ω e, possivelmente, pela escala e podem ser submetidos a funções espaciais e não espaciais;
  * **classes convencionais :** não sofrem restrições espaciais e podem ser submetidos a funções convencionais;

- **atributos projetados:** o valor do atributo base é projetado para o atributo da classe de visão;
- **atributos virtuais :** a expressão/método que valora o atributo virtual é avaliada/executado e o resultado é atribuído ao atributo virtual;

4.4.3 Aspectos funcionais da GeoVDL

A linguagem GeoVDL deve permitir a representação sintática e semântica do modelo de GeoVisões, o qual é, por sua vez, suportado pelas funcionalidades de visões identificadas na seção 4.4.1.

Funcionalidades	Aparato Sintático (Semântico) da GeoVDL
Re-estruturação	[PROJECT <Attributes_Projection>] [PROJECT <Methods_Projection>] INCLUDES <Query_Command> GENERALIZES <Class_name_list>
Ger. de Informações	[<View_Attributes>] {{PUBLIC|PRIVATE} <attribute_name>:{<View_Method_Name>|<type_definition>}
Reutilização	IMPORT FROM SCHEMA <schema_name> CLASS <Base_Class_Name_list> ON VIEW SCHEMA <View_Schema_Name> BASE EXTENSION <Extension_Name> INHERITS <Class_name_list>
Integração	IMPORT FROM SCHEMA <schema_name> CLASS <Base_Class_Name_list>
População Virtual	[<View_Attributes>] [EXTENSION <View_Extension_Name>] <View_Classes_Extension_Definition> <View_Attribute_Population>
Materialização	Não é suportada na linguagem, mas através de funções da interface do gerenciador de GeoVisões
Atualização	[UPDATABLE BY <Method_Name>]
Funções	<Aspatial_Operations> <GeOperations>

Tabela 4.2: Funcionalidades de visões na GeoVDL

A tabela 4.2 identifica os recursos sintáticos e semânticos oferecidos pela GeoVDL para oferecer as funcionalidades tradicionais de visões:

- Re-estruturação de Dados
  - Projeção: através da projeção de atributos e métodos;

  - Renomeação: de atributos (cláusula <View_attribute_name>) e de métodos (cláusula <View_method_name>) importados;
  - Particionamento : pela avaliação dos comandos de consulta do comando de povoamento;
  - Geração de Informação : através da criação de atributos e métodos na visão, permitidos pelos não-terminais <View_Attributes> e <View_Methods> , este último acompanhado da seção de definição dos corpos dos métodos de visão, através da cláusula <View_Method_Body_Specification> ;

- Integração de Bases : através da importação de esquemas e da reutilização de esquema e dados de GeoVisões;
- Reutilização
  - de esquema: é permitida devido ao fato da definição do esquema estar separada da definição do comando de povoamento e pelo fato de definições de esquema poderem ser importadas para auxiliar a definição de GeoVisões. Como não existe restrição em relação ao fato do esquema base ser o esquema de uma visão ou de uma base real, esquemas de outras GeoVisões podem ser também utilizados. Outro tipo de reutilização permitida pela GeoVDL proposta é a utilização do esquema da GeoVisão ligado a diferentes comandos de povoamento;
  - de dados: GeoVisões podem ser utilizadas como base para a ativação de outras GeoVisões, devido à possibilidade de vários comandos de povoamento estarem vinculados a um mesmo esquema de GeoVisão;
- Atualização de visões: realizada por meio de *atributos atualizáveis* e suportada pelo mecanismo de propagação implementado pelo gerenciador de GeoVisões;
- População real e virtual: dependendo da ativação ser feita sobre uma base real ou uma GeoVisão materializada;

4.5 Suporte a GeoVisões

Esta seção analisa como o mecanismo de GeoVisões proposto e a GeoVDL suportam o modelo de GeoVisões, considerando as necessidades de SIG (seção 3.2.2) que o modelo busca atender.

4.5.1 Abstrações de geoprocessamento

As abstrações de geoprocessamento são obtidas através das definições de geo-classes. Geo-classes têm sua definição suportada pela re-estruturação de classes base, pela definição de

atributos e métodos virtuais e pela aplicação de funções de geoprocessamento. O suporte à re-estruturação de classes é resultado da combinação das definições de esquema $\mathcal{E}'$ e do comando de povoamento $\mathcal{P}$ de uma GeoVisão, como mostra a tabela 4.2.

Uma geo-classe é definida sobre um conjunto de classes base. Sua extensão é obtida através da avaliação de comandos de consulta. A sintaxe de definição do esquema $\mathcal{E}'$ da GeoVisão fornece ao gerenciador de GeoVisões informações necessárias para criação da extensão da geo-classe (por exemplo, **GENERALIZES**). À extensão de uma geo-classe podem ser aplicadas funções espaciais ou convencionais. Um exemplo do uso de tal funcionalidade é a criação de um campo temático (abstração de campo geográfico) como resultado da sobreposição de outros campos temáticos. Tal sobreposição é realizada aplicando um comando de povoamento que invoque uma operação de sobreposição (<overlay_operation>) sobre os dados das classes base.

4.5.2 Múltiplas Representações

A aplicação de funções de geoprocessamento permite que funções de mapeamento entre representações de abstrações geográficas sejam aplicadas às classes base, possibilitando as criação de *representações alternativas* para as abstrações geográficas. As GeoVisões criadas para derivar múltiplas representações não garantem a manutenção de consistência entre as representações, pois tal garantia depende da existência de funções inversas para as funções de mapeamento utilizadas na derivação da representação.

As representações criadas por GeoVisões originalmente utilizam dados virtuais, os quais podem ser materializados. As representações materializadas são armazenadas em repositórios específicos, isto é, não são integradas ao banco de dados original. O mesmo processo de criação de representações materializadas para abstrações geográficas é utilizado para a criação de *versões*. Logo, versões apresentam o mesmo problema de manutenção de integridade encontrado em múltiplas representações usando visões.

4.5.3 Construção de cenários

GeoVisões diferem de visões tradicionais pois buscam modelar uma abstração de trabalho específica de geoprocessamento: os cenários. Um cenário é formado por um conjunto de planos de informação definidos sobre uma área-alvo, de acordo com uma escala geográfica (seção 3.2.2). Além disso, os dados do cenário devem ser analisados sobre um contexto.

A modelagem de um cenário pode ser entendida como:

- definição da área-alvo (ω): implicitamente definida no esquema de GeoVisões, a área-alvo é povoada no cabeçalho do comando de povoamento. O suporte à restrição definida pela área-alvo é obtido por sua introdução nos predicados dos comandos de consulta às classes base;

- escala: a definição de escala é utilizada para parametrizar os comandos de consulta às classes base;

- contexto: as classes de contexto podem ser re-modeladas e re-estruturadas tal qual classes geográficas, sem serem, contudo, passíveis de atualização. As informações de contexto podem ser utilizadas para análise visual do cenário, através de interfaces e mecanismos de visualização;

- geo-classes : a construção das abstrações de geoprocessamento é fundamental para a composição de um cenário, pois modelam os seus planos de informação componentes, no modelo de campos ou de objetos geográficos. Os planos de informação da GeoVisão podem ser re-estruturados e combinados por meio da aplicação de funções de geoprocessamento. As funções definidas em nível de cenário são aplicadas aos dados das abstrações de geoprocessamento;

- informações do cenário: além de criar as abstrações de geoprocessamento, a GeoVDL, através dos comandos de consulta, permite a seleção do conjunto de dados relevante para o cenário, respeitando sua área-alvo e sua escala, quando for o caso. Além disso, permite a criação de dados virtuais, não presentes no SGBDOO subjacente;

- funções: a execução de funções espaciais e não-espaciais na definição do comando de povoamento permite a construção de cenários sofisticados. Em um SIG tradicional, cenários são criados como parte de programas de aplicação, pois a recuperação de dados é baseada somente na avaliação de comandos de consultas e as linguagens de consulta, mesmo espaciais (como por exemplo [Ege95]), não suportam visões. Dessa forma, a utilização de GeoVisões transfere a composição do cenário diretamente para a definição da GeoVisão, permitindo que as aplicações se concentrem na manipulação dos dados do cenário. Além disso, estas funções podem ser posteriormente aplicadas aos dados da visão.

Capítulo 5

Estudo de Caso e Aplicação do Mecanismo

Este capítulo mostra a utilização do mecanismo proposto de GeoVisões através da modelagem, em GeoVDL, de uma aplicação real desenvolvida por pesquisadores da Faculdade de Engenharia Civil da Unicamp, sem o auxílio de SIG ou outras ferramentas computacionais, em especial SGBD. Por isso, utilizamos uma modelagem fictícia para o BDOO geográfico subjacente, a qual se baseou na modelagem de [OPM97], que utilizou a mesma modelagem como estudo de caso.

5.1 Descrição da Aplicação

A aplicação, realizada por pesquisadores da Faculdade de Engenharia Civil da Unicamp [dA95b], estuda a utilização do planejamento ambiental como instrumento à prevenção de doenças infecto-contagiosas e parasitárias em uma região. Neste estudo, especificamente, a região escolhida foi o município de Paulínia, no estado de São Paulo.

O objetivo da aplicação é avaliar as relações diretas e indiretas existentes entre as estruturas sócio-econômica e fisico-biológicas e a saúde pública de uma região. O estudo utilizou a incidência de doenças infecto-contagiosas e parasitárias como indicadores de saúde e elementos físico-biológicos como indicadores de meio ambiente.

5.1.1 Metodologia de desenvolvimento da aplicação

Do ponto de vista de análise ambiental, o processo de desenvolvimento do estudo é dividido em 4 etapas:

- *reconhecimento epidemiológico*, onde os indicadores de saúde são analisados e distribuídos de forma absoluta ou relativa em relação a um conjunto de características ou evidências;

- *identificação dos fatores ambientais* que conduzem o quadro de saúde. Estes fatores são ponderados de acordo com o potencial transmissor das doenças em estudo;

- *identificação das unidades de paisagem*, isto é, aplicação de técnicas de priorização, ponderação e classificação dos fatores bióticos e antrópicos da área de estudo, gerando *mapas de risco* ou de *criticidade*, a partir dos quais são identificadas regiões homogêneas. Neste estudo de caso, as unidades de paisagem identificam áreas pontencialmente suscetíveis à ocorrência de epidemias de doenças infecto-contagiosas e parasitárias.;

- *formulação de diretrizes*, que visam estabelecer diretrizes para ações corretoras e preventivas para melhorar as condições do tema de estudo na região (neste caso, a saúde pública em Paulínia).

Na etapa de reconhecimento epidemiológico ocorre a seleção dos agentes propagadores de moléstias. Neste estudo, a seleção dos agentes foi realizada a partir da aplicação de questionários ao médicos responsáveis pelos setores de epidemiologia dos centros de saúde do município e do tabelamento da frequência de ocorrência de doenças infecto-contagiosas e parasitárias no município.

A identificação dos indicadores de saúde ligados ao meio ambiente corresponde ao levantamento das características ambientais, populacionais e de ações antrópicas relacionadas a doenças infecto-contagiosas e parasitárias. A tabela 5.1 mostra os indicadores sócio-econômicos e físico-biológicos utilizados no estudo.

Na fase de *identificação das unidades de paisagem* são estabelecidas as correlações entre os indicadores de saúde pública, sócio-econômicos e físico-biológicos. Esta etapa foi realizada em 5 fases distintas:

1. *Criticidade*: cada indicador, sócio-econômico ou físico-biológico, teve suas categorias classificadas de acordo com a potencial contribuição para a transmissão de doenças infecto-contagiosas e parasitárias. A ponderação, chamada de *valor de criticidade*, é um valor inteiro que oscila entre 1 e 5 e que foi atribuído em ordem crescente às categorias do indicador. Assim, às categorias que mais contribuem para a transmissão de doenças infecto-contagiosas e parasitárias foi atribuído o valor de criticidade 5 (seção 5.2);

2. *Reclassificação*: para cada indicador foi gerado um mapa de criticidade, isto é, um mapa temático reclassificado de acordo com os níveis de criticidade de suas categorias;

3. *Ponderação*: os indicadores foram ponderados entre si, para refletir a influência que cada um tem na transmissão de doenças infecto-contagiosas e parasitárias, como

Tipos	Componentes	Indicadores
Sócio-econômico	Estrutura Urbana	Adensamento populacional
		Crescimento populacional
		Concentração de migrantes (bairros)
	Industriais	Zonas de maior concentração de empregos
		Zonas de maior concentração de população
		Zonas de maior concentração de poluentes
	Econômicos	Renda per capita (bairro)
		Densidade populacional
		Tamanho de lote
	Infra-estrutura Social	Hospitais
		Centros de Saúde
		Problemas de Saúde
		Relação médico/habitante
		Taxa de natalidade
		Taxa de mortalidade
		Principais causas de óbitos
	Infra-estrutura de Educação	Número de escolas
		Tipo de escolas
		Taxa de alfabetização
		População em idade escolar
	Saneamento Básico	Sistema de tratamento e abastecimento de água
		Redes coletoras de esgoto e tratamento posterior
		Limpeza pública
Físico-biológicos	Clima	Temperatura
		Precipitação
		Direção do vento
	Relevo	Tipo
		Forma
	Solos	—
	Água superficial	Rede hidrográfica
		Qualidade da água
	Cobertura vegetal	Tipos
		Estados de preservação
	Uso da Terra	Áreas de pasteio
		Agricultura
		Irrigação
		Represamentos
		Áreas industriais
	Fauna	Animais domésticos
		Animais Silvestres

Tabela 5.1: Indicadores sócio-econômicos e físico-biológicos

Tipo	Indicador	Peso
Sócio-econômico	Migração	14,00
	Irrigação e agricultura	8,30
	Represamento	8,50
	Agricultura não irrigada	1,50
	Indústrias	5,20
	Infra-estrutura de saúde e educação	8,30
	Crescimento populacional	8,70
	Rede de abastecimento (água)	9,00
	Rede de coleta (esgoto)	8,90
	Pavimentação	2,00
Físico-Biológicos	Recursos hídricos	7,50
	Solo	4,70
	Relevo	2,50
	Vento	2,00
	Cobertura Vegetal	3,50
	Fauna	6,00

Tabela 5.2: Ponderação dos indicadores sócio-econômicos e físico-biológicos

mostra a tabela 5.2. A ponderação foi realizada de forma que a soma dos pesos dos indicadores totalizasse 100;

4. *Produção de mapas intermediários*: os mapas de criticidade de cada indicador foram superpostos (por soma), produzindo mapas de criticidade intermediários que indicam as áreas homogêneas (unidades de paisagem). Nesta etapa, a superposição ocorreu somente entre indicadores do mesmo tipo. Os mapas de criticidade gerados foram:

   - Áreas de criticidade, de acordo com fatores sócio-econômicos;
   - Áreas de criticidade, de acordo com fatores físico-biológicos

5. *Geração de unidades de paisagem*: os pares de mapas intermediários são reclassificados segundo 5 valores de criticidade e, então, superpostos, gerando um mapa de criticidade final que indica as áreas pontencialmente críticas à ocorrência de doenças infecto-contagiosas e parasitárias.

5.1.2 A aplicação sob a ótica de GeoVisões

Esta aplicação pode ser encarada como uma combinação de sucessivas operações de superposição de campos temáticos de criticidade, resultantes das reclassificações dos indicadores sócio-econômicos e ambientais da região, de acordo com os valores de criticidade atribuídos a cada uma de suas categorias.

A utilização do mecanismo de GeoVisões ocorre a partir da fase de *identificação das unidades de paisagem*. Para GeoVisões, unidades de paisagem são as categorias (unidades homogêneas) nas quais está classificado um campo temático. Nesta fase são criados os

mapas de criticidade que, em GeoVisões, correspondem à reclassificação da região de estudo segundo a ponderação das categorias dos indicadores.

5.2 Descrição do Cenário

Um cenário, em GeoVisão, é o resultado de combinar um conjunto de planos de informação (PI) definidos sobre uma área-alvo de estudo (seção 3.2.2). Esta seção descreve como o uso de GeoVisões pode auxiliar na definição dos planos de informação sócio-econômico e físico-biológico utilizados na composição do cenário de regiões críticas à saúde pública de Paulínia. A descrição dos PIs identifica os mapas de criticidade de cada indicador e os valores de criticidade atribuídos às suas categorias e a sequência de operações realizadas para sua criação.

A identificação de planos de informação no contexto do termo usado na dissertação é, de certa forma, subjetiva. Neste estudo de caso, optamos por definir 2 planos de informação: o plano de informação sócio-econômico (PISE) e o plano de informação físico-biológico (PIFB) que, superpostos, criam o cenário.

5.2.1 O plano de informação sócio-econômico

O PISE é resultado da análise dos indicadores sócio-econômicos, identificados na seção 5.1, sendo obtido a partir da superposição de 7 mapas temáticos de criticidade, que descrevem a área alvo da aplicação de acordo com os respectivos indicadores. Os valores de criticidade utilizados para gerar mapas são apresentados na tabela 5.3.

Indicador	Categorias	Criticidade
Densidade Populacional	> 70 hab/ha	5
	20-70 hab/ha	3
	< 20 hab/ha	1
Centros de Saúde	Centro de Saúde Escola	5
	Monte Alegre	4
	PS João Aranha	3
	PS Jardim Planalto	2
Rede abastecedora de água	Ausente	5
	Presente	1
Rede coletora de esgoto	Ausente	5
	Presente	1
Pavimentação	Ausente	5
	Presente	1
Uso da Terra	Industrial	5
	Ocupação Dispersa	4
	Campo Antrópico	3
	Solo Exposto	2
	Monoculturas	1

Tabela 5.3: Mapas de criticidade do PISE

O mapa de criticidade de *densidade de povoamento* é obtido a partir de uma operação de reclassificação dos níveis de adensamento populacional. O mapa de criticidade de *infra-estrutura de saúde* resulta da reclassificação das áreas de abrangência dos centros de saúde, por bairros. A cada centro de saúde foi atribuído um nível de criticidade, embora o estudo não apresente os critérios adotados. Para a confecção do mapa de criticidade de *infra-estrutura de saúde e educação* foram correlacionados dados de saúde com os dados de educação contidos na tabela 5.1. Mais uma vez, a forma de correlação não está descrita no trabalho. Os mapas temáticos de criticidade referentes à *rede abastecedora, rede coletora* e *pavimentação* são resultantes da sobreposição entre as referidas redes e o mapeamento urbano do município. O mapa de criticidade de *uso da terra* foi resultado da reclassificação das categorias de uso identificadas a partir da análise de uma imagem de satélite.

A geração do PISE

A geração do mapa temático de criticidade que modela o PISE é realizado em 3 etapas:

1. ponderação de cada mapa de criticidade pelo peso do indicador, apresentado na tabela 5.2;
2. 6 superposições numéricas de adição;
3. reclassificação do temático resultante em 5 níveis de criticidade. Os critérios de reclassificação do temático resultante não foram apresentados no estudo;

5.2.2 O plano de informação físico-biológico

O PIFB é resultado da análise dos indicadores físico-biológicos, identificados na seção 5.2, resultando da superposição de 4 mapas temáticos de criticidade, que descrevem a área alvo da aplicação de acordo com os respectivos indicadores. Os mapas temáticos de criticidade são apresentados na tabela 5.4.

Os critérios de reclassificação do mapa de criticidade de *solos* não foram apresentados no trabalho. O mapa de criticidade de *recursos hídricos* foi utilizado na determinação dos níveis de criticidade. Para a confecção do mapa de criticidade de *fauna*, o mapa temático de uso do solo foi reclassificado, associando a ocorrência de animais domésticos à zona urbana, de animais silvestres às zonas de matas e de animais de criação às zonas de pasteio. O mapa de criticidade de *direção dos ventos* dividiu a região do município em 5 grandes áreas a partir da localização da Refinaria do Planalto (REPLAN).

A geração do PIFB

A geração do mapa temático de criticidade que modela o PIFB é realizado em 3 etapas:

Indicador	Categorias	Criticidade
Relevo	Planície Pluvial Pedimento	5
	Colina ampla e média Colina pequena Colina média	3
	Colina ampla horizontal Colina ampla	1
Fauna	Domésticos	5
	Silvestres	3
	Criação	1
Ventos	Sudeste	5
	REPLAN	4
	Nor-nordeste	3
	Noroeste	2
	Sudoeste	1

Tabela 5.4: Mapas de criticidade do PIFB

1. ponderação de cada mapa de criticidade pelo peso do indicador, apresentado na tabela 5.2;
2. 3 superposições numéricas de adição;
3. reclassificação do temático resultante em 5 níveis de criticidade. Os critérios de reclassificação do temático resultante não foram apresentados no estudo;

5.2.3 A criação do cenário

Os planos de informação PISE e PIFB foram definidos de forma análoga e possuem algumas características em comum, como:

- a área-alvo de estudo foi a região do município de Paulínia, no estado de São Paulo;
- os mapas temáticos de criticidade foram criados a partir de um mapa da região de Paulínia, em escala geográfica de 1 : 50000, com execeção do mapa de solos que estava na escala de 1 : 100000 e teve der ser convertido;
- o contexto de desenvolvimento do estudo foi formado pelo mapa de divisão administrativa do município de Paulínia, incluindo ainda: hidrografia, rodovias, ferrovias e a indicação dos centros de saúde pública e escolas;
- os mapas de criticidade foram obtidos, em sua maioria, a partir da reclassificação do mapa de divisão administrativa de acordo com os níveis de criticidade de cada indicador. Os mapas de criticidade intermediários (PIs) resultaram da sobreposição dos mapas de criticidade de cada indicador;

O cenário, desta forma, foi obtido a partir das operações sucessivas de reclassificação e superposição, sendo formado pelos planos de informação PISE e PIFB.

5.3 Modelagem em GeoVDL

As seções anteriores descreveram o estudo de caso de acordo com seus requisitos de informações (mapas de criticidade) e de operações (reclassificação, superposição). Do ponto de vista do modelo de GeoVisões, a modelagem da aplicação segue as mesmas etapas do trabalho realizado, isto é, modelar os planos de informação e aplicar as funções de geoprocessamento necessárias. Todavia, a modelagem de dados em GeoVDL depende diretamente da modelagem do BDOO geográfico subjacente. Ressalte-se, igualmente, que o trabalho original foi manual, sem seguir nenhum modelo ou metodologia reconhecidas em computação.

5.3.1 O BDOO geográfico subjacente

De acordo com a definição apresentada no capítulo 4, um BD subjacente ao mecanismo de GeoVisões é um BDOO geográfico. A figura 5.1 apresenta parte de uma modelagem hipotética do esquema do BDOO geográfico, descrito em ODL, a linguagem de definição de objetos do SGBDOO O_2. O esquema deste BDOO será chamado de BDSchema, a extensão será chamada de BD e cada classe será associada a uma raiz de persistência (comentário). O conjunto de funções de geoprocessamento associado ao BDSchema envolve tipicamente as funções descritas na seção 3.1.4.

A figura 5.1 mostra parte das classes que compõem o esquema do BD (BDSchema). As outras classes identificadas na análise da aplicação do estudo de caso foram **PropriedadeRural, Rede, RedeMunicipal, AbastecimentoAgua, ColetaEsgoto, Rodovia, Ferrovia, CentroSaude, Escola, Hidrografia, MapeamentoUrbano, Categoria, Tematico, Relevo, Vento, Solo, UsodaTerra, Ponto, Linha,** e **Poligono** , que não serão descritas aqui. Contudo, vale ressaltar que as classes **AbastecimentoAgua** e **ColetaEsgoto** são especializações das classe **RedeMunicipal** ; as classes **Rodovia** e **Ferrovia** são especializações da classe **Rede** ; e as classes **Relevo, Vento, Solo** e **UsodaTerra** são especializações da classe **Tematico** .

5.3.2 A descrição do cenário em GeoVDL

Esta seção apresenta o cenário de *áreas críticas à saúde* do município de Paulínia, especificado como uma GeoVisão. O cenário será descrito na seguinte ordem:

- Esquema: cabeçalho e classes;
- Funções

```
class Municipio      /* Municipios */
public type tuple(   nome              : string;
                     zonaurbana        : set (Bairro);
                     zonarural         : set (PropriedadeRural);
                     abastagua         : AbastecimentoAgua;
                     coletaesgoto      : ColetaEsgoto;
                     rodovia           : Rodovia;
                     ferrovia          : Ferrovia;
                     saude             : set (CentroSaude);
                     educacao          : set (Escola);
                     hidrografia       : set (Hidrografia);
                     mub               : MapeamentoUrbano;
                     geodesico         : Ponto;
                     geometria         : Poligono)
end;

class Bairro         /* Bairros */
public type tuple(   nome              : string;
                     populacão         : integer;
                     municipio         : Municipio;
                     centrosaude       : list(CentroSaude);
                     escolas           : list(Escola);
                     geometria         : Poligono)
end;
```

Figura 5.1: Parte do esquema do BDOO geográfico

- Comando de Povoamento

O esquema

O cenário é composto pelos PIs PISE e PIFB. Esta seção descreve a modelagem do esquema do cenário através de um exemplo de modelagem de campos temáticos, incluindo suas classes, métodos e funções. Como ambos os PIs são utilizados no mesmo cenário, o cabeçalho da definição do esquema é o mesmo para ambos. A figura 5.2 mostra a especificação de um campo temático do PISE.

As classe contidas na lista de importação poderiam ser omitidas, com execeção das classes **Municipio, Solo, Relevo, Vento** e **UsodaTerra** . A rigor, de acordo com a descrição do mecanismo de importação, na seção 4.4.2, as demais classes também seriam importadas para o esquema **SaudePublica** pois são componentes destas classes.

Neste exemplo, quatro classes são modeladas na GeoVisão: uma classe de geo-objeto **BairroDP** , uma classe de geo-campo **(DensidadePopulacionalBairro)** , uma classe convencional **(TabelaReclassificacao)** e uma classe de contexto **Contexto** . A classe **BairroDP** é uma classe

```
DEFINE      VIEW SCHEMA SaudePublica
IMPORT      FROM SCHEMA BDSchema CLASS Municipio, Bairro, PropriedadeRural
            Rede, RedeMunicipal, AbastecimentoAgua, ColetaEsgoto,
            Rodovia, Ferrovia, CentroSaude, Escola, Hidrografia,
            MapeamentoUrbano, Categoria, Tematico, Relevo, Vento,
            Solo, UsodaTerra, Ponto, Linha, Poligono

CLASS       Contexto
            ISA CONTEXT
            INHERITS Municipio
            TYPE TUPLE (
            zonaurbana: BDSchema.Municipio.zonaurbana;
            zonarural: BDSchema.Municipio.zonarural;
            rodovia: BDSchema.Municipio.rodovia;
            ferrovia: BDSchema.Municipio.ferrovia;
            saude: BDSchema.Municipio.saude;
            educacao: BDSchema.Municipio.escola;
            hidrografia: BDSchema.Municipio.hidrografia;
            geodesico: BDSchema.Municipio.geodesico;
            geometria: BDSchema.Municipio.geometria)
            END;

CLASS       BairroDP
            ISA OBJECT
            PUBLIC TYPE TUPLE (
            PUBLIC nome: BDSchema.Bairro.nome;
            PUBLIC categoria : TUPLE (
            PUBLIC geometria: BDSchema.Bairro.geometria;
            PUBLIC VIEW densidade: CalculoDensidade))
            METHODS
            PRIVATE area: BDSchema.Poligono.area;
            PUBLIC VIEW CalculoDensidade(): integer
            END;

CLASS       TabelaReclassificacao
            ISA ASPATIAL
            PRIVATE TYPE SET(
            limiteinferior: integer;
            limitesuperior: integer;
            novovalor: integer)
            METHODS
            PUBLIC VIEW InsereLinha(integer,integer,integer): boolean;
            PRIVATE VIEW SetUpLimiteInferior(integer): boolean;
            PRIVATE VIEW SetUpLimiteSuperior(integer): boolean;
            PRIVATE VIEW SetUpNovoValor(integer): boolean;
            END;

CLASS       DensidadePopulacionalBairro
            ISA THEMATIC FIELD
            END;

OPERATIONS  RECLASSIFY
            VIEW FUNCTION GeraTabelaDP(TabelaReclassificacao) : boolean;
            VIEW FUNCTION BairroDP2Tematico (BairroDP,ThematicField) : boolean
            VIEW FUNCTION PonderaTematico (ThematicField,real) : boolean;
```

Figura 5.2: Esquema com classes do PISE

de visão que possui atributos diretamente retirados do BD (nome, geometria) e um atributo virtual (densidade) cujo valor está associado à execução do método CalculoDensidade , também virtual. Esta classe de visão permite gerar a informação derivada *densidade populacional*, não presente em BDSchema . A classe TabelaReclassificação corresponde à tabela utilizada para realizar reclassificação. A título de ilustração, os atributos da tabela foram definidos como PRIVATE , assim como os métodos de visão, responsáveis pela atualização dos seus valores. Como o único método PUBLIC é o método InsereLinha, esta modelagem sugere que os valores da tabela de mapeamento para reclassificação não estão sujeitos a efeitos colaterais na modelagem das aplicações sobre a GeoVisão. A classe DensidadePopulacional modela um geo-campo temático, que será povoado a partir da aplicação de funções que combinam as duas outras classes do esquema SaudePublica . finalmente, a classe Contexto é modela o contexto de execução da aplicação.

As modelagens de BDSchema e SaudePublica sugerem a utilização de duas funções: uma para criar DensidadePopulacionalBairro a partir de BairroDP (função de conversão objeto-campo) e outra que atribua às categorias resultantes do campo temático os corretos valores de criticidade, como definido na tabela 5.3 (função de reclassificação). Além disso, os valores de criticidade devem ser ponderados de acordo com os valores apresentados na tabela 5.2 e é preciso criar a tabela com os valores para a reclassificação (função não-espacial). Assim, tais funções estão presentes na definição do PISE, respectivamente BairroDP2Tematico, Reclassify, PonderaTematico e GeraTabelaDP .

É necessário ressaltar que a operação de ponderação pode ser realizada de duas formas: (1) uma função que modifique os valores das categorias dos campo temático e (2) através de superposição numérica de multiplicação com um campo temático de categoria única cuja geometria corresponde à área alvo e cujo valor da categoria é igual ao peso do indicador. No entanto, esta última operação não satisfaz a definição da operação de Análise de Ponderação (seção 3.1.4).

O comando de povoamento

A definição do comando de povoamento para o esquema SaudePublica é apresentado na figura 5.3. Os comandos de consulta serão escritos em uma extensão OQL que simula a inclusão de operadores espaciais.

O comando de povoamento para esta aplicação pode ser definido em 3 partes: o cabeçalho, o povoamento das classes e a aplicação de funções. O cabeçalho do comando de povoamento da GeoVisão SaudePublica define o nome pelo qual o comando de povoamento (SaudePublicaPaulinia) poderá ser executado pelo gerenciador de GeoVisões. A seguir, indica que esquema (SaudePublica) descreverá a extensão do BD virtual (SPP) que será criado a partir do BD base (BD) . Além disso, o cabeçalho permite a definição da área alvo do cenário que, neste estudo de caso, corresponde ao municipio de Paulínia, sendo obtido a partir de uma consulta.

```
DEFINE   VIEW EXTENSION SaudePublicaPaulinia
         ON VIEW SCHEMA SaudePublica
         BASE EXTENSION BD
         EXTENSION SPP
         TARGET AREA = BEGIN QUERY
          select m.geometria
          from m in Municipios
          where m.nome = "Paulinia"
          END QUERY;
         ON SCALE = 1:50000

         VIEW CLASS Contexto NAME TheContexto
         INCLUDES BEGIN QUERY
          select m
          from m in Municipios
          where m.nome = "Paulinia"
          END QUERY;

         VIEW CLASS BairroDP NAME TheBairroDP
         INCLUDES BEGIN QUERY
          select b
          from b in Bairros
          where IN(b.geometria,TARGETAREA) and
          b.municipio = "Paulinia")
          END QUERY

         VIEW CLASS TabelaReclassificacao NAME TRBairroDP
         OPERATION VIEW FUNCTION GeraTabelaDP(TRBairroDP)

         VIEW CLASS DensidadePopulacionalBairro NAME DPBairro

         OPERATIONS VIEW FUNCTION BairroDP2Tematico(BairroDP,DPBairro)
          RECLASSIFY DPPBairro
          BY TRBairroDP
          ONTO dp in DPBairro
          VIEW FUNCTION PonderaTematico(DPBairro,peso)
```

Figura 5.3: Comando de povoamento para o esquema SaudePublica

A área alvo poderia ter sido determinada através da definição de um polígono fechado. Contudo, como a classe base **Municipio** já contém a geometria da região, a definição da área alvo como um comando de consulta sobre a classe **Municipio** permite que possíveis alterações na geometria (pela emancipação de um distrito, por exemplo) não afetem a semântica da GeoVisão. A definição da escala geográfica segue os padrões de sua definição em cartografia.

A especificação do povoamento das classes ocorre na ordem em que elas foram definidas no esquema, embora isto não seja uma restrição. A classe de contexto **Contexto** é, como todas as demais classes do esquema, associada a uma raiz de persistência **(TheContexto)** e o comando de consulta, neste caso, realiza apenas uma seleção convencional.

O comando de consulta que povoa a classe **BairroDP** , por sua vez, inclui um predicado espacial **(IN)** e utiliza a palavra reservada **TARGETAREA** , utilizada para fazer referência à geometria que define a área alvo. As classes **TabelaReclassificacao** e **DensidadePopulacional-**

Bairro não são povoadas por comandos de consulta, mas sim pela execução de funções. O povoamento da classe **TabelaReclassificacao** pode ser realizado de forma independente, considerando que os dados de criação da tabela fazem parte do código da função **GeraTabelaDP** . Por outro lado, o povoamento da classe **DensidadePopulacionalBairro** é feito em etapas:

1. a função **BairroDP2Tematico** é executada para criar a extensão que
2. é reclassificada **(RECLASSIFY)** com base na tabela de reclassificação, já criada. Finalmente,
3. o campo temático resultante da reclassificação é ponderado, através da função **PonderaTematico** ;

5.3.3 Modelagem adicional

A modelagem completa do problema, em GeoVDL, exigiria a definição de mais 21 classes adicionais para o PISE e ainda 12 classes adicionais para PIFB. Além disso, seria necessário definir classes para o mapeamento urbano do município. A definição de todas estas classes adicionais seria feita de forma análoga ao descrito anteriormente neste capítulo, não consistindo, desta forma, em contribuição adicional ao trabalho ou à sua compreensão sendo, por isso, omitida.

Ao final a aplicação exige a realização de uma operação de **OVERLAY** dos 2 planos de informação, PISE e PIFB. Esta operação pode ser especificada em GeoVDL como mostrado na figura 5.4.

```
OPERATIONS OVERLAY PISE,PIFB
  ONTO AreaCriticaSaudePublica
  ADDING
```

Figura 5.4: Superposição entre PIs

Capítulo 6

Conclusões

6.1 Conclusões

Esta dissertação propôs um modelo e um mecanismo de visões para sistemas de informações geográficas, validados através da modelagem de uma aplicação real.

O estudo dos mecanismos de visões foi motivado pela forte dependência que SIG têm em relação a SGDB. Na revisão foram estudados mecanismos de visões relacionais e orientados a objetos e indentificadas as principais funcionalidades obtidas em SGBD através do uso de visões e o inter-relacionamento entre tais funcionalidades. A revisão permitiu também identificar os principais problemas resultantes da utilização de mecanismos de visões, como o problema de manutenção da consistência entre os dados de uma visão e o BD a partir do qual ela foi criada, e os fatores envolvidos no projeto de um modelo de visões orientadas a objetos (capítulo 2).

O estudo de SIG permitiu a familiarização com os problemas e necessidades dessa área de pesquisa e a identificação de necessidades de SIG que podem ser atendidas através de um mecanismo de visões, principalmente a criação de cenários. Para atender a tais necessidades, foi realizado um estudo sobre os modelos de dados para SIG e foi identificado um conjunto de operações comumente utilizados em aplicações que envolvem geoprocessamento, sem a pretensão que tal conjunto seja mínimo e/ou completo. O principal resultado do estudo de SIG foi identificar como visões podem ser úteis em SIG e quais os requisitos que um mecanismo de visões para SIG deve atender e que o difereciam de um mecanismo de visões convencional (capítulo 3).

O modelo de visões proposto foi fortemente influenciado por [AB91, Ber91, dSAD93, dS95] principalmente quanto à definição de uma visão como um banco de dados e ao conceito de objetos virtuais. O modelo de visões proposto - GeoVisões - incorporou conceitos de geoprocessamento ligados à construção de cenários, como área alvo, contexto e escala geográfica. Da manipulação de cenários surge o requisito de suporte à manipulação e exploração de alternativas, que motivou o suporte à reutilização de esquemas, através de

sua importação, e de dados, através do aninhamento de visões. Para traduzir os conceitos do modelo de visões para o mecanismo que o implementa foi especificada uma linguagem, a GeoVDL (*GeoView Definition Language*). Um resultado interessante é a possibilidade de utilizar um esquema associado a diversos comandos de povoamento (capítulo 4).

As principais contribuições desta dissertação são:

- estudo detalhado do papel de visões em SIG;
- proposta de um modelo de visões orientado a objetos a ser usado em SIG, mostrando a necessidade de dados e informação semântica adicional em relação aos modelos convencionais
- apresentação de um mecanismo para implementar tal modelo
- especificação de uma linguagem para definição destas visões

Além disso, a dissertação realizou um estudo comparativo de diversas taxonomias de operações definindo, como resultado, uma taxonomia global. Finalmente, o modelo e a linguagem propostos foram validados através de um estudo de caso real.

6.2 Extensões

Existem diferentes tipos de extensões a esta dissertação, tanto práticas quanto teóricas. A extensão mais evidente é a implementação do mecanismo, o que acarreta problemas que vão desde a execução até a necessidade de especificar determinados algoritmos e operações para realizar funções como integração de esquema, evolução de esquema, manutenção de consistência entre visão e banco de dados, entre outros. Cada uma dessas funções corresponde a um problema a ser resolvido como extensão desta dissertação.

Estas e outras extensões desta dissertação serão apresentadas nas sub-seções seguintes, divididas em extensões ao modelo e extensões ao mecanismo.

6.2.1 Extensões ao modelo

Suporte temporal

O aspecto temporal é intrínseco a dados geográficos e pode ser suportado de duas maneiras: ou pelo SGBD subjacente ou pelo mecanismo de visões. SGBD com suporte temporal não existem comercialmente. Uma maneira de suportar a temporalidade de dados espaciais é acoplar um mecanismo de tempo como uma camada sobre o SGBD [MB96]. Outra maneira é incorporar ao mecanismo de visões funcionalidades de modelagem e gerência temporal de dados geográficos, o que, na verdade, significa acoplar a camada temporal ao mecanismo de visões ao invés de acoplá-la ao SGBD. Todavia, qualquer uma das soluções

influencia o modelo de visões para SIG, quer seja em nível de modelagem quer seja em nível de povoamento, ou em ambos.

Suporte a versões

Assim como o tempo, o suporte a versões não é plenamente oferecido por SGBD comerciais. O modelo de visões proposto nesta dissertação sugere a utilização de visões para criar versões, sem, contudo, propor a gerência dos dados versionados. Logo, o modelo de visões proposto não supõe a existência de versões no SGBD subjacente. Assim, a existência de um mecanismo de versões sob o mecanismo de visões pode influenciar o modelo e o mecanismo, espacialmente no que tange à linguagem de definição de visões e os processos de gerenciamento de visões (figura 4.7).

Suporte a consultas exploratórias

Consultas exploratórias pressupõem navegação no cenário modelado pela visão e criação de novos cenários a partir deste. A forma como o mecanismo de GeoVisões suporta consultas exploratórias está baseada na reutilização dos dados da GeoVisão quando da ativação de outra GeoVisão. Neste caso, a garantia de integridade entre os dados da visão ativa e o esquema da visão a ser ativada é de responsabilidade do programador de aplicações. Uma proposta para tentar diminuir tal responsabilidade é o estabelecimento de um relacionamento entre os esquemas e comandos de povoamento de GeoVisões, como o objetivo de monitorar dois fatos:

- a consistência entre o esquema da GeoVisão ativa e o da GeoVisão a ser ativada e
- a garantia que a extensão da GeoVisão ativa satisfaz aos predicados de povoamento da GeoVisão a ser ativada

Este relacionamento entre visões tem como característica principal a existência de um relacionamento de especialização entre esquemas de BDOO.

6.2.2 Extensões ao mecanismo

Integração de esquemas

O suporte à integração de esquemas aqui considerado não vislumbra soluções para heterogeneidade entre esquemas, mas está centrado no problema de importação de esquema para a definição de GeoVisões.

Em um ambiente que suporta a importação de mais de um esquema base, diferentes esquemas podem ser utilizados como base para uma GeoVisão. A associação das classes base ao seu esquema de origem elimina a possibilidade de conflitos entre nomes de classes

importadas de esquemas diferentes. Por outro lado, considere uma classe c utilizada na modelagem de diferentes GeoVisões as quais, por sua vez, estão sendo utilizadas como base para a modelagem de uma terceira GeoVisão. A mesma classe base c pode vir a ser duplicada na composição do esquema desta nova GeoVisão e, com ela, todas as classes envolvidas em sua composição. Este problema pode ocorrer mesmo que o mecanismo não suporte múltiplos esquemas base na definição da GeoVisão.

Os principais aspectos de estudo desta extensão são:

1. como detectar tais duplicações?
2. quais os efeitos que a duplicação causa no esquema de uma GeoVisão?
3. como evitar tais duplicações?
4. qual a relação custo/benefício entre evitar tais duplicações e dispender esforços para evitá-las?

Suporte à evolução de esquema

O problema de suporte à evolução de esquema está normalmente relacionado a garantir a consistência entre o esquema e a extensão descrita por ele [dSAD93]. Na presença de visões, a evolução de esquema implica também na garantia de consistência entre o esquema do BD e o esquema da visão. Um problema ainda maior é garantir tal consistência de maneira transparente. Os aspectos a serem estudados no problema de suporte à evolução do esquema são:

1. como garantir a consistência desta evolução?
2. existem níveis de inconsistência suportáveis?

Suporte à materialização

O estudo do suporte à materialização está diretamente relacionado com o problema de manutenção de integridade entre os dados do BD e da visão e com a criação de versões.

Definição do mecanismo de atualização de visões

O mecanismo de manutenção de consistência entre o BD e as GeoVisões está baseado no gerenciador de atualizações, que é responsável por detectar a ocorrência de atualizações, e o gerenciador de visões, que é responsável pela implementação da política de propagação de atualizações (seção 4.3). A definição completa de tal mecanismo ainda deve considerar:

1. quais as informações de controle necessárias ao gerenciador de visões?

2. quais as informações de controle necessárias ao gerenciador de atualizações?
3. quais são as formas de monitoramento de atualizações?
4. quais são os mecanismos de verificação que permitem decidir quando uma atualização deve ser propagada?
5. quais os mecanismos e políticas de notificação que o gerenciador de atualizações pode utilizar?
6. qual é o protocolo entre os gerenciadores de visão e de atualizações?
7. como implementar as políticas de propagação de atualizações?
8. qual o impacto da propagação de atualização em cascata sobre tais políticas e mecanismos?

Utilização de mecanismos ativos

Esta extensão contempla investigar como mecanismos ativos podem ser acoplados a mecanismos de visões, especialmente para:

- auxiliar o mecanismo de manutenção de consistência entre os dados da visão e do BD subjacente e
- auxiliar a manutenção da consistência das definições das visões na presença de evolução de esquema

Acoplamento de visões a SIG genérico

O mecanismo de visões proposto considera que o BD subjacente é OO e suporta o modelo de geo-classes de [CCH+96]. No entanto, isso nem sempre é verdade. Como extensão, propomos um dispositivo de customização (figura 6.1). Na figura, que é uma extensão da figura 4.6, o repositório de dados de customização contém o conjunto de dados necessários para permitir a criação de GeoVisões segundo o modelo proposto, sobre um BD de um SIG genérico.

A proposta de um mecanismo de visões para SIG genérico esbarra na heterogeneidade em diversos níveis, passando pela heterogeneidade de modelos de dados dos SGBDOO e pela modelagem geográfica e geométrica existentes em pacotes de suporte a geoprocessamento e aplicações espaciais. Assim, oferecer um modelo de visões a ser suportado por SGBDOO genérico implica necessariamente a integração de bancos de dados heterogêneos, caracterizados pelo modelo de GeoVisões e pelo modelo de dados subjacente. Embora visões sejam uma alternativa para fazer a integração de BD heterogêneos, a implantação

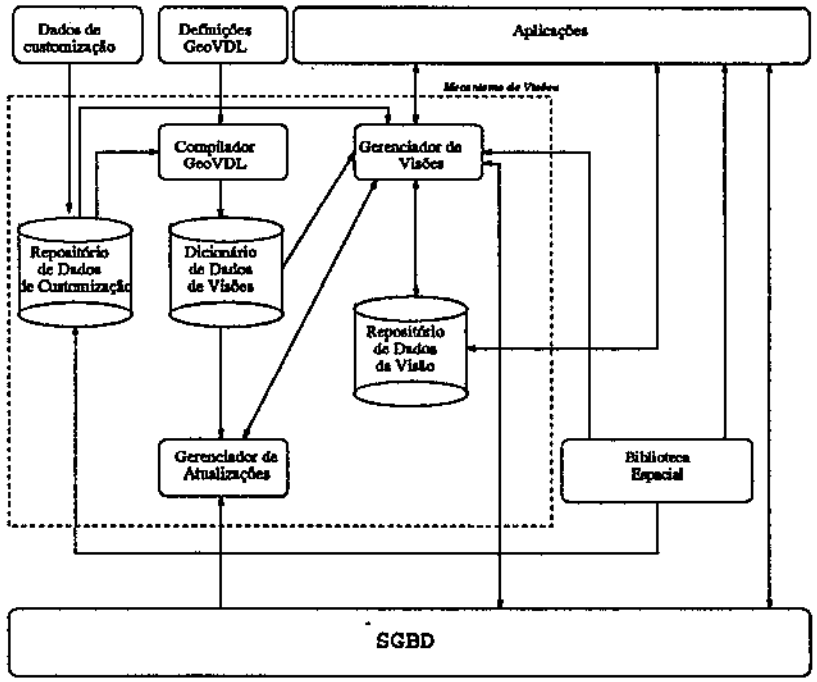


Figura 6.1: Arquitetura de um mecanismo de visões para SIG genérico

de um *repositório de dados de customização* está mais próxima da solução de integração por dicionário de dados global.

O repositório de dados de customização armazena os comandos de mapeamento entre as definições em GeoVDL e o modelo de dados do SGBDOO subjacente. Em outras palavras, como o mecanismo de visões proposto é um mecanismo externo a ser acoplado a um SGBDOO, o repositório de customização contém os dados necessários a tal acoplamento. Tais informações servem de parâmetro para que as definições em GeoVDLsejam corretamente mapeadas para o modelo de dados do SGBDOO e para que o gerenciador de visões possa criar corretamente o repositório de dados da visão. O conteúdo do repositório de dados de customização abrange:

- o mapeamento entre os tipos primitivos;
- o mapeamento da sintaxe de definição de classes;
- o mapeamento da sintaxe de definição de métodos;
- o mapeamento entre as linguagens de consulta (para povoamento);
- o mapeamento entre as funções de manipulação de dados geográficos previstos no modelo de GeoVisões e as funções oferecidas pela biblioteca espacial acoplada ao SGBDOO;

O mapeamento armazenado no repositório de dados de customização determina em que nível as funcionalidades oferecidas pelo modelo de GeoVisões será suportado pelo SGBDOO.

O modelo de GeoVisões oferece os requisitos necessários para a modelagem de visões orientadas a aplicações geográficas (seção 3.2.2). A garantia de que tais requisitos serão atendidos pelo mecanismo proposto depende do grau de impedância existente entre os requisitos do modelo de GeoVisões e as funcionalidades oferecidas pela biblioteca espacial e pelo SGBDOO. Esta impedância fica caracterizada pelo grau de dificuldade encontrada para gerar os dados de customização do mecanismo de visões. O modelo de dados oferecido pelo modelo de GeoVisões pressupõe os padrões determinados pela ODMG [Ban94], os quais devem ser seguidos pelas futuras implementações de SGBDOO comerciais. Por outro lado, os modelos de dados geométricos utilizados em SIG estão longe de serem padronizados e representam uma restrição à garantia de funcionalidade do mecanismo proposto e um grande desafio à comunidade científica envolvida em pesquisa em geoprocessamento.

Bibliografia

[AABT94] E. Apolloni, F. Arcieri, L. Barella, and M. Talamo. Requirements and Design Issues of Spatial Data Handling Systems. (347):49–69, 1994.

[AB91] S. Abiteboul and A. Bonner. Objects and Views. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 238–247, 1991.

[AGMK95] B. Adelberg, H. Garci-Molina, and B. Kao. Applying Update Streams in a Soft Real-Time Database System. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 245–256, 1995.

[AKK94] M. Arikawa, H. Kawakita, and Y. Kambayashi. Dynamic Maps as Composite Views of Varied Geographic Database Servers. *Lecture Notes in Computer Science*, (819):142–157, 1994.

[ANS75] ANSI. American National Standards Institute Study Group on Database Management Systems. Technical report, ANSI, 1975.

[Ban94] F. Bancilhon. Object Database Systems: the ODMG Standard. Technical Report 12, O_2 Technology, 1994.

[Bee89] C. Beeri. Formal Models for Object-oriented Databases. In *Proc. 1st International Conference on Deductive and Object-oriented Databases*, pages 370–395, 1989.

[Ber91] E. Bertino. A View Mechanism for Object-Oriented Databases. In *Proc. 3rd EDBT Conference*, pages 137–151, 1991.

[Ber92] E. Bertino. Data Hiding and Security in Object-Oriented Databases. In *Proc IEEE Data Engineering Conference*, pages 338–347, 1992.

[BH88] E. Bertino and L.M. Haas. Views and Security in Distributed Database Management System. *Lecture Notes in Computer Science*, (303):155–169, 1988.

[BLT86] J.A. Blakeley, P.-A. Larson, and F.W. Tompa. Efficiently Updating Materialized Views. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 61–71, 1986.

[Bra92] S.E. Bratsberg. Unified Class Evolution by Object Oriented Views. *Lecture Notes in Computer Science*, (645):423–439, 1992.

[Cam95] Gilberto Camara. *Modelos, Linguagens e Arquiteturas para Bancos de Dados Geográficos.* PhD thesis, Instituto Nacional de Pesquisas Espaciais, December 1995.

[CCH+96] G. Camara, M.A. Casanova, A.S. Hemerly, G.C. Magalhães, and C.B. Medeiros. *Anatomia de Sistemas de Informação Geográfica.* Instituto de Computação - UNICAMP, July 1996.

[CdFvO93] E. Clementini, P. di Felice, and P. van Oosterm. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *3rd Sympoium on Spatial Database Systems*, pages 277–295, 1993.

[CFS+94] G. Camara, U.M. Freitas, R.C.M. Souza, M.A. Casanova, A.S. Hemerly, and C.B. Medeiros. A model to cultivate objects and manipulate fields. *2nd Workshop on Advances in GIS*, pages 30–37, December 1994.

[Cif95] R.R. Ciferri. Um Benchmark voltado à Análise de Desempenho de Sistemas de Informação Geográfica. Master's thesis, DCC - UNICAMP, 1995.

[dA95a] C.D. de Aguiar. Integração de Sistemas de Bancos de Dados Heterogêneos em Aplicações de Planejamento Urbano. Master's thesis, DCC - UNICAMP, 1995.

[dA95b] E. A. de Aguiar. Planejamento Ambiental como instrumento à prevenção de doenças infecto-contagiosas e parasitárias - Estudo de Caso: Paulínia - SP. Master's thesis, Faculdade de Engeharia Civil - UNICAMP, 1995.

[DGS94] J. Durand, M. Ganti, and R. Salinas. Object View Broker: A Mediation Service Architecture to Provide Object-Oriented View. *Lecture Notes in Computer Science*, (819):412–428, 1994.

[dS95] C.S. dos Santos. Design and Implementation of Object Oriented Views. In *The International Conference on Data and Expert Systems Applications (DEXA)*, pages 91–102, 1995.

[dSAD93] C.S. dos Santos, S. Abiteboul, and C. Delobel. Virtual Schemas and Bases. In *Proceedings of the Conferende EDBT*, pages 335–353, 1993.

[EF91] M.J. Egenhofer and R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information*, 5(2):161–174, 1991.

[Ege95] M. J. Egenhofer. Naive Geography. *Lecture Notes in Computer Science*, (988):1–16, 1995.

[FD91] A.V. Frank and D.M.Mark. Language Issues for GIS. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 11, pages 147–163. Longman Scientific and Technical, 1991.

[Fre75] J. Freeman. The modelling of Spatial Relations. *Computer Graphics and Image Processing*, 4:156–171, 1975.

[FSdS79] A.L. Furtado, K. Sevcik, and C.S. dos Santos. Permitting Updates through Views of Databases. *Information Systems*, 4:269–283, 1979.

[Gat91] A.C. Gatrell. Concepts of Space and Geographical Data. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 9, pages 119–134. Longman Scientific and Technical, 1991.

[GM95] A. Gupta and I.S. Mumick. Maintenance of Materialized Views : Problems, Techniques, and Applications. *Data Engineering*, 18(2):3–18, 1995.

[GMS92] A. Gupta, I.S. Mumick, and V.S. Subrahmanian. Maintaining Views Incrementally. Technical Report TR 921214-19-TM, ATT Bell Labs., 1992.

[GPST94] A. Gutierrez, P. Pucheral, H. Steffen, and J.-M. Thevenin. Database Graph Views: A Practical Model to Manage Persistent Graphs. In *Proceedings of the 20th VLDB Conference*, pages 391–402, 1994.

[Gut94] R. H. Guting. An Introduction to Spatial Databases. In *The VLDB Journal*, volume 3, pages 357–400. VLDB Endowment, October 1994.

[Her93] D. Hernàndez. Maintaining Qualitative Spatial Knowledge. *Lecture Notes in Computer Science*, (716):36–53, 1993.

[HR95] W. Huang and G.A. Riccardi. Modeling and Implementing Dynamic Object-Oriented Views. pages 184–199, 1995.

[HZ88] S. Heiler and S. Zdonick. Views, Data abstraction and inheritance in the FUGUE Data model. In *Lecture Notes in Computer Science*, volume 334, pages 225–241. Springer Verlag, 1988.

[HZ90] S. Heiler and S. Zdonik. Object Views: Extending the Vision. *Proceedings International IEEE Data Engineering Conference*, pages 86–97, February 1990.

[KK95] W. Kim and W. Kelley. *Modern Database Systems*, chapter 6, pages 108–129. ACM Press, New York, 1 edition, 1995.

[LL92] F.W. Ling and M.L. Lee. A Thoery for Entity-Relationship Views Updates. *Lecture Notes in Computer Science*, (645):262–279, October 1992.

[LMSS95] J.J Lu, G. Moerkotte, J. Schue, and V.S. Subrahmanian. Efficient Maintenance of Materialized Mediated Views. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 340–351, 1995.

[MB96] C.B. Medeiros and M. Botelho. Managing Time in GIS. In GIS *Brasil*, pages 534–544, 1996.

[Med94] C.B. Medeiros. Extending Views to Suport GIS Applications. Private Communication, 1994.

[Mey96] B. Meyer. The many faces of Inheritance: A taxonomy of taxonomy. *IEEE Computer*, 29(5):105–108, May 1996.

[MM91] J.C. Mamou and C.B. Medeiros. Interactive Manipulation of Object-Oriented Views. *Proceedings International IEEE Data Engineering Conference*, pages 21–28, 1991.

[MMC94] C.B. Medeiros, M.A.Casanova, and G. Camara. The DOMUS Project-Building an OODB GIS for Environmental Control. *International Journal of Geographical Information*, (884):45–54, 1994.

[MP94] C.B. Medeiros and F. Pires. Databases for GIS. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 23(1):107–115, 1994.

[OM95] J.L. Oliveira and C.B. Medeiros. A Direct Manipulation User Interface for Querying Geographic Databases. In *Proceedings of ADB'95*, 1995.

[OPM97] J. L. Oliveira, F. Pires, and C. M. Medeiros. UAP'E - an Environment for Integrated Modelling and Analysis of Geographic Information. *GeoInformatica*, 1997. to appear.

[OR92] F. Olken and D. Rotem. Maintenance of Materialized Views of Sampling Queries . *Proceedings International IEEE Data Engineering Conference*, pages 632–641, 1992.

[PMSL94] H. Pirahesh, B. Mitschang, N. Sudkamp, and B. Linday. Composite-Object Views in Relational DBMS: An Implementation Perspective. *Information Systems*, 19(1):69–88, 1994.

[SE90] J. Star and J. Estes. *Geographical Information Systems: An Introduction.* Prentice Hall, Inc., 1990.

[SJGP90] M. Stonebraker, A. Jhingram, J. Goh, and S. Potamianos. On rules, Procedures, Caching and Views in Data Base Systems. *ACM Transactions on Database Systems*, pages 281–290, 1990.

[SS89] J. Shilling and P. Sweeney. Three Steps to Views: Extending the Object-Oriented Paradigm. In *Proceedings of ACM Conference on Object-Oriented Programing Systems, Language and Aplications*, pages 353–361, 1989.

[Sto75] M. Stonebraker. Implementation of Integrity Constraints and Views by Query Modification. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 281–290, 1975.

[TAPR95] K. Tanskanen, A. Aaltonen, D. Paasiala, and A. Riitahuhta. Interfacing of Object-oriented Databases and Knowledge based Engineering Systems using Views. 1995.

[Tom93] A. Tomasić. Correct View Update Translations via Containment. Technical Report STAN-CS-TN-93-3, Stanford University, 1993.

[Vie93] L. Vieu. A Logical Framework for Reasoning about Space. *Lecture Notes in Computer Science*, (716):25–35, 1993.

[Wie86] G. Wiederhold. Views, Objects and Databases. *IEEE Computer*, 19(12):37–44, 1986.

[Wor95] M.F. Worboys. *GIS: A Computing Perspective.* Taylor & Francis Ltd, 1995.

[YA95] M. Yuan and J. Albrecht. Structural Analysis of Geographic Information and GIS Operations from a User Perspective. *Lecture Notes in Computer Science*, (988):107–122, 1995.

[ZGMHW95] Y. Zughe, H. Garcia-Molina, J. Hammer, and J. Widom. View Maintenance in a Warehousing Environment. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 316–327, 1995.

Bibliografia Adicional

[AAT94] E. Apolloni, F. Arcieri, and M. Talamo. Environments for Land Application Development: Issues and Design Guidelines. *Lecture Notes in Computer Science*, (884):1–14, 1994.

[ABC+91] J. Antenucci, K. Brown, P. Croswell, M. Kevany, and H. Archer. *Geographic Information Systems - a Guide to the Technology.* Van Nostrand Reinhold, 1991.

[ANT94] F Arcieri, E. Nardelli, and M. Talamo. CARTECH: a prototype of geographical information system. *Lecture Notes in Computer Science*, (884):178–191, 1994.

[Bau94] P. Baumann. Management of Multidimensional Discrete Data. In *The VLDB Journal*, volume 3, pages 401–444. VLDB Endowment, October 1994.

[BK94] T. Brinkhoff and H.-P. Kriegel. Approximations for a Multi-Step Processing of Spatial Join. *Lecture Notes in Computer Science*, (884):25–34, 1994.

[Cil96] M.A. Cilia. Mecanismos Ativos para Manutenção de Restrições Topológicas em Sistemas de Informação Geográficas. Master's thesis, Insituto ce Computação - UNICAMP, 1996.

[CIT94] W.W. Chu, I.T. Ieong, and R.K. Taira. A Semantic Modeling Approach for Image Retrieval by Content. In *The VLDB Journal*, volume 3, pages 445–477. VLDB Endowment, October 1994.

[CZ95] E.P.F. Chan and R. Zhu. QL/G - a query language for geometric data bases. Department of Computer Science - University of Waterloo, Canada, February 1995.

[DdST95] C. Delobel, C.S. dos Santos, and D. Tallot. Object Views of Relations. pages 14–18, 1995.

[Den91] P.J. Densham. Spatial Decision Support Systems. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 26, pages 403–412. Longman Scientific and Technical, 1991.

[DKL+94] D. J. DeWitt, N. Kabra, J. Luo, J. M. Patel, and J.-B Yu. Client-Server Paradise. *Proceedings of the International Conference on Very Large Data Bases*, 20(1):558–569, 1994.

[DR94] P. Dechamboux and C. Roncancio. *Peplom*^{d}: and Object Oriented Database Programming Language Extended with Deductive Capabilitites. In *The International Conference on Data and Expert Systems Applications (DEXA)*, pages 2–14, 1994.

[DRSM93] B. David, L. Raynal, G. Schorter, and V. Mansart. GeO_2: Why objects in a Geographic DBMS? *Lecture Notes in Computer Science*, 1(692), 1993.

[dS95] C.S. dos Santos. *O_2 Views - User Manual (version 2)*. O_2 Technology, January 1995.

[dSM91] C.M.G. da Silva and C.B. Medeiros. Um Mecanismo de Visões para SGDB Orientados a Objetos. Private Communication, 1991.

[DT88] S. Danforth and C. Tomlinson. Type Theories and Object-Oriented Programming. *ACM Computig Surveys*, 20(1):29, March 1988.

[Edd93] J. A. Eddey. Environmental Research: what we must do? In M.F. Goodchild, B.O. Parks, and L.T. Steyaert, editors, *Environmental Modelling with GIS*, volume 1, chapter 1, pages 3–7. Oxford Universty Press, 1993.

[Ege92] M.A. Egenhofer. Why not SQL? *International Journal of Geographical Information*, 2(6):71–86, 1992.

[EH91] M.J. Egenhofer and J.R. Herring. High-level Spatial Data Strucutures for GIS. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 16, pages 227–237. Longman Scientific and Technical, 1991.

[Fed93] K. Fedra. GIS and Environmental Modeling. In M.F. Goodchild, B.O. Parks, and L.T. Steyaert, editors, *Environmental Modelling with GIS*, volume 1, chapter 5, pages 35–50. Oxford Universty Press, 1993.

[Flo91] R. Flowerdew. Spatial Data Integration. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 24, pages 375–387. Longman Scientific and Technical, 1991.

[Goo93] M.F. Goodchild. The state of GIS Environmental Problem-Solving. In M.F. Goodchild, B.O. Parks, and L.T. Steyaert, editors, *Environmental Modelling with GIS*, volume 1, chapter 2, pages 8–15. Oxford Universty Press, 1993.

[Hea91] R.G. Healey. Database Management Systems. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 18, pages 251–267. Longman Scientific and Technical, 1991.

[Kan92] Kanellakis and Lécluse and Richard. Introduction to the Data Model. In F. Bancilhon and C.Delobel and P. Kanellakis, editor, *Building and Object-Oriented Database System: The story of O_2*, chapter 3. Morgan Kauffman Publishers Inc., 1992.

[KPS95] G. Korsters, B.-U. Pagel, and H.-W. Six. Object-Oriented Requirements Engineering for GIS-Applications. *3rd ACM Workshop on Advances in Geographic Information Systems*, pages 61–69, 1995.

[KPS96] G. Korsters, B.-U. Pagel, and H.-W. Six. Object-Oriented Analysis for Geographic Information Systems. *2nd IEEE International Conference on Requirements Egineering - Colorado Springs*, 1996.

[Lor94] M. De Lorenzi. The XYZ GeoServer for Geometric Computation. *Lecture Notes in Computer Science*, (884):202–213, 1994.

[MD91] D.J. Maguire and J. Dangermond. The Functionality of GIS. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 21, pages 319–335. Longman Scientific and Technical, 1991.

[NM95] H. Naja and N. Mouaddib. The Multiple Representation in Architectural Application. 1995.

[Ope91] S. Openshaw. Developing Appropriate Spatial Analysis Methods for GIS. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 25, pages 309–402. Longman Scientific and Technical, 1991.

[Par93] B.O. Parks. The need for Integration. In M.F. Goodchild, B.O. Parks, and L.T. Steyaert, editors, *Environmental Modelling with GIS*, volume 1, chapter 4, pages 31–34. Oxford Universty Press, 1993.

[RS94] P. Rigaux and M. Scholl. Multiple Representation Modelling and Querying. *Lecture Notes in Computer Science*, (884):59–69, 1994.

[RW94] L. Relly and A. Wolf. A Storage Manager for the Development of Spatial Data Structures. *Lecture Notes in Computer Science*, (884):168–177, 1994.

[She91] I.D.H. Shepherd. Information Integration and GIS. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 22, pages 337–360. Longman Scientific and Technical, 1991.

[Sta93] L.T Stayaert. A Perspective on the State od Environmental Simulation Modeling. In M.F. Goodchild, B.O. Parks, and L.T. Steyaert, editors, *Environmental Modelling with GIS*, volume 1, chapter 3, pages 16–30. Oxford Universty Press, 1993.

[TK93] T. Takahashi and A.M. Keller. Querying Heterogeneous Object Views of a Relational Database. Technical report, Stanford University, 1993.

[TYI88] K. Tanaka, M. Yoshikawa, and K. Ishihara. Schema Virtualization in Object Oriented Databases. *IEEE Computer*, pages 23–30, 1988.

[WLON94] A. Wolf, M. De Lorenzi, T. Ohler, and V.H. Nguyen. Cartech: a prototype of geographical information system. *Lecture Notes in Computer Science*, (884):192–201, 1994.

[YPS95] Y. Ye, C. Parent, and S Spaccapietra. On the Specification of Views in DOOD Systems. *4th International Conference* DOOD *'95*, pages 539–555, 1995.