Map-tree: Um Método de Acesso para Mapas Longos

Maurício Riguette Mediano
mediano@inf.puc-rio.br

Marco Antonio Casanova
Centro Científico Rio, IBM Brasil
casanova@vnet.ibm.com

Marcelo Gattass
gattass@inf.puc-rio.br

PUC-RioInf.MCC07/96 March, 1996

Abstract: Access methods for maps play a fundamental role on geographic databases performance that typically stores very large maps. It becomes necessary to treat a map as a long object. This paper introduces a new data structure, called Map-tree, designed to store geometrical and topological information of very large maps. The VR-tree and winged-edge structures are adopted to index the complete geometry description of the map and optimize topological operations, respectively. The paper also defines clustering techniques to optimize the access to secondary storage.

Keywords: spatial access methods, spatial data structures, topological data structures, GIS, spatial databases

Resumo: Métodos de acesso para mapas tem papel fundamental no desempenho de bancos de dados geográficos, onde tipicamente são armazenados mapas muito grandes. Surge então a necessidade de tratar um mapa como um objeto longo. Este trabalho introduz uma nova estrutura de dados, chamada Map-tree, projetada para armazenar informações geométricas e topológicas de mapas muito grandes. As estruturas VR-tree e a winged-edge são adotadas para indexar a descrição geometrica do mapa e otimizar operações topológicas, respectivamente. O trabalho também define técnicas de agrupamento para otimizar o acesso a memória secundária.

Palavras-Chave: métodos de acesso espacial, estruturas de dados espaciais, estruturas de dados topológicos, SIG, bancos de dados espaciais

1 Introdução

Na última década, foram propostos muitos trabalhos na área de métodos de acesso espacial: a Z-order [Ore86] transforma pontos do espaço n-dimensional em pontos no espaço uni-dimensional, para então indexar os mesmos usando B-trees; as R-trees [Gut84, SRF87, BKSS90] e Grid-files [NHS84] indexam pontos e retângulos. Neste caso, cada objeto espacial é aproximado para o seu retângulo envolvente, quando o retângulo envolvente do objeto não é suficiente para resolver uma operação espacial, a geometria completa do objeto é recuperada. Dois métodos de acesso foram propostos para indexar a geometria completa dos objetos espaciais: a TR-tree [SK91, BKSS94] que representa regiões através de um conjunto de trápezios armazenados em uma R-tree, onde já estão representados os buracos da região (tal estrutura não admite edição, pois os trapézios precisariam ser gerados novamente); e a família das V-trees [MCD94a, Med95] que admitem edição e são 3 a 4 vezes mais compactas para representar as mesmas geometrias quando comparadas à TR-tree, pois indexam as linhas de cada objeto espacial, o que as torna mais eficientes. Trabalhos recentes na área de métodos de acesso não consideravam possível otimizar o acesso a descrição completa da geometria de objetos espaciais.

Até agora, nada de concreto foi proposto no sentido de otimizar o acesso a um mapa com todas as suas funcionalidades, levando em consideração questões importantes como: as relações topológicas entre pontos, linhas e regiões; e a manutenção da consistência topológica; do ponto de vista de um mapa como um objeto longo.

São dois os objetivos deste trabalho:

- definir um método de acesso para um mapa longo, chamado Map-tree, utilizando: a V-tree e a VR-tree, estruturas que indexam a descrição completa da geometria dos objetos do mapa; e a winged-edge, que representa de forma eficiente a topologia do mapa, facilitando então a manutenção da consistência topológica do mesmo.

- propor algumas técnicas de agrupamento para a Map-tree, agrupando os objetos do mapa e as partes da descrição geométrica dos mesmos, quando próximos entre si, em unidades de armazenamento em memória secundária.

2 Preliminares

2.1 Definição de Mapas

De acordo com [PD95], um mapa plano é composto de entidades espaciais de três classes: pontos, linhas e regiões no espaço euclideano. Será feita uma distinção entre tais entidades e os objetos espaciais (do ponto de vista semântico) que elas representam. As entidades espaciais que formam um mapa temático tem seus interiores disjuntos, i.e., elas formam uma partição do domínio do mapa.

Um *ponto* é definido unicamente pelas suas coordenadas em um sistema de coordenadas no $\Re^2$. Uma *linha* é definida por uma função contínua $l : I \rightarrow \Re^2$, onde $I = [0,1]$ é a unidade de intervalo na linha. Se $l(0) = l(1)$, a linha l é dita *fechada*, caso contrário é dita *aberta*; os pontos $l(0)$ e $l(1)$ de uma linha aberta l são chamados os *pontos finais* de l, o conjunto formado por uma linha l sem seus pontos finais é chamado de *interior relativo* de l, enquanto que o interior relativo de uma linha fechada é ela própria. Uma linha é chamada *simples* se l é injetora, ou l é fechada e injetora em $[0,1)$. Uma porção $l_{a,b}$ de uma linha l, para um $0 <= a < b <= 1$ é definida por uma restrição de l para o intervalo $[a,b]$, e é ela própria uma linha com uma ecuação paramétrica $l_{a,b}(x) = l(a + x(b-a))$. Pelo teorema de Jordam, uma linha fechada simples separa o $\Re^2$ em dois conjuntos abertos, um conjunto tem fronteiras, é chamado de *conjunto interno*, e denotado por $int(l)$; o outro conjunto não tem fronteiras, é chamado de *conjunto externo*, e denotado por $ext(l)$. Uma *cadeia* é uma sequência de linhas $c = (l_0, ..., l_{k-1})$ tal que $\forall i = 1, ..., k-1, \quad l_{i-1}(1) = l_i(0)$. Abusando da notação, os pontos $l_0(0)$ e $l_{k-1}(1)$ são chamados de pontos finais de c, apenas no caso de $l_0(0) \neq l_{k-1}(1)$; todos os outros pontos pontos finais das linhas que formam c são chamados pontos de junção de c. Uma cadeia c admite parametrização:

$c(x) = l_i(kx - i)$ para $x \in [\frac{i}{k}, \frac{(i+1)}{k}], i = 0, ..., k-1$

Uma cadeia é dita *simples*, *aberta* ou *fechada* se for simples, aberta ou fechada como uma linha, respectivamente.

Uma *região* é caracterizada por uma linha (ou cadeia) fechada simples l_0, chamada de *fronteira externa*, mais um conjunto de linhas (ou cadeias) fechadas simples $l_1, ..., l_k$, chamadas de *fronteiras internas*, tal que: (i) nenhum par destas linhas se interceptam; (ii) para $i = 1, ..., k$ uma linha l_i está contida em $int(l_0)$ e em $ext(l_j)$, onde $0 \neq j \neq i$. A região definida por $l_0, ..., l_k$ é o subconjunto do $\Re^2$ definido como

$\cap_{i=1}^{k} ext(l_i) \cap int(l_0)$,

i.e., é a região interior a fronteira externa, e exterior a todas as fronteiras internas. As fronteiras internas definem *buracos* na região. O número $X(r) = k + 1$ de linhas fechadas que definem a região r é chamada de caracteristica de r. Uma região r sem buracos tem $X(r) = 1$, e é dita *simplesmente conectada*, enquanto que uma região com buracos tem $X(r) > 1$, e é dita *multiplamente conectada*. O *interior* e o *fechamento* de uma região r tem correspondência com o interior e o fechamento padrão de r como um conjunto do $\Re^2$ com a topologia Euclideana. A *fronteira* de r é um subconjunto do $\Re^2$ coberto pelas linhas que definem r, e corresponde a diferença entre o seu fechamento e o seu interior. Uma *região fechada* será referenciada como uma região, caso contrário será feita uma referência explícita a uma *região aberta*.

As seguintes relações entre pares de entidades serão usadas: *disjunto, toca, contém, cobre, igual, sobrepõe, dentro e cobertoPor*.

Mapas podem ser definidos como coberturas disjuntas de uma porção D do plano $\Re^2$, chamada de domínio do mapa, como uma coleção de entidades atômicas. Uma cobertura disjunta é um conjunto de entidades tal que o interior relativo de quaisquer duas regiões nunca se interceptam. Linhas abertas e pontos isolados podem estar completamende dentro ou cobertos por qualquer região, tais linhas e pontos são chamados *detalhes* da região, sendo diferentes das outras linhas e pontos do mapa por não serem parte da fronteira de nenhuma região, e por não serem pontos finais de nenhuma linha, respectivamente. Tal cobertura é chamada de *cobertura disjunta não robusta*.

Um mapa pode ser representado como uma tripla $M = (P, L, R)$, onde os conjuntos P, L e R contém pontos, linhas e regiões de M, respectivamente. Note que M é completamente caracterizado por P e L, pois cada região de R é entendida como uma porção maximal r do plano tal que quaisquer dois pontos interiores a r podem ser conectados por uma linha que não toca nenhuma linha de L.

Todas as possíveis relações entre pares de entidades em um mapa são altamente simplificadas, em relação ao caso genérico. No contexto de um mapa simples, pode-se ter as seguintes situações: (i) duas regiões distintas podem ser *disjuntas*, ou se *tocar* em uma fronteira comum (linha ou ponto); (ii) dada uma linha l e uma região r pode-se ter: l e r *disjuntas*; l e r se *tocando* em um ponto p, onde p é um ponto final de l e p é um ponto final de uma das linhas que *pertence* à fronteira de r; l *pertence* à fronteira de r; l está *dentro* de r; l é *coberta por* r (i.e., o interior relativo de l está dentro de r, e pelo menos um ponto final de l é um ponto final que pertence a uma das linhas que *pertence* ao contorno de r); (iii) dados um ponto p e uma região r pode-se ter o seguinte: p e r são disjuntos; p *pertence à fronteira* de r (i.e., p é ponto final de duas linhas consecutivas da fronteira de r); p está dentro de r; (iv) duas linhas distintas podem ser *disjuntas* ou *tocar* em um ponto final comum a ambas; (v) um ponto p e uma linha l podem ser *disjuntos* ou então *p pertence ao contorno* de (i.e., é um ponto final de) l; (vi) dois pontos distintos são *disjuntos*.

2.2 Preliminares sobre Métodos de Acesso Espacial

V-trees foram inicialmente propostas em [MCD94b, MCD94a], como uma família de estruturas de dados com diversas aplicações nas áreas de bancos de dados geográficos e computação gráfica.

Dados dois inteiros $m > 1$ e $n > 1$, uma *V-tree* de ordem (m, n) é uma árvore m-área tal que: todas as folhas estão no mesmo nível; cada folha N contém uma seqüência de pontos de tamanho entre $n/2$ e n; o retângulo envolvente de N é o retângulo envolvente da seqüência de pontos que N armazena; cada nó interior tem entre $m/2$ e m filhos, exceto a raiz, que tem entre 2 e m filhos; para cada filho M de um nó interior N, N contém uma lista de pares da forma (p, r), onde p aponta para M e r é o retângulo envolvente de M; o retângulo envolvente de N é o retângulo envolvente que cobre todos os retângulos envolventes das entradas em N.

Dada uma linha P descrita por uma sequência finita de pontos, pode-se armazenar P em uma V-tree V de ordem (m, n) quebrando P em fragmentos consecutivos de tamanho n e armazenando cada fragmento em uma folha de uma V-tree. No processo de quebrar uma linha em fragmentos consecutivos, o último ponto de um fragmento é também o primeiro ponto do fragmento seguinte tal que os retângulos envolventes dos fragmentos cobrem completamente a linha. Naturalmente, todos os algoritmos que reconstroem P a partir de V precisam estar atentos para o fato de o último ponto de um fragmento ser o primeiro ponto do próximo fragmento (exceto se o fragmento é o último).

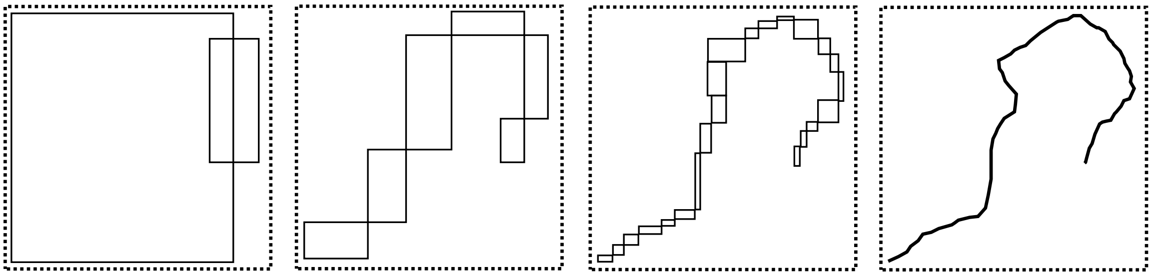

Figura 1: Vários níveis de uma V-tree.

Neste ponto pode-se discutir algumas considerações práticas. Primeiro, os nós interiores e folhas não precisam ocupar páginas de mesmo tamanho, pois elas armazenam entradas de naturezas diferentes. Em outras palavras, os parâmetros m e n a princípio não têm relação. Segundo, pode-se manter as folhas encadeadas e agrupadas em memória secundária para facilitar o acesso a fragmentos consecutivos de uma linha. Se a linha é fechada, as folhas podem ser organizadas como uma lista circular. A Figura 1 mostra a visão de vários níveis de uma V-tree.

Há pelo menos duas maneiras de representar a fronteira externa de regiões: uma delas é através de uma linha e outra é através de uma cadeia (sequência de linhas). A forma mais simples é através de uma linha. Porém, é muito comum a utilização de regiões como parte de mapas. Nesta segunda forma, linhas representam informação geométrica, enquanto que cadeias, representam informação topológica (as sequências de linhas adjacentes que formam a fronteira externa das regiões). Tipicamente, linhas pertencentes a partições disjuntas de mapas são compartilhadas por duas regiões. Ao representar a fronteira externa de regiões como linhas, é necessário que uma mesma seqüência de pontos relativa a fronteira externa de duas regiões seja repetida na fronteira externa de cada região. Ao representar regiões como cadeias, evita-se esta redundância, pois para representar uma linha comum a duas regiões, basta que as duas cadeias que representam a fronteira externa das duas regiões possuam uma referência para a linha comum.

Para armazenar a fronteira externa de uma região como uma seqüência de linhas, define-se uma estrutura chamada *PV-tree*, uma variante da V-tree que, em vez de armazenar seqüências de pontos, armazena seqüências de linhas. Na raiz da PV-tree há uma referência para um conjunto de PV-trees que representam as regiões internas à região em questão, quando a referência é nula significa que não há regiões internas. A estrutura que armazena conjuntos de PV-trees é chamada de PVR-tree e será vista mais a frente.

Pode ser útil armazenar *a priori*, nas entradas das folhas da PV-tree, uma indicação para mostrar se cada linha referente a uma entrada se encontra no mesmo sentido ou no sentido contrário da seqüência de linhas que compõe a fronteira externa da região. Ao armazenar esta informação, pode-se percorrer a PV-tree até as seqüências de pontos armazenadas nas folhas das linhas, sem testar o sentido das mesmas.

Discutiu-se como armazenar uma linha (longa), esteja ela aberta ou fechada. Agora será discutido como armazenar um conjunto de linhas (longas).

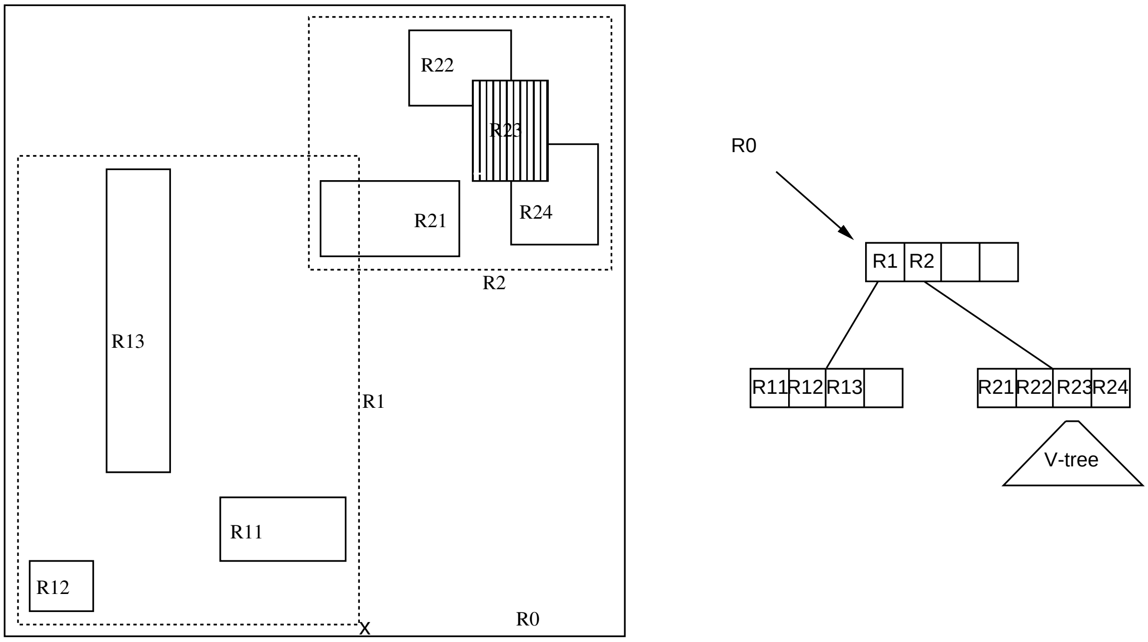

Figura 2: Um exemplo de VR-tree.

Seja Π um conjunto de linhas que segue.

A abordagem sugerida é armazenar cada linha P de Π em uma V-tree V separada, aproximando P por um retângulo envolvente B_P, inserindo uma entrada composta por B_P e uma referência a V em qualquer um dos vários métodos projetados para acessar retângulos. Quando o método de acesso a retângulos escolhido é uma R-tree, chama-se a estrutura final de uma *VR-tree*.

A Figura 2 ilustra uma VR-tree. O retângulo $R0$ representa a raiz da R-tree que possui duas folhas: $R1$ e $R2$. A folha $R1$ contém três retângulos envolventes e três referências às raizes de três objetos armazenados em V-trees: $R11$, $R12$ e $R13$. A V-tree relativa ao objeto $R23$, contido na folha $R2$, está representada na figura como uma subárvore, uma continuação da R-tree.

Para cada VR-tree, haverá sempre um certo nível l tal que: ao desprezar-se todos os nós abaixo do nível l, a estrutura resultante será uma R-tree, chama-se esta estrutura resultante de *componente R-tree* da VR-tree; cada subárvore referenciada através dos nós no nível l, é uma V-tree, chama-se esta estrutura resultante de *componente V-tree* da VR-tree.

Note-se que, diferente de outros métodos de acesso que tentam minimizar a sobreposição de retângulos envolventes, VR-trees armazenam todos os fragmentos das linhas nas folhas dos seus componentes V-tree.

Note-se ainda que uma VR-tree não é uma árvore de busca balanceada, pois as folhas não estão necessariamente no mesmo nível, as folhas pertencem a V-trees distintas que são subárvores da VR-tree. Sendo assim, uma VR-tree não é uma V-tree, pois armazena pontos pertencentes a linhas distintas.

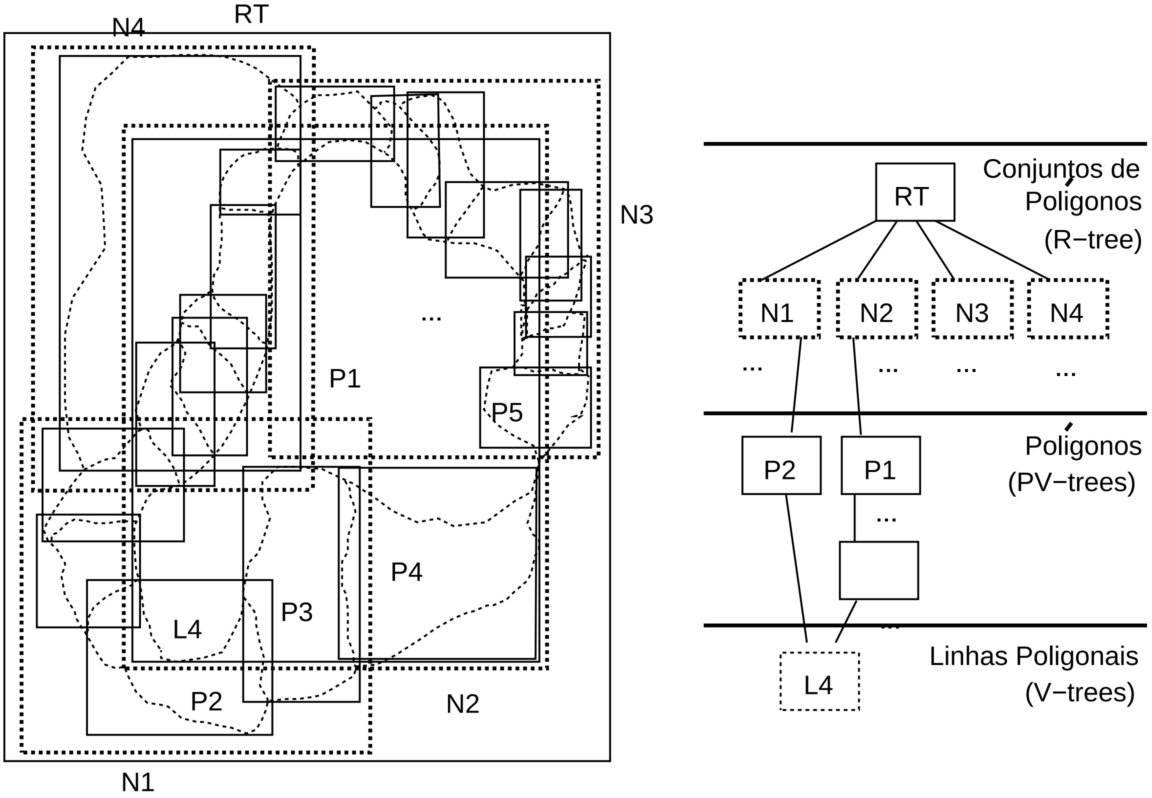

Figura 3: Um exemplo de PVR-tree.

A *PVR-tree* é uma variante da VR-tree, onde a única diferença entre as duas estruturas é a possibilidade da PVR-tree armazenar regiões que compartilham uma mesma linha. Em vez de armazenar V-trees contendo seqüências de pontos, a PVR-tree armazena PV-trees que armazenam seqüências de linhas. Na Figura 3 pode-se ver que a linha $L4$ é alcançada tanto através da região $P1$ da VR-tree quanto através da região $P2$. Qualquer alteração em $L4$ feita através de $P1$ é automaticamente refletida em $P2$, assim como quaquer alteração feita por $P2$ em $L4$ é automaticamente refletida em $P1$.

Uma região com buracos r pode ser representada por uma PV-tree que contém a fronteira externa de r e uma PVR-tree contendo um conjundo de regiões internas a r, onde a fronteira externa destas regiões correspondem às fronteiras internas de r. Nos termos da seção 2.1, a PVR-tree é capaz de armazenar uma partição disjunta não-robusta de um mapa.

3 Armazenamento de Mapas

O estrutura de armazenamento proposta para mapas, chamada *Map-tree*, armazena as partições disjuntas do domínio de um mapa, bem como os detalhes (pontos e linhas) de cada partição. Outra característica da estrutura é armazenar informações relacionadas com a adjacência entre as entidades do mapa. O armazenamento destas informações, otimiza operações comuns a mapas e suas entidades, tais como: operações sobre regiões, operações de edição de mapas e operações relacionadas com a manutenção da consistência topológica do mapa. Tais informações são utilizadas em estruturas topológicas conhecidas como *winged-edge* e *half-edge* [PS85]. A Map-tree também otimiza operações de consulta sobre os conjuntos das entidades do mapa, com restrições quanto a geometria e as classes das entidades.

Seja um mapa M, armazena-se em uma R-tree, chamada T_M, todas as linhas que definem as fronteiras externas das regiões de M, todas as linhas que são detalhes de M e todos os pontos que são detalhes de M. Cada linha l de M é uma entrada em T_M na forma (v, b, c, q) onde: v referencia a V-tree que armazena l, b é o retângulo envolvente de l, c é o identificador da classe a qual l pertence e q é uma referência para um registro na parte convencional do SGBD onde estão informações adicionais sobre l. Quando a referência q é nula significa que l é uma entidade fraca, não possuindo informações adicionais. Neste caso a única informação disponível a respeito de l é a classe a qual l pertence. Cada ponto p de M é uma entrada em T_M na forma $((x, y), c, q)$ onde: (x, y) são as coordenadas de p, c é o identificador da classe de p e q é uma referência para um registro na parte convencional do SGBD onde estão informações adicionais sobre p. Quando a referência q é nula significa que p é uma entidade fraca, não possuindo informações adicionais. Neste caso a única informação disponível a respeito de p é a classe a qual p pertence. Obviamente, os registros na parte convencional do SGBD também possuem referências para as geometrias de seus objetos que são entidades do M, pois consultas sem restrições geométricas, são resolvidas na parte convencional do SGBD. Os pontos e linhas de M armazenados em T_M são suficientes para caracterizar M completamente, pois é possível a partir da geometria das linhas descobrir: as regiões que particionam o domínio de M, quais pontos são detalhes de M e quais linhas são detalhes de M.

Dada uma linha l de M que pertence à fronteira de duas regiões $r1$ e $r2$, e dada uma V-tree v que armazena l, são armazenadas junto com a raiz de v as referências para as raízes das duas V-trees que armazenam as duas linhas adjacentes a l na fronteira externa de $r1$ e as referências para as raízes das duas V-trees que armazenam as duas linhas adjacentes a l na fronteira externa de $r2$. São armazenadas também, junto com a raiz de v, duas referências para as PV-trees que representam a fronteira externa de $r1$ e $r2$, as duas regiões vizinhas a l.

As informações relativas à adjacência entre as linhas que particionam o domínio de M não são suficientes para tratar todas as relações de adjacência entre as entidades de M. Além de armazenar as linhas que compõem as partições do domínio de M e as linhas e pontos que são detalhes de M, há dois outros grupos de entidades, relacionados com a topologia de M, que devem ser armazenados: os pontos finais das linhas que formam as partições de M, chamados *vértices* de M, e as próprias partições do domínio de M, as regiões de M. Como estas entidades são geradas a partir de T_M, elas são armazenadas em duas estruturas separadas, como será visto a seguir.

Os vértices são armazenados em uma R-tree, chamada T_V, que pode ser gerada a partir de T_M, onde cada vértice p de M é uma entrada em T_V na forma $((x, y), v)$ onde: (x, y) são as coordenadas de p e v é uma referência para uma V-tree que armazena qualquer uma das linhas que têm p como ponto final. Apenas um vértice é armazenado para dois ou mais pontos finais idênticos. Não faz muito sentido que o conjunto dos vértices de M possua classes associadas a seus elementos, uma vez que estes vértices são gerados apenas para otimizar operações de adjacência entre os mesmos e as linhas de M, não possuindo qualquer significado para o usuário. Pelo mesmo motivo, não faz sentido associar a p uma referência para um registro na parte convencional do SGBD.

O acesso ao conjunto de regiões que particiona o domínio de M é otimizado mantendo-se uma R-tree, chamada T_R, que pode ser gerada a partir de T_M, onde cada região r de M é uma entrada em T_R na forma (b, e, c, q) onde: b é o retângulo envolvente de r, e é uma referência para a PV-tree que armazena r, c é o identificador da classe a qual r pertence e q é uma referência para um registro na parte convencional do SGBD onde estão informações adicionais sobre r. Quando a referência q é nula, significa que r é uma entidade fraca, não possuindo informações adicionais, apenas a classe a qual pertence.

A informação sobre a classe c associada a uma região, linha ou ponto de M não deve residir unicamente na parte convencional do SGBD, pois toda a entidade do mapa teria que possuir um registro na parte convencional do SGBD, o que é desnecessário. Consultas a conjuntos de entidades de uma determinada classe ou grupo de classes, com restrições geométricas, podem ser executadas percorrendo T_R ou T_M sem que seja necessário recorrer a parte convencional do SGBD para descobrir a que classe pertencem os objetos que satisfazem as restrições geométricas. Para otimizar ainda mais este tipo de consulta, armazena-se em cada referência a uma subárvore ou folha das R-trees T_M e T_R uma *máscara de classes* informando quais classes possuem objetos na subárvore ou folha em questão. Eventualmente, diversas subárvores de T_M e T_R podem ser desprezadas quando as máscaras negam a existência de objetos de uma classe ou grupo de classes de interesse durante a consulta.

A figura 4 mostra um exemplo da Map-tree. No nível A estão armazenadas as R-trees T_M, T_V e T_R de M. No nível B, estão as PV-trees de T_R que indexam as regiões de M. No nível C estão as raizes das V-trees e as informações de adjacência entre as linhas. Finalmente, no nível D estão os níveis das V-trees abaixo do primeiro nível, das raizes. Na estrutura topológica winged-edge, as árvores T_M, T_V e T_R são implementadas como listas e não é definida uma conexão com a parte convencional do SGBD.

Em uma Map-tree completa (com o nível B), a raiz da V-tree que armazena uma linha pode ser referenciada até nove vezes. A R-tree T_M referencia cada linha uma vez. Cada vértice em T_V referencia uma linha adjacente a ele. No nível B, cada linha é referenciada duas vezes, pelas duas PV-trees que armazenam as duas regiões adjacentes a linha. Cada linha é referenciada também pelas quatro linhas adjacentes. As linhas, regiões e pontos de M podem referenciar registros na parte convencional do SGBD e vice-versa. Na winged-edge, cada região referencia apenas uma das linhas de sua fronteira externa, para ter acesso a toda a geometria da região, é preciso percorrer as listas de adjacência até encontrar a primeira referência usada.

A quantidade de informação contida na estrutura aumenta a cada nível. As R-trees T_M, T_R e T_V no nível A possuem apenas referências para as informações geométricas e topológicas das entidades de M armazenadas nos níveis B e C. Os retângulos envolventes e as máscaras de classes nos nós intermediários das subárvores nas R-trees funcionam como chaves de busca, cada retângulo envolvente encontrado nas folhas é uma aproximação da geometria de uma linha, enquanto pontos e vértices tem suas coordenadas armazenadas nas folhas de T_M e T_V, respectivamente. Sendo assim, T_V, T_M e T_R otimizam as operações com restrições

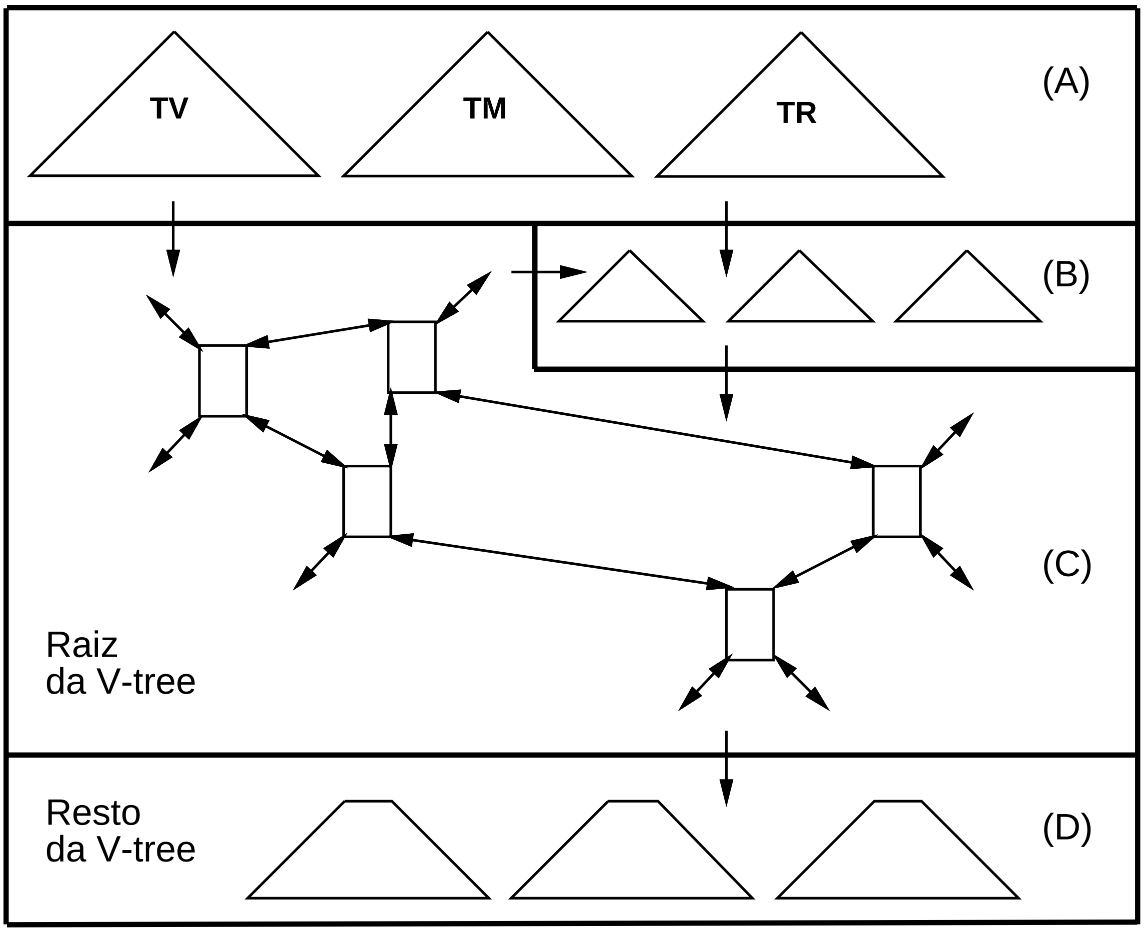

Figura 4: Um exemplo da Map-tree.

geométricas e com restrições por classes, sobre conjuntos de vértices, linhas (e pontos) e regiões, respectivamente. As PV-trees no nível B contém as sequências de referências e retângulos envolventes das linhas que formam as fronteiras externas das regiões de M, o que otimiza as operações geométricas sobre as regiões. Cada raiz de uma V-tree no nível C contém alguma informação geométrica sobre a linha armazenada (os retângulos envolventes das subárvores, ou a própria sequência de pontos quando a mesma cabe em apenas um nó) e as quatro referências para as raízes das quatro V-trees que armazenam as quatro linhas adjacentes, otimizando operações geométricas e de adjacência entre linhas, respectivamente. Finalmente, os nós das V-trees no nível D, contém o resto da informação sobre a geometria das linhas, o que permite otimizar operações sobre a geometria das mesmas.

Quanto ao tamanho da estrutura, supondo que todas as V-trees possuem pelo menos um nível além da raiz, a menor quantidade de informação capaz de descrever completamente as partições do domínio do mapa, as sequências de pontos que formam as linhas, estão no último nível de cada V-tree, nas folhas no nível D. Os níveis acima das folhas possuem sempre uma fração do tamanho do nível das folhas, pois cada referência em um nível superior referencia grupos de elementos e referências no nível inferior. Esta cararterística se aplica a todos os níveis das subárvores, do nível D ao nível A. Em compensação, as raizes das V-trees são referenciadas mais de uma vez. Logo, quanto maior é a abertura das árvores, menor é a diferença entre o tamanho total da estrutura e o tamanho original dos dados.

As operações de edição de mapas são semelhantes as operações de edição de mapas da estrutura winged-edge, a única diferença é que em vez de percorrer listas para procurar os vértices, linhas e faces próximas a um ponto ou que interceptam um ponto, as operações de consulta a ponto são otimizadas percorrendo as R-trees T_V, T_M e T_R. As operações de inserção, remoção e consulta em R-trees estão bem definidas em [Gut84, RL85, SRF87, BKSS90, HNP95]. A questão das V-trees e das PV-trees é ortogonal a esta discussão, pois a winged-edge trata apenas da topologia de mapas, deixando em aberto como são implementadas as operações sobre a geometria das linhas. O armazenamento e consequente otimização do acesso a geometria de linhas e regiões usando V-trees e PV-trees são abordadas em [Med95].

Seja a consulta Q sobre M que retorna um mapa M', tal que:

- C_p, C_l e C_r são subconjuntos de classes as quais pertencem os pontos, linhas e regiões de M, respectivamente;
- γ_p, γ_l e γ_r são restrições geométricas sobre pontos, linhas e regiões, respectivamente.

O novo mapa M', resposta da consulta Q, contém pontos, linhas e regiões de M que satisfazem as restrições geométricas γ_p, (γ_l ou γ_r) e γ_r, e pertencem as classes contidas em C_p, C_l e C_r, respectivamente. Quando uma região é selecionada por γ_r, todas as suas linhas são selecionadas, sendo assim, uma linha pode ser selecionada por γ_l ou γ_r.

Dada uma Map-tree Map_M que armazena M e possui as R-trees T_V, T_M e T_R, e dada uma Map-tree $Map_{M'}$ que armazena M' e possui as R-trees T'_V, T'_M e T'_R, a consulta Q deve seguir os seguintes passos para gerar M':

- Percorrer T_M procurando os pontos e linhas cujas classes pertencem aos conjuntos de classes C_p e C_l, e que satisfazem as restrições geométricas γ_p e γ_l, respectivamente. Inserir os pontos e linhas em T'_M.
- Percorrer T_R procurando as regiões cujas classes pertencem ao conjunto de classes C_r e que satisfazem a restrição geométrica γ_r. Inserir as linhas das fronteiras externa e internas de cada região selecionada em T'_M. Só são inseridas as linhas que não foram inseridas em T'_M.
- Quando uma cadeia fechada é gerada pela inserção de uma linha, uma PV-tree representando a fronteira externa de uma região é criada e inserida em T'_R, é gerada também a R-tree que contém as referências para as fronteiras internas da região em questão.
- Para cada linha de T'_M, inserir os dois pontos finais como vértices em T'_V. Só são inseridos os vértices que não foram inseridos em T'_V.
- As informações de adjacência entre linhas e linhas de M', ou entre linhas e regiões, podem ser corrigidas após cada inserção, procurando as linhas ou regiões adjacentes, e modificando as informações de adjacência entre as mesmas.

4 Métodos de Agrupamento

Quando o tempo de acesso em memória secundária é ordens de grandeza mais lento quando comparado a velocidade do processador, torna-se importante tentar agrupar objetos e seus fragmentos de uma forma coerente com o acesso feito aos mesmos durante consultas ou atualizações. Na Map-tree, o fato de utilizar árvores balanceadas diminui o número de acessos, o que torna este tipo de estrutura adequado também para meios de acesso lento. O aumento do número de elementos por nó diminui a altura da árvore, provocando a diminuição do número de acessos aos nós. Em contra-partida, quanto maior é o número de elementos por nó, maior é a perda na otimização proporcionada por estas estruturas, pois maior é o esforço do processador face a um número maior de elementos por nó. Torna-se adequado agrupar vários nós de árvores em unidades de armazenamento, mantidas em memória principal através de sistemas de buffers. Quando bem suscedida, esta estratégia reduz o número de acessos, sem contudo aumentar o número de elementos por nó.

Esta seção discute estratégias de agrupamento dos nós das árvores balanceadas da Map-tree em unidades de armazenamento. As estratégias de agrupamento geométrico e por adjacência são usadas para otimizar o acesso a Map-trees em memória secundária. Em particular, será discutido um terceiro método, o de agrupamento hierárquico, para otimizar o acesso feito por grupos específicos de consultas.

4.1 Agrupamento Geométrico

O método de agrupamento geométrico consiste em agrupar objetos próximos quanto a distância geométrica dos mesmos entre si. O próprio método de divisão de nó da R-tree

[Gut84, RL85, SRF87, BKSS90] se constitui em uma estratégia eficiente de agrupamento geométrico. Quando um nó e dividido, os elementos do mesmo são divididos em dois grupos distintos de elementos próximos entre si. Sendo assim, quando ocorre uma divisão, pode-se usar o próprio resultado da divisão para agrupar os elementos referenciados pelos dois novos nós em duas unidades de armazenamento distintas. É adequado agrupar segundo esta estratégia: os nós dos níveis abaixo do nível das raizes das R-trees, no nível A; as raizes das PV-trees no nível B e as raizes das V-trees no nível C, pois os nós pai desses nós são divididos segundo o critério de divisão da R-tree.

O método proposto em [BK94] não define um método de acesso para a geometria completa dos objetos, a geometria é toda inserida em unidades de armazenamento ("cluster units"). Cada unidade armazena a geomentria completa dos objetos referenciados por uma folha de R-tree. Como o tamanho dos objetos é arbitrário, a divisão da folha acontece quando a unidade de armazenamento não suporta o novo objeto a ser inserido ou quando a folha alcança o número máximo de elementos. Quando a geometria dos objetos é armazenada em métodos de acesso como as V-trees e PV-trees, deixa de ser necessário condicionar o momento da divisão do nó ao tamanho dos objetos armazenamdos nas unidades, pois os nós destas árvores tem tamanho fixo. Nas Map-trees, o tamanho máximo das unidades de armazenamento é igual ao número máximo de elementos no nó pai vezes o tamanho dos nós armazenados. O momento da divisão da unidade de armazenamento não influencia a divisão do nó pai e é feita da mesma forma, no mesmo momento.

4.2 Agrupamento por Adjacência

O método de agrupamento por adjacência é uma opção para agrupar as raizes das V-trees e PV-trees, nos níveis B e C, respectivamente. Basta trocar o método de divisão de nó por proximidade geométrica das folhas das R-trees por um método de divisão por proximidade topológica. Excetuando-se as folhas, os demais nós das R-trees continuariam utilizando o mesmo método de agrupamento por proximidade geométrica (não faz sentido adjacência entre subárvores de R-trees), muito embora a divisão das folhas influencie na forma geométrica dos nós.

O algoritmo exposto na Figura 5 consiste em tentar dividir cada unidade de armamazenamento em dois grupos de entidades adjacentes entre si. Este algoritmo é bastante eficiente, dado que todas as informações de adjacência já estão armazenadas. De uma forma geral, o algoritmo de divisão da R-tree e o algoritmo SplitADJ tendem a ter resultados parecidos, uma vez que objetos próximos topologicamente tendem a ser próximos geometricamente. A estratégia de agrupar objetos topologicamente tem maior sucesso durante consultas e atualizações a grupos objetos adjacentes entre si. Um exemplo seria a edição de mapas. Agrupar objetos geometricamente é adequado para consultas como: retornar todos objetos que interceptam uma determinada região.

Os nós das V-trees e PV-trees, que não são raizes, são agrupados em unidades de armazenamento pelo método de divisão de seus nós superiores. Neste caso, como a divisão é feita respeitando a sequência dos pontos e linhas armazenados nestas estruturas, pode-se considerar que também é utilizado um método de agrupamento por adjacência.

algorithm *SplitADJ*
  input: *nó antigo*
  output: *nó antigo, nó novo*
begin
  remove do *nó antigo* os dois elementos mais distantes topologicamente
  insere um elemento na *fila1*
  insere outro elemento na *fila2*
  while há elementos no *nó antigo* **do**
    if algum *elemento na fila1* tem adjacente no *nó antigo*
      remove *adjacente* do *nó antigo* e insere na *fila1*
    else
      remove *próximo geométrico* do *nó antigo* e insere na *fila1*
    if algum *elemento na fila2* tem adjacente no *nó antigo*
      remove *adjacente* do *nó antigo* e insere na *fila2*
    else
      remove *próximo geométrico* do *nó antigo* e insere na *fila2*
  insere elementos da *fila1* no *nó antigo*
  insere elementos da *fila2* no *nó novo*
end *SplitADJ*

Figura 5: O algoritmo SplitADJ.

4.3 Agrupamento Hierárquico

Os métodos de agrupamento geométrico e por adjacência dispõem cada nó, de um caminho entre a raiz e as folhas, em uma unidade de armazenamento diferente, como se cada árvore fosse armazenada ao ser percorrida em sua largura. Esta estratégia é adequada para V-trees e PV-trees, pois cada nível de uma dessas árvores é uma aproximação da geometria da entidade nela armazenada, por este motivo é comum que os algoritmos que trabalhem com margens de erro nestas duas estruturas tenham um melhor desempenho acessando os nós agrupados desta forma. De uma forma geral, o objetivo destas árvores balanceadas, inclusive da R-tree, é prover seletividade para evitar o acesso aos níveis próximos as folhas, pois quanto mais próximo das folhas é um determinado nível de uma árvore balanceada, maior a quantidade de informação a ser acessada. Porém, há situações onde é comum percorrer o caminho da raiz até as folhas, por exemplo quando se deseja conhecer a área do mapa coberta por uma região sem tolerar aproximações ou quando se deseja recuperar a geometria completa de um objeto espacial. O método de agrupamento hierárquico tenta agrupar subárvores inteiras em unidades de armazenamento. Em ambos os casos as tarefas serão executadas de forma mais eficiente quando cada grupo de nós dispostos hierarquicamente, pertencentes aos vários níveis das árvores, for armazenado em uma unidade de armazenamento. Para tanto basta percorrer cada árvore, guardando os nós em unidades de armazenamento em pré-ordem.

5 Conclusão

O método de acesso para mapas proposto, chamado Map-tree, otimiza as operações com restrições geométricas e por classes sobre os conjuntos dos pontos, linhas e regiões de um mapa e otimiza também operações geométricas e topológicas entre os mesmos pontos, linhas e regiões.

Os métodos de agrupamento geométrico, por adjacência e hierárquico propostos, tendem a otimizar o desempenho da Map-tree quando utilizados em meios de acesso e procura lentos.

6 Agradecimentos

Este trabalho foi parcialmente financiado pelo projeto PROTEM/GEOTEC. O primeiro autor gostaria de agradecer também a CAPES pelo financiamento de sua bolsa de estudos.

Referências Bibliográficas

[BK94] Thomas Brinkhoff and Hans-Peter Kriegel. The Impact of Global Clustering on Spatial Database Systems. In *Proceedings of the 20th VLDB Conference*, September 1994.

[BKSS90] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Retangles. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 322–332, May 1990.

[BKSS94] Thomas Brinkhoff, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. Multi-Step Processing of Spatial Joins. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 197–208, May 1994.

[Gut84] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the ACM SIGMOD Conference on Data Engineering*, pages 47–56, June 1984.

[HNP95] Joseph M. Hellerstein, Jeffrey F. Naughton, and Avi Pfeffer. Generalized Search Trees for Database Systems. Technical report, Submitted to VLDB, 1995.

[MCD94a] Maurício Riguette Mediano, Marco Antônio Casanova, and Marcelo Dreux. A Family of Storage Methods for Geographic Data. In *9º Simpósio Brasileiro de Bancos de Dados*, Setembro 1994.

[MCD94b] Maurício Riguette Mediano, Marco Antônio Casanova, and Marcelo Dreux. V-tree - A Storage Method for Long Vector Data. In *Proceedings of the 20th VLDB Conference*, September 1994.

[Med95] Maurício Riguette Mediano. *V-trees: Um Método de Armazenamento para Dados Vetoriais Longos*. PhD thesis, Pontifícia Universidade Católica, Rio de Janeiro, Fevereiro 1995.

[NHS84] J. Nievergelt, H. Hinterberguer, and K. C. Sevick. The Grid File: An Adaptable, Symmetric Multikey File Structure. *ACM Transactions on Database Systems*, 9(1):38–71, March 1984.

[Ore86] J.A. Orenstein. Spatial Query Processing in an Object-Oriented Database System. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 326–336, May 1986.

[PD95] Enrico Puppo and Giuliana Dettori. Towards a formal model for multiresolution spatial maps. In *Proceedings SSD*, 1995.

[PS85] F. P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[RL85] Nick Roussopoulos and Daniel Leifker. Direct Spatial Search on Pictorial Databases Using Packed R-trees. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 17–31, May 1985.

[SK91] Ralf Schneider and Hans-Peter Kriegel. The TR*-tree: A New Representation of Polygonal Objects Supporting Spatial Queries and Operations. In *Proceedings of 7th Workshop on Computational Geometry*, pages 507–518, 1991.

[SRF87] Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The R^+-Tree: A Dynamic Index for Multi-Dimensional Objects. In *Proceedings of the 13th VLDB Conference*, pages 507–518, September 1987.