Versões em Bancos de Dados para SIGs*

Luis M. del Val Cura† Claudia Bauzer Medeiros
{delval,cmbm}@dcc.unicamp.br
IC - UNICAMP-CP6176
13081-970 Campinas, SP

Sumário

Sistemas de Informação têm introduzido modelos e mecanismos de versões para o gerenciamento de múltiplos estados ou variações das entidades modeladas. As aplicações fundamentais de versões estão associadas à manutenção de alternativas de projetos em sistemas CAD/CASE e na representação histórica de entidades em sistemas temporais. Este artigo discute as possíveis aplicações de versões em SIG enfatizando aspectos relativos a aplicações temporais, manipulação de múltiplas representações e facilidades para manutenção de alternativas de projeto em um contexto espacial. O trabalho propõe um modelo e mecanismo de versões com recursos básicos para dar apoio a estas aplicações, mostrando como estender um SGBDOO para este fim, viabilizando desta forma várias operações hoje não contempladas em SIG

1 Introdução

Um Sistema de Informações Geográficas (SIG) é um software que gerencia grandes volumes de dados geo-referenciados, isto é, referenciados em relação à sua posição sobre a superfície terrestre. O usuário relaciona e combina estes dados através de funções de análise e processamento espacial para garantir os objetivos de sua aplicação. Os SIGs modernos usam em geral algum banco de dados espacial, onde os dados geo-referenciados são armazenados.

A manipulação de dados por parte de um SIG típico apresenta atualmente, dentre outras, as seguintes limitações:

- Não permite manipulação de dados temporais.
- Não considera a possibilidade de múltiplas representações espaciais.
- Não suporta gerenciamento de cenários alternativos.

Este artigo apresenta uma solução para estas deficiências, baseada em acoplar ao banco de dados espacial do SIG um mecanismo de gerenciamento de versões. Esta solução parte da constatação de que, na prática, todas estas limitações envolvem implicitamente a necessidade de gerenciamento de múltiplos estados (as versões) de uma mesma entidade do mundo real.

As principais contribuições apresentadas são portanto as seguintes:

- Discussão dos problemas em aberto em SIG que podem ser solucionados pelo uso de versões.

*Projeto parcialmente financiado pelo CNPq, FAPESP, Protem(projeto GEOTEC) e CEE (projeto ICDT 116)
†Professor do Departamento de Ciência da Computação, Universidad de La Habana, Cuba

- Apresentação de um modelo e mecanismo de versões voltado a dar apoio às necessidades de usuários de SIG.
- Esboço da implementação do mecanismo em um SGBDOO.

O restante do texto está dividido da seguinte forma. A seção 2 dá uma breve descrição de trabalhos correlatos. A seção 3 analisa as atuais necessidades de SIG, indicando aquelas que podem ser respondidas através do uso de versões. A seção 4 propõe o modelo de versões e a seção 5 descreve o mecanismo para implementá-lo. A seção 6 mostra como implementar o mecanismo em um SGBDOO. Finalmente, a seção 7 apresenta conclusões e extensões.

2 Trabalhos correlatos

Um sistema de informação com possibilidades para versionamento é aquele no qual podem ser mantidos múltiplos estados ou variações das entidades modeladas. O uso de versões é encontrado em diversos contextos. Inicialmente, surgiu associado basicamente a aplicações em sistemas CAD [CK86, Kat90, TOC93] visando a manutenção das múltiplas alternativas de projetos desenvolvidos por vários usuários. Estes modelos foram sintetizados e generalizados por [Kat90], propondo uma terminologia atualmente utilizada na área. Versões têm também sido amplamente utilizadas em sistemas CASE para a manutenção de múltiplas configurações de sistemas de software [HK87, SS94, BCJ92] envolvendo problemas similares aos encontrados na área de CAD. Necessariamente, muitos dos mecanismos de versões desenvolvidos neste contexto refletem características da semântica destes sistemas e em geral não possuem uma abordagem de integração a um SGBD, isto é, o versionamento aparece como funcionalidade *ad hoc*.

Ainda outro contexto de uso de versões é o de *dados temporais*. Sistemas de bancos de dados temporais [Sno90] têm utilizado formas de manipulação de versões. Estas são, no entanto, orientadas à manutenção dos diferentes estados temporais das entidades, que geralmente são organizados de forma linear - i.e., o tempo evolui linearmente.

Mais recentemente, os sistemas de bancos de dados orientados a objetos têm incorporado facilidades para o gerenciamento de versões [Gol93, BH89, KBC+89, CJ90, Sci91]. Alguns modelos adotados por estes sistemas foram influenciados pelas propostas desenvolvidas na área de CAD [KBC+89]. Outros modelos, como o proposto em [CJ90] para o sistema O_2, ou em [Sci91], tentam definir propostas gerais que sejam igualmente aplicáveis a sistemas temporais, CAD ou CASE.

A idéia de usar versões em SIGs é ainda pouco difundida. No entanto, vários dos problemas existentes nos sistemas geográficos atuais seriam solucionados se estes contemplassem uso de versões. Por exemplo, [Lan93, CT95, Bot95] propõem o uso de versões para resolver questões de gerenciamento espaço-temporal. [MJ93b, MJ93a] apresentam possíveis soluções de implementação de múltiplas representações a partir do uso de versões. Finalmente [Bat92, VFMa95] apresentam o uso de versões para projeto espacial, com características semelhantes às apresentadas em ambientes CAD e CASE.

3 Versões em SIG

Há três aspectos em SIG onde mecanismos de versionamento podem ajudar: projeto espacial, múltiplas representações e modelagem espaço-temporal.

3.1 Versões para projeto em SIG

O conceito de projeto, em SIG, está ligado à noção de planejamento (urbano e ambiental), onde vários usuários podem trabalhar cooperativamente. A introdução de facilidades para projeto em SIG deve permitir que : (1) usuários delimitem, usando o banco de dados, uma *região geográfica para trabalho*, onde várias entidades espaciais serão manipuladas ; e (2) dentro desta região, seja possível simular diferentes *cenários* ou *alternativas*. Por exemplo, um projeto urbano para expansão de um bairro é executado sobre uma região que inclui a área do bairro e áreas vizinhas; e as entidades espaciais manipuladas são ruas, edificações, relevo, etc.

A unidade de trabalho para projeto em SIG é semelhante à noção de *configuração* em versões (ou seja, um conjunto básico de entidades que constitui uma unidade semântica consistente). Alguns elementos devem ser considerados para a criação de configurações em SIG:

- A configuração é espacial, sendo associada a uma área alvo de trabalho, isto é, o usuário poderá transformar somente os objetos dentro do espaço (região) predeterminado.
- As entidades dentro da região podem ser consideradas como objetos complexos, interrelacionados por agregação e composição. A noção de composição se traduz em geral por relacionamentos topológicos (o objeto composto contém "espacialmente" seus componentes espaciais).

Assim, um modelo e mecanismo de versões para projetos em SIG deve garantir o gerenciamento de configurações espaciais. Para permitir atividades de trabalho cooperativo é preciso considerar a definição de espaços de trabalho individuais e coletivos. Finalmente, como em ambientes de projeto CAD, é necessário permitir operações de *check-in, check-out* e possibilidades de cópias e *merge* entre espaços de trabalho

3.2 Versões para múltiplas representações

O problema da representação múltipla das entidades envolvidas na modelagem de dados está presente em diversas áreas de aplicação [NM95]. De fato, distintos usuários representam (percebem, manipulam e utilizam) de maneira diferente uma mesma entidade. Esta multiplicidade se traduz em valores, esquemas ou funcionalidade diferentes para cada representação.

No caso da modelagem espacial, o problema da manipulação de múltiplas representações é mais complexo. Uma entidade espacial pode ter diferentes formas de representação que correspondem a variações na resolução, escala, projeções cartográficas, dentre outras. Uma mesma entidade geográfica pode assim ser representada (e armazenada) usando simultaneamente diversas estruturas

Uma proposta de mecanismo de versões para resolver o problema das múltiplas representações deve considerar os seguintes aspectos:

- Deve garantir independência entre modelagem conceitual e implementação.
- Deve permitir que uma entidade espacial tenha diferentes representações.
- Deve providenciar mecanismos para garantir a consistência entre as diferentes representações ante operações que mudam características espaciais da entidade.

3.3 Versões para a modelagem temporal em SIG

Um dos objetivos mais importantes no desenvolvimento atual dos SIGs é a inclusão nestes de possibilidades para a análise das mudanças espaciais no tempo [Lan93, Peu94]. Isto inclui o problema da

variação das estruturas e representações geométricas e dos relacionamentos topológicos no tempo. A evolução de objetos geográficos no tempo (evolução espaço-temporal) pode ser modelada a partir de um conjunto de eventos, que modificam a geometria ou a localização de um objeto, de conjuntos de objetos e de seus relacionamentos espaciais [CT95].

Para permitir a modelagem da evolução espaço-temporal, um mecanismo de versões deve considerar os seguintes aspectos:

- Representação da evolução temporal de cada objeto espacial a partir das suas versões.
- Manutenção dos diferentes estados temporais do banco de dados, isto é, o registro de todos os eventos que produzem alguma mudança em alguma entidade modelada.
- Manutenção explicita de relacionamentos entre entidades diferentes em estados temporais diferentes.

4 O modelo de versões proposto

O modelo proposto é resultado das constatações feitas na seção 3. Ele pressupõe o uso de um BD espacial orientado a objetos que contém objetos espaciais e convencionais. Estes objetos podem ser *versionáveis* e *não versionáveis*. Cada objeto versionável tem múltiplos estados. Cada estado de um objeto versionável corresponde a uma *versão* do objeto. Todas as versões de um objeto compartilham a mesma identidade, isto é, uma versão de um objeto sempre é manipulada como ligada ao *oid* do objeto. A formalização do modelo se encontra em [dV97], sendo omitida por razões de espaço.

4.1 Conceitos principais.

O modelo é baseado em três conceitos principais: *contexto espaço-temporal* (CET), *espaço de trabalho* (ET) e *contexto de trabalho* (CTR). Um CET é um estado espaço-temporal consistente, sendo formado por um conjunto de versões de objetos. O BD é formado por um conjunto de CETs. Em outras palavras, cada CET corresponde a uma versão do mundo modelado segundo uma perspectiva diferente – ou seja, o BD contém vários estados consistentes do mundo. Por exemplo, os fenômenos de uma região se forem armazenados em múltiplas escalas, dentro do BD pode-se distinguir um CET para cada escala. Se além disso, houver dados armazenados em várias projeções, haverá um CET para cada par [escala, projeção] e assim por diante.

Um *espaço de trabalho* (ET) corresponde ao conceito de *banco de dados privado* (ou de trabalho) em um ambiente CAD. O conceito de ET é introduzido para permitir trabalho cooperativo. Cada ET é caracterizado pela área alvo que abrange (região geográfica). Em outras palavras, trata-se de um banco de dados secundário criado a partir do banco de dados original através de seleção de determinados objetos e suas versões, restritos a uma região alvo.

Por exemplo, suponha que o banco de dados contém objetos versionados segundo escala e tempo (ou seja, contém vários CETs que diferem em valores de [escala , tempo]) para todas as cidades de São Paulo. Seja um projeto para Campinas. Um determinado grupo de usuários pode querer trabalhar apenas com escalas 1:1000 e 1:5000, para qualquer valor de tempo. Outro grupo pode querer trabalhar com escalas 1:5000 e 1:10000 para valores de tempo t_1 e t_2. No primeiro caso, um ET é criado (a) definindo *Área-alvo* = *Campinas* e (b) selecionando todas as versões de objetos

nesta área que têm escala 1:1000 e/ou 1:5000, ao longo de todo o tempo. No segundo caso, o ET é restrito aos objetos de Campinas nas escalas 1:10000 e 1:5000 e seus estados nos tempos t_1 e t_2.

Como um ET é um (sub)banco de dados, também tem vários contextos espaço-temporais, denominados CTR (um CTR é um CET de *de trabalho* restrito a um ET). Por exemplo, o segundo ET descrito contém 4 CTR: o primeiro corresponde a versões de objetos no tempo t_1 e escala 1:10000; o segundo a [t_1,1:5000]; o terceiro [t_2,1:10000] e o quarto [t_2,1:5000].

O trabalho cooperativo de diferentes usuários pode estabelecer relacionamentos de dependência entre versões de um mesmo objeto em diferentes contextos, tanto CET ou CTR. Estes relacionamentos indicam o compartilhamento dessas versões pelos diferentes contextos, isto é, quando em um contexto a versão é modificada, esta mudança deve ser refletida no outro contexto.

4.2 Ciclo de trabalho no modelo

A figura 1 mostra os relacionamentos entre CET, ET e CTR. O conjunto de CET representa o mundo geográfico modelado sobre o qual são realizadas as operações tradicionais de um BD: criação e eliminação de objetos, atualização e consultas.

Um ET é criado a partir de *Check out* de versões de objetos do banco de dados. Equipes de usuários podem trabalhar paralelamente em ETs distintos, criando versões que finalmente podem ser incorporados ao banco de dados através de operações de *check in*, refletidas em atualizações do banco de dados.

Para o caso do trabalho em equipe, os espaços de trabalho associados a diferentes usuários dentro de um projeto podem estruturar espacial e hierarquicamente seus objetos. Os relacionamentos de dependência garantem a possibilidade do intercâmbio de informação entre os diferentes usuários, ETs e CTRs.

4.3 Operações do modelo

O modelo define quatro tipos de operações: operações sobre CET, sobre ET, sobre CTR e sobre objetos individuais e suas versões.

As operações sobre CET são criação, eliminação, definição de relacionamentos entre CETs e *check out* para criar um ET. De forma semelhante, CTRs podem ser criados, eliminados e ter relacionamentos estabelecidos. ETs podem ser criados (a partir de *check out*) ou eliminados (fazendo *check-in* sobre o banco de dados). Objetos podem ser criados, eliminados ou modificados (gerando novas versões). Além disto, é possível estabelecer relacionamentos entre versões de objetos.

Os relacionamentos de dependência entre CTRs não se restringem a um ET, mas podem ser feitos entre CTR de ET diferentes, como mostra a figura 1. No caso, CTR_2 e CTR_a são inter-relacionados porque uma parte dos seus objetos é comum (parte sombreada). A figura mostra também que CETs podem ser relacionados (por exemplo, se um CET é criado de outro a partir de derivação). O gerenciamento destes relacionamentos é um fator complicador.

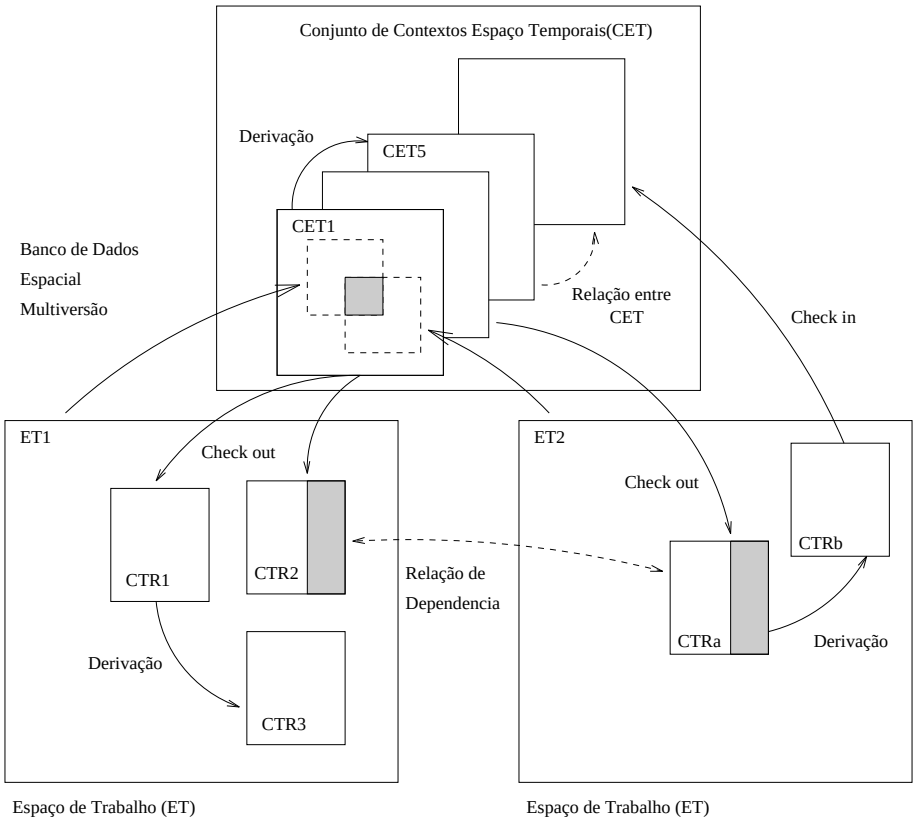


Figura 1: O Modelo Proposto

5 Mecanismo de versões

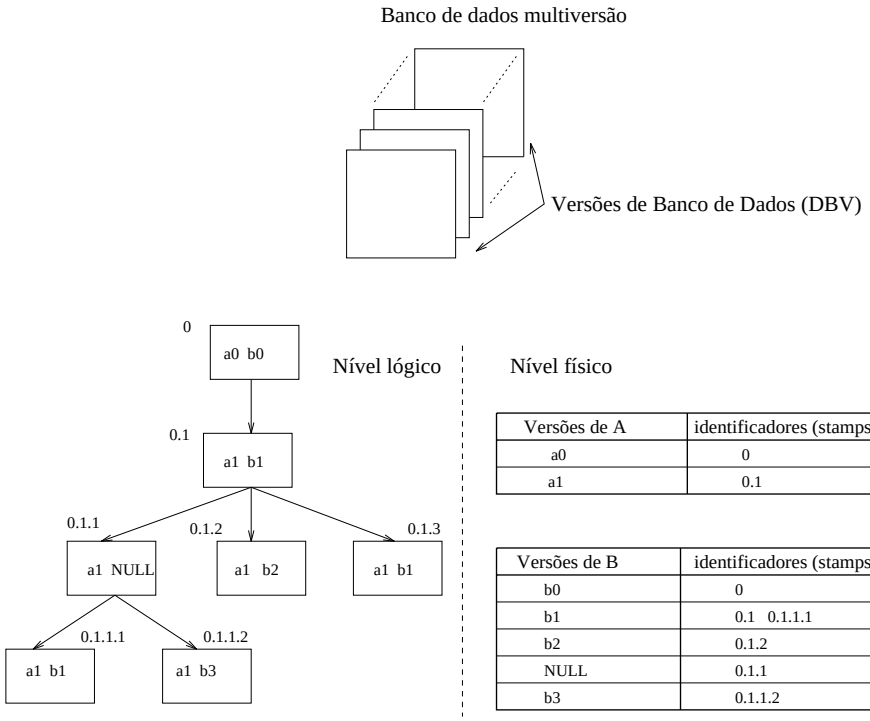
5.1 Mecanismo base: MDBV (Sistema O_2)

.

O mecanismo proposto é baseado na extensão do modelo MDBV (*multi-version database* ou BD multiversão) [CJ90, Gan94, GJ94], implementado no sistema O_2. A característica fundamental do MDBV é a distinção entre nível lógico e físico de versionamento. O nível lógico corresponde à visão do usuário enquanto o nível físico corresponde à visão de implementação.

No nível lógico, um BD é entendido como um conjunto de Versões de Banco de Dados (*database version*-DBV), isto é, a unidade de versionamento é o banco de dados como um todo. Cada DBV representa um estado diferente do mundo real modelado, contendo versões lógicas dos objetos no BD. A figura 2 mostra o modelo, para um conjunto de versões de banco de dados (DBV).

Cada versão lógica de um objeto pode ser identificada pelo par (*oid, dbvid*) onde: *oid* corresponde ao identificador do objeto e *dbvid* corresponde ao identificador da DBV na qual aquela versão ocorre. A figura 2 representa à esquerda sete DBV, cada uma com uma versão dos objetos A e B. No caso da DBV 0.1.1, o valor da versão de A é *a1* e da versão de B é NULL.



Versões de A	identificadores (stamps)
a0	0
a1	0.1

Versões de B	identificadores (stamps)
b0	0
b1	0.1 0.1.1.1
b2	0.1.2
NULL	0.1.1
b3	0.1.1.2

Figura 2: Mecanismo e Modelo MDBV

Em geral uma nova DBV é criada como uma derivação de uma DBV já existente. Os identificadores *dbvid* permitem determinar os relacionamentos de derivação. Por exemplo, na figura 2 as DBV derivadas da DBV 0.1 são 0.1.1 , 0.1.2 e 0.1.3.

No nível físico, cada objeto está formado pelo conjunto de suas versões físicas e uma *tabela de associação*. Uma mesma versão física de um objeto poderá ser compartilhada por várias versões

lógicas. Este relacionamento entre versões físicas e lógicas pode se estabelecer explicita ou implicitamente:

- *Explicitamente:* A tabela de associação estabelece o relacionamento entre uma versão física e versões lógicas, armazenando os identificadores de DBV cujas versões lógicas compartilham a mesma versão física. Quando uma versão lógica é modificada e compartilha a mesma versão física com outras versões, então uma nova entrada na tabela de associação deverá ser criada para a nova versão física resultante da modificação.
- *Implicitamente:* Todas as versões lógicas de um objeto *ob* compartilham inicialmente uma única versão física, até que *ob* seja modificado por versionamento em alguma DBV. Desta forma, quando o identificador de uma DBV não aparece explicitamente na tabela de associação, pode-se interpretar que compartilha o mesmo valor físico com sua DBV ancestral e assim recursivamente.

Na figura 2 as versões lógicas 0.1 e 0.1.1.1 do objeto B compartilham explicitamente a mesma versão física, neste caso *b1*, enquanto as versões lógicas do objeto A descendentes de 0.1 compartilham implicitamente a mesma versão física *a1*.

Para o gerenciamento do modelo, é necessário manter a nível físico a *Árvore de derivação de DBV*, também representada na figura 2. Os algoritmos de correspondência entre níveis físico e lógico são descritos em detalhe em [GJ94].

5.2 Extensão do modelo MDBV para suportar o modelo proposto

O modelo proposto pode ser mapeado para o modelo MDBV, considerando-se que cada CET e cada CTR podem ser associados de maneira natural a uma DBV. Este mapeamento exige, no entanto, estender o modelo MDBV. Estas extensões, acopladas a um *vetor de dimensões* permitem manipulação de representações e estados temporais distintos e o gerenciamento de cenários. Elas introduzem igualmente a noção de dependência entre estados. São esboçadas a seguir suas principais características. Os detalhes, descritos em [dV97], foram omitidos por falta de espaço.

5.2.1 Versionamento seletivo

O MDBV considera que todos os objetos são versionáveis e possuem uma interpretação para cada DBV. Nossa primeira extensão modifica esta hipótese para permitir o versionamento seletivo de objetos. Isto pode ser realizado identificando, na estrutura de um objeto, atributos *não versionáveis* e *versionáveis*. Estes últimos continuariam representados como o conjunto de versões físicas junto à tabela de associação. A figura 3 mostra a estrutura de dois objetos A e B utilizando esta extensão ao MDBV. O objeto A possui atributos não versionáveis os quais são representados separadamente dentro da estrutura do objeto. O objeto B, totalmente versionável, segue o MDBV. Um objeto não versionável é representado como um único estado, no modo tradicional de SGBDOO. O versionamento seletivo facilita a modelagem temporal, já que nesta nem todos os objetos/atributos variam no tempo.

5.2.2 Combinação de CET/CTR

Um CET/CTR pode ser criado por derivação ou combinação a partir de outros contextos existentes. A segunda possibilidade advém da necessidade de criar cenários em SIGs e obriga estender o

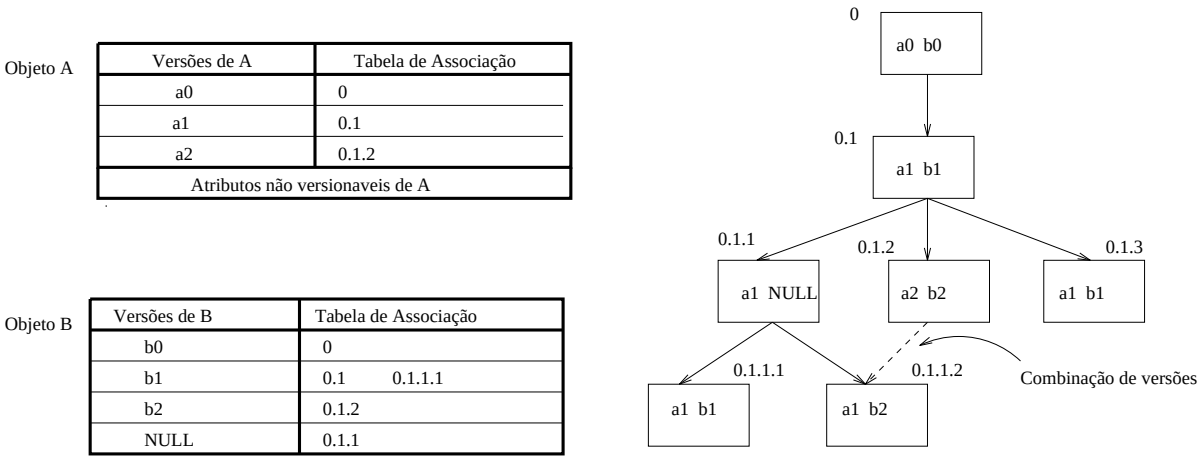


Figura 3: Versionamento seletivo

MDBV para permitir criação de DBV a partir de mais de uma DBV. A solução aqui proposta é o critério adotado em [WR95], onde a combinação de n DBV é reduzida ao problema de combinar sucessivamente duas DBV. Uma das DBV é definida como principal e a outra como secundária. As versões lógicas dos objetos na nova DBV criada serão as versões lógicas dos objetos na versão principal mais as versões lógicas dos objetos na DBV secundária que não existem na DBV principal.

Este critério garante a replicação consistente dos relacionamentos de agregação embora seu resultado nem sempre corresponda ao desejado pelo usuário quando as DBV tiverem objetos em comum. No nível físico, isto implica estender a estrutura da Árvore de Derivação de DBV, transformando-a em um *Grafo de derivação de DBV* com estrutura de Grafo Acíclico Orientado (*DAG*).

Os algoritmos de correspondência entre versão lógica e versão física para esta extensão se encontram em [dV97].

5.2.3 Relacionamentos de dependência entre contextos

O modelo introduziu relacionamentos de dependência para permitir que versões de objetos em diferentes contextos possam evoluir em conjunto. Relacionamentos de dependência são um tipo especial de restrições de integridade, que garantem a unicidade de versões de objetos em CET/CTR distintos, ou seja, as versões ob_a e ob_b de um objeto *ob* em contextos diferentes C_a e C_b são sempre idênticas. Relacionamentos de dependência podem ser mantidos garantindo-se que as versões interdependentes compartilhem a mesma versão física.

No mecanismo MDBV, no entanto, estas dependências não existem: qualquer atualização de versões implica criação de novas versões físicas afetando a tabela de associação. Já no modelo proposto, se a versão atualizada tem relacionamento de dependência com outra, a versão física compartilhada é atualizada, sem afetar a tabela.

Para permitir o relacionamento de dependência proposto, o MDBV deve ser estendido para restringir esta evolução independente. Para isto, foi criada a noção de *classe de equivalência*. Uma classe de equivalência contém todas as versões de objetos que guardam um relacionamento de dependência: uma mudança em uma versão lógica de um objeto em uma DBV será refletida nas demais versões lógicas na mesma classe de equivalência.

Do ponto de vista físico, é preciso garantir que todas as versões lógicas em uma mesma classe

de equivalência estejam associadas sempre à mesma entrada na tabela de associação. Para isto, a tabela de associação deve ser modificada, sendo criadas subtabelas para cada entrada, cada uma das quais contendo conjuntos de identificadores de DBV de uma classe de equivalência (que compartilham a mesma versão física).

Suponha (figura 4) que existe um relacionamento de dependência entre as versões do objeto A nas DBV 0.1 e 0.1.3 e as versões do objeto B nas DBV 0.1.2.1 e 0.1.1.2. Isto é traduzido nas tabelas, onde as linhas verticais particionam as classes de equivalência de instâncias (Veja a diferença para a figura 3). Suponha que o estado da versão lógica de A na DBV 0.1 (CET) é mudado de a_1 para um novo estado a_3. Neste caso, a tabela de associação é modificada garantindo que 0.1 e 0.1.3 continuem compartilhando a nova versão física a_3, enquanto as versões lógicas 0.1.1 e 0.1.2.1 explicitamente compartilham a mesma versão física a_1 mas em classes de equivalência diferentes. No mecanismo MDBV original, isso não é possível.

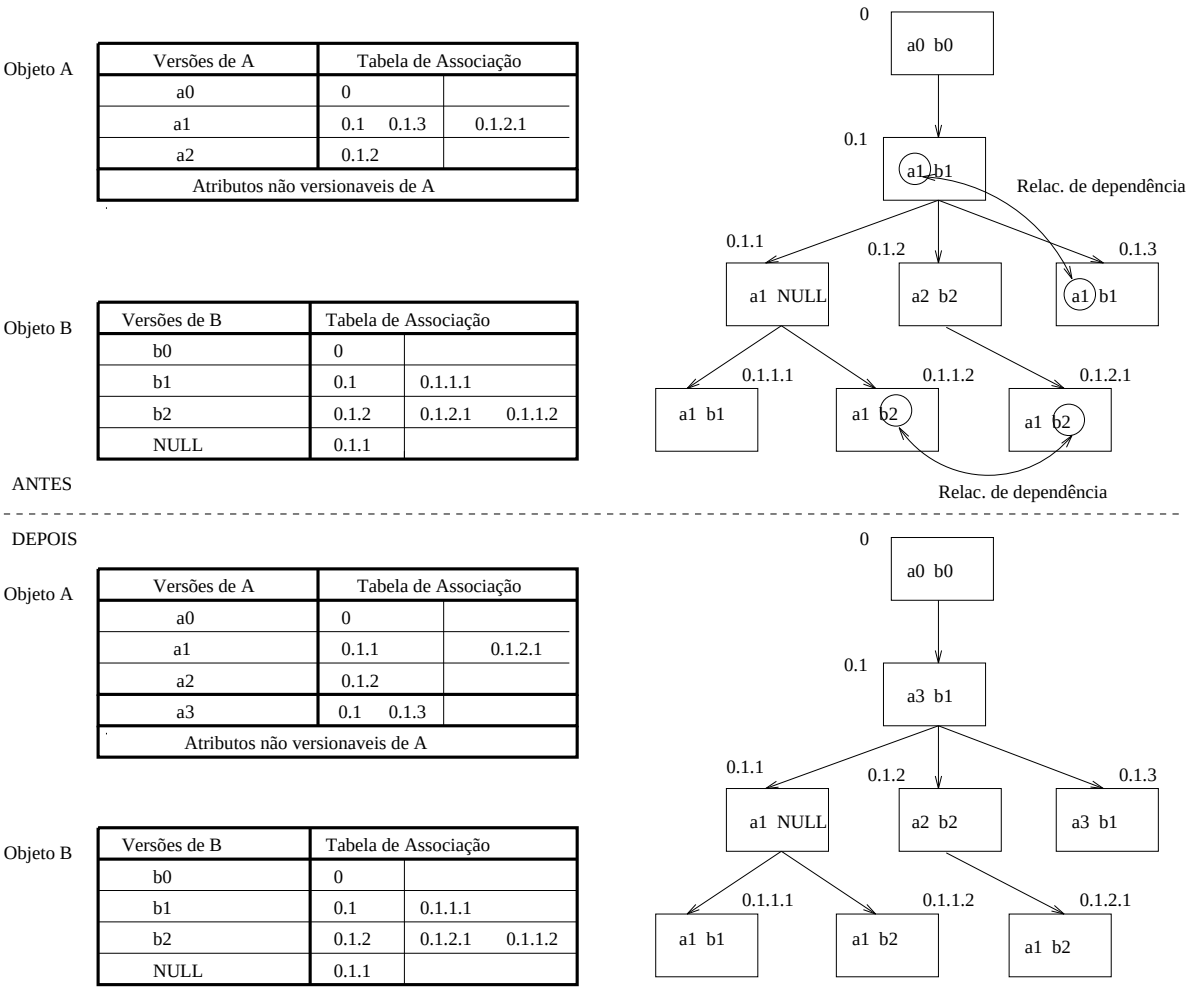


Figura 4: Dependências entre versões

6 O modelo de versões em um SGBDOO

O MDBV está implementado no SGBD O_2. As extensões propostas podem ser implementadas a partir das seguintes modificações:

- Extensões à linguagem de definição de objetos (ODL) para permitir versionamento.
- Definição de classes e objetos predefinidos, para manipular os conceitos de CET, CTR e ET.
- Extensões às linguagens de aplicação e consulta para fornecer as operações sobre versões definidas no modelo.

A extensão da ODL consiste basicamente em incluir um qualificador **versionable** para indicar quais classes/atributos são versionáveis. Modificações mais complexas são descritas a seguir resumidamente. Maiores detalhes se encontram em [dV97].

6.1 Classes e objetos predefinidos

O modelo provê dois tipos de classes predefinidos: aqueles para suportar versões (por exemplo: ET) e os relativos à dimensão espacial. Para a modelagem das últimas é introduzida uma classe abstrata **GeoObject**, similar à apresentada em [CFS+94].

Os CETs, CTRs e ETs são definidos como objetos não versionáveis do Banco de Dados, sendo instâncias de classes predefinidas básicas: **StContext, WrkContext, WrkSpace** (modelando CET, CTR e ET respectivamente). Cada objeto da classe **WrkSpace** é definido sobre uma área alvo e corresponde a um conjunto de CTR (instâncias da classe **WrkContext**).

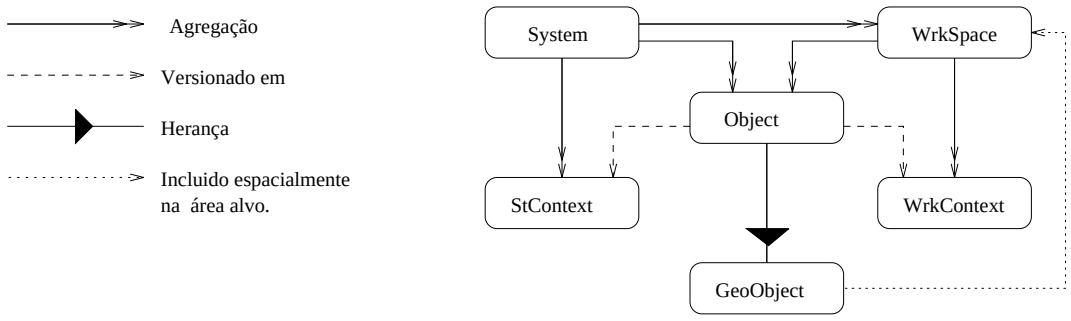


Figura 5: Classes predefinidas do modelo

A classe **System** oferece funcionalidade para o acesso aos conjunto de CETs e de ETs e para a definição do contexto *ativo*, o contexto (CET ou CTR) mais recente em uso em uma sesão de trabalho. Esta classe garante persistência de todos os CET e ET. A inicialização de um BD corresponde à criação de uma instância de **System** e sua atribuição a um objeto persistente *Sys*. Esta inicialização gera um primeiro CET, a partir do qual novos CETs podem ser derivados.

O usuário interage com as versões a partir da seleção de instâncias das classes **StContext**, **WrkContext** ou de suas subclasses. O acesso a um CTR Ctr_i exige inicialmente acesso a *Sys* e, em seguida, seleção do ET ao qual Ctr_i pertence. Esta hierarquia de acesso facilita restrições de acesso segundo diferentes critérios, visando segurança e privacidade no uso de espaços de trabalho.

A figura 5 descreve os relacionamentos entre as diferentes classes predefinidas, onde **Object** é a classe genérica do O_2 que é raiz de todas as classes de um BD (**GeoObject** é subclasse de **Object**). Uma instância da classe **System** possui uma agregação de instâncias das hierarquias **StContext** e **WrkSpace**. Por sua vez, uma instância de **WrkSpace** possui uma agregação de instâncias de **WrkContext** e os objetos versionáveis que pertencem a um **WrkSpace** estão incluídos espacialmente na área alvo associada.

6.2 Linguagem de aplicações estendida

As extensões introduzidas à linguagem estão relacionadas a três aspectos: criação de objetos versionáveis, acesso às versões de objetos nos contextos e versionamento de corpos de métodos.

Modificação na sintaxe inclui uma variável de contexto *ctx*, que referencia o CET ou CTR sobre o qual o objeto é criado. O acesso à versão de um objeto em um contexto (StContext ou WrkContext) é realizado usando uma referência à variável de contexto. A sintaxe introduzida é:

obj[ctx]...

onde:
- obj : referência a uma instância de uma classe versionável.
- ctx : referencia a uma instância das hierarquias **StContext, WrkContext**.

Assim, o comando

O2 Farm c = **new** Farm [ctx];

cria uma versão do objeto da classe **Farm** no contexto *ctx*.

6.2.1 Versionamento de métodos

Diferentes versões do corpo de um método de uma classe versionável poderiam ser associadas dinamicamente a contextos diferentes de acordo com uma sintaxe similar:

Classe [ctx].metodo M = nome de versão

interpretada como "no contexto **ctx** o método **M** está associado à versão indicada".

6.3 Linguagem de consulta (OQL) estendida

A extensão da linguagem de consulta para o modelo proposto deve permitir que diferentes versões de objetos possam ser utilizadas como parte de uma consulta (por exemplo, combinando dados em escalas diferentes). Isto é realizado usando o conceito de variável de contexto.

Como, no modelo, os contextos são objetos, coleções de contextos podem ser percorridas nas consultas. Assim, dada uma coleção de objetos versionáveis S_o e uma coleção de contextos S_c, é possível percorrer todas as versões dos objetos em S_o para cada um dos contextos em S_c. Com a introdução, nos predicados, de referências *ctx*, coleções de contextos podem ser combinados com diversas coleções de objetos, permitindo expressar predicados acerca de relacionamentos de versões de objetos de classes diferentes em contextos diferentes.

Por exemplo, suponha a existência da coleção persistente *TheFarms* da classe **Farm** (classe de objetos versionáveis). A consulta "Selecionar as fazendas construídas no ano 1950 que em alguns dos CET do Banco de Dados tenham "X" como proprietário" pode ser expressada como:

SELECT h
FROM h in TheFarms ,d in Sys-¿GetStContext
WHERE (h.Creation= 1950) and (h[d].Owner.name = "X")

7 Conclusões e extensões

Este artigo discutiu alguns problemas encontrados em SIGs que podem ser resolvidos através do uso de versões nos bancos de dados espaciais subjacentes. As principais contribuições são: a apresentação de um modelo e mecanismo de versões que atenda às necessidades das operações dos usuários de SIG; e a indicação de como o mecanismo pode ser implementado a partir de um mecanismo existente em um SGBDOO.

Há várias extensões possíveis ao trabalho, tanto práticas como teóricas. Do ponto de vista prático, é necessário investigar implementações eficientes do mecanismo. Do ponto de vista teórico, um primeiro trabalho sendo iniciado é o da manutenção de dependências mais gerais entre versões, seguindo o conceito de restrições de integridade. Outra extensão envolve considerar o versionamento de esquema.

Referências

[Bat92] P. Batty. Exploiting relational database technology in a GIS. *Computers and Geosciences: An International Journal*, 18(4):453–462, 1992.

[BCJ92] MJ. Blin, W. Cellary, and G. Jomier. A model of configurations for hardware/software system deliveries. In *Proc. 5th Int. Conf. on Software Engineering and its Applications, Toulouse, France*, pages 338–347, December 1992.

[BH89] A. Bjornerstedt and C. Hulten. Version control in an object-oriented architecture. In W. Kim and K. Lochovsky, editors, *Object-Oriented Concepts, Databases and Applications*, chapter 18, pages 451–485. ACM Press, 1989.

[Bot95] M. Botelho. Incorporação de facilidades espaço-temporais em bancos de dados orientados a objetos. Master's thesis, Universidade Estadual de Campinas, IMECC, DCC, 1995.

[CFS+94] G. Câmara, U. Freitas, R. Souza, M. Casanova, A. Hemerly, and C.B. Medeiros. A model to cultivate objects and manipulate fields. In *Proc. 2nd ACM workshop on advances in GIS, Maryland, USA*, pages 30–37, December 1994.

[CJ90] W. Cellary and G. Jomier. Consistency of versions in object oriented databases. In *Proceedings International VLDB Conference, Brisbane, Australia*, pages 432–441, August 1990.

[CK86] H. Chou and W. Kim. A unifying framework for versions in a CAD environment. In *Proceedings International VLDB Conference*, pages 336–344, August 1986.

[CT95] C. Claramunt and M. Theriault. Managing time in GIS. An event-oriented approach. In *Recent Advances in Temporal Databases*, pages 23–42, September 1995.

[dV97] L. del Val. Tratamento de versões em bancos de dados para sistemas de informações geográficas. Master's thesis, Universidade Estadual de Campinas. Instituto de Computação UNICAMP, March 1997.

[Gan94] S. Gançarski. *Versions et Bases de Donnes: modèle formel, supports de langage et d'interface-utilisateur*. PhD thesis, Paris-Sud University, Centre d'Orsay, France, 1994.

[GJ94] S. Gançarski and G. Jomier. Managing entity versions within their contexts: A formal approach. In *Proc. International DEXA Conference*, volume 856 of *Lecture Notes in Computer Sciences*, pages 400–409, 1994.

[Gol93] L. Golendziner. Um estudo sobre versões em bancos de dados orientados a objetos. Technical report, Universidade Federal de Rio Grande do Sul, 1993.

[HK87] S.E. Hudson and R. King. Object-oriented database support for software environments. *ACM SIGMOD Conference*, pages 491–503, May 1987.

[Kat90] R. Katz. Toward a unified framework for version modelling in engineering databases. *ACM Computing Surveys*, 22(4):375–408, December 1990.

[KBC+89] W. Kim, N. Ballou, H. Chou, J Garza, and D Woelk. Features of the ORION object-oriented database system. In W. Kim and F. Lochovski, editors, *Object-Oriented Concepts, Databases and Applications*, pages 251–282. ACM Press, 1989.

[Lan93] G. Langran. Issues of implementing a spatiotemporal system. *International Journal of Geographical Information Systems*, 7(4):305–314, 1993.

[MJ93a] C. B. Medeiros and G. Jomier. Managing alternatives and data evolution in GIS. *Proc. ACM/ISCA Workshop on Advances in GIS, Arlington, Virginia, USA*, pages 34–37, 1993.

[MJ93b] C. B. Medeiros and G. Jomier. Using versions in GIS. Technical Report 94-05, IMECC - UNICAMP, 1993.

[NM95] H. Naja and N. Mouaddib. The multiple representation in an architectural application. In *Proc. COSIT'95*, volume 988 of *Lectures Notes in Computer Science*, pages 237–246, 1995.

[Peu94] D. Peuquet. It's about time: A conceptual framework for the representation of temporal dynamics in GIS. *Annals of the Association of American Geographers*, September 1994.

[Sci91] E. Sciore. Using annotations to support multiple kinds of versioning in object oriented datasbases. *ACM Transactions on Database Systems*, 16(3):417–438, September 1991.

[Sno90] R.T. Snodgrass. Temporal databases: status and research directions. *ACM SIGMOD Record*, 19(4):83–89, December 1990.

[SS94] S. Sachweh and W. Schafer. Version management for tighly integrated software engineering environments. Technical Report 22, GOODSTEP ESPRIT-III Project, July 1994.

[TOC93] G Talens, C. Oussalah, and M. Colinas. Versions of simple and composite objects. In *Proc. International VLDB Conference*, pages 62–72, 1993.

[VFMa95] E. Victorelli, S. Fortes, and G. Magalhães. Uso de versões na garantia de consistência em ambientes mistos de projeto e operação. Technical report, TELEBRAS/CPqD, 1995.

[WR95] W Wieczerzycki and J. Rykowsky. Version support for CAD CASE. In *Proc. COSIT'95*, volume 988 of *Lecture Notes in Computer Science*, pages 249–260, 1995.