

7 Processamento de consultas e gerência de transações

Marco Antonio Casanova

7.1 Introdução

Este capítulo aborda dois aspectos da implementação de um sistema de gerência de bancos de dados geográficos: processamento de consultas espaciais e gerência de transações. O texto enfatiza as questões que necessitam soluções distintas daquelas empregadas em sistemas convencionais.

Um dos principais componentes de todo sistema de gerência de bancos de dados, o processador de consultas, recebe uma consulta e prepara um plano de execução, consistindo de operações de mais baixo nível, implementadas pelo subsistema de armazenamento. A parte do plano responsável pelo processamento dos objetos geográficos tipicamente começa com uma fase de filtragem, que identifica quais objetos podem satisfazer a qualificação da consulta. Esta fase pode utilizar índices espaciais, em geral definidos sobre aproximações das geometrias dos objetos. A fase de refinamento do plano recupera para memória principal a geometria exata dos objetos identificados na fase anterior, computando quais deles de fato satisfazem a qualificação da consulta. A fase final do plano aplica transformações aos objetos retornados na fase anterior, produzindo a resposta final da consulta.

Um segundo componente importante de todo sistema de gerência de banco de dados implementa a noção de transação atômica, tipicamente de curta duração. No caso de sistemas de informação geográfica, a interação com o banco de dados é comumente mais longa exigindo uma revisão dos mecanismos de controle de transações, principalmente a criação de mecanismos para versionamento, atualização cooperativa e bloqueio parcial de objetos.

Referências para problemas específicos sobre processamento de consultas e gerência de transações encontram-se ao longo do texto e sugestões para leitura adicional, ao final do capítulo.

7.2 Estratégia para processamento centralizado de consultas

Um dos principais componentes de todo sistema de gerência de bancos de dados é o processador de consultas que, ao receber uma consulta Q , prepara um *plano de execução* para Q consistindo de operações de mais baixo nível, implementadas pelo subsistema de armazenamento.

O principal componente do processador de consultas, por sua vez, é um otimizador capaz de gerar planos alternativos para execução de consultas, estimar o custo de cada plano e escolher o de menor custo estimado. O otimizador incorpora heurísticas que limitam o espaço de busca, evitando uma explosão combinatorial de planos alternativos. Seguindo (Brinkhoff et al., 1993), a parte do plano responsável pelo processamento da componente espacial de uma consulta contém três fases (ver Figura 7.1):

Fase 1: Filtragem

Um banco de dados geográfico normalmente contém estruturas de dados auxiliares, chamadas de *índices espaciais*, *estruturas de indexação espacial* ou simplesmente *índices*, que facilitam a localização dos dados cuja componente espacial satisfaz à qualificação de uma consulta. Reservamos o termo *método de indexação espacial* para designar uma família de estruturas de indexação espacial. De fato, diversos métodos de indexação espacial foram discutidos no Capítulo 6.

O desempenho deste passo depende da distribuição espacial dos dados e da forma como a estrutura decompõe o espaço, sendo que em geral é baixo o fator de seletividade. Dada a complexidade da componente espacial e da semântica dos próprios operadores espaciais, em geral os índices trabalham apenas com aproximações das componentes espaciais (por exemplo, o retângulo mínimo envolvente). Assim, o uso de índices para resolver a parte espacial de uma consulta apenas filtra, dos conjuntos de dados armazenados no banco de dados, aqueles que certamente não satisfazem à qualificação da consulta, sendo necessário um segundo passo.

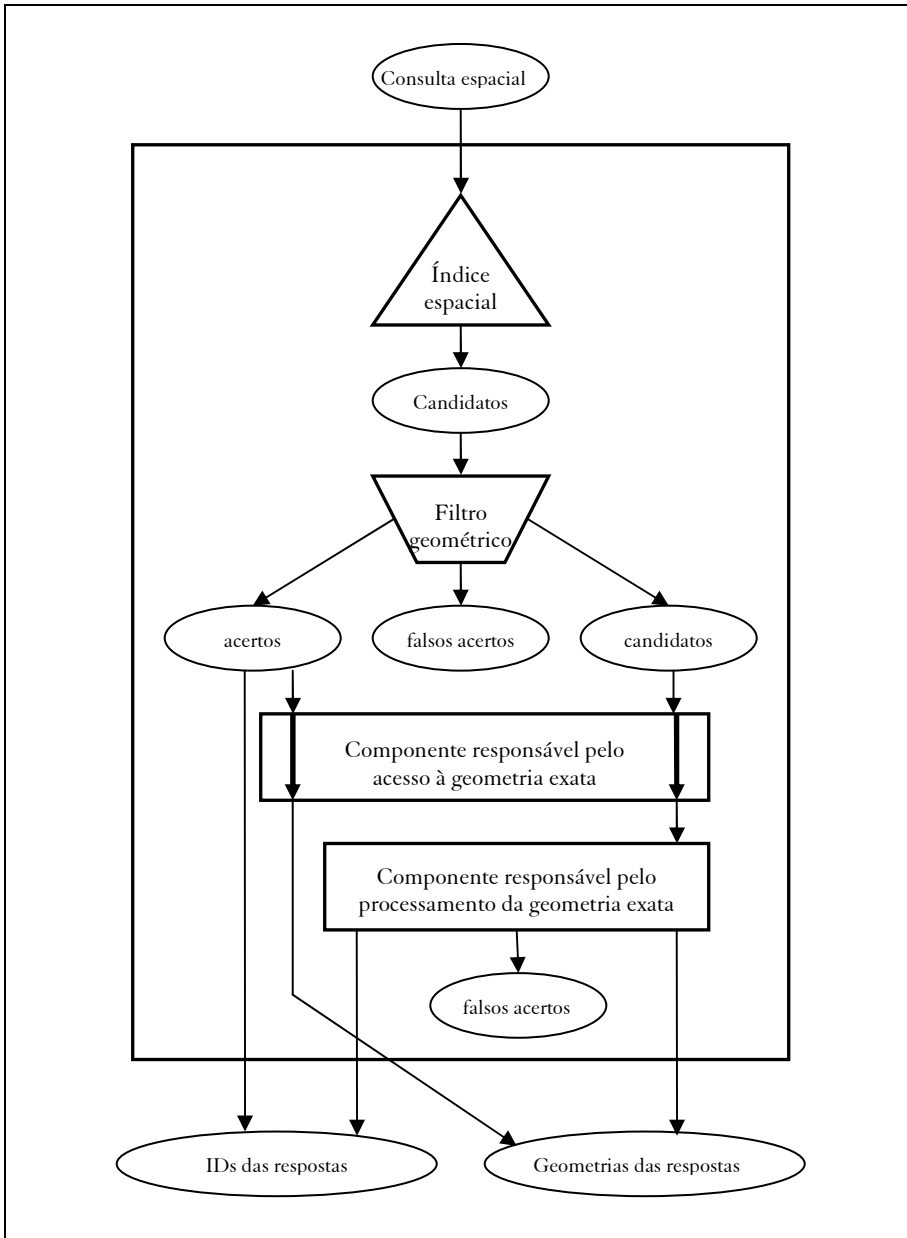


Figura 7.1 – Processamento de consultas espaciais (Kriegel et al., 1993).

Fase 2: Refinamento

Uma vez identificados os dados que se candidatam a satisfazer a qualificação da consulta, as geometrias exatas das componentes espaciais dos dados são trazidas para memória principal. É sobre estas geometrias que são então executados os operadores espaciais especificados na qualificação. As estruturas de armazenamento e indexação devem preferencialmente permitir que seja trazida para memória principal apenas a parte das componentes espaciais necessária à execução dos operadores.

O desempenho deste passo depende da quantidade de dados recuperados, da sua complexidade e do custo dos operadores a aplicar, que em geral é elevado.

Fase 3: Pós-Processamento

Depois de identificados os dados que satisfazem à qualificação da consulta, usualmente é necessário passá-los para as camadas superiores da arquitetura onde sofrerão processamento posterior ou serão visualizados. Assim, o sistema deve oferecer um mecanismo eficiente para passar a geometria das componentes espaciais dos dados, na sua totalidade ou em parte, para as camadas superiores.

Cabe ao subsistema de armazenamento implementar os *métodos de armazenamento*, que governam a utilização das páginas físicas em memória secundária, onde residem tanto as componentes espaciais dos dados quanto os próprios índices espaciais.

Este subsistema deve equacionar dois problemas:

espaço de armazenamento: a representação interna de uma componente espacial pode necessitar um espaço substancial, ocupando mais de uma página física;

contigüidade física: as páginas físicas de uma mesma componente espacial ou de componentes contíguas no espaço devem ser vizinhas em memória secundária, diminuindo o custo de acesso.

Uma estratégia para resolver estes problemas consiste em armazenar a geometria exata das componentes espaciais dos dados fora dos índices, digamos, em um segmento comum de páginas físicas. Esta forma de

armazenamento leva a índices mais compactos e não impõe limitações ao tamanho das componentes, resolvendo assim o problema de espaço de armazenamento. Porém, para endereçar o problema de contigüidade física, é necessário definir uma estratégia de gerência para o segmento de página físicas de tal forma que componentes vizinhas no espaço geográfico tenham as suas geometrias armazenadas em páginas próximas e, idealmente, que permita armazenar nas mesmas páginas diferentes tipos de geometrias.

7.3 Exemplos de otimização de consultas

Esta seção aborda, por meio de exemplos, o problema de otimização de consultas espaciais, à luz dos comentários feitos na Seção 7.2.

7.3.1 Exemplo de processamento de seleção por região

Considere um banco de dados geográfico sobre cidades brasileiras consistindo de apenas um conjunto de geo-objetos, CIDADES. Para cada cidade (do mundo real), há um objeto em CIDADES, cujos atributos convencionais correspondem aos dados convencionais da cidade e cuja localização é dada por um par ordenado indicando a latitude e longitude da cidade; ou seja, a localização destes geo-objetos é representada por um ponto em coordenadas geográficas. No que se segue, suporemos que estas coordenadas estão armazenadas junto com o próprio geo-objeto, embora as operações sobre elas possam se beneficiar de um índice espacial sobre CIDADES.

Considere agora a consulta Q, formulada informalmente como:

Q. *Selecione o nome e população de todas as cidades a menos de 50km da cidade de Belo Horizonte e com mais de 50 mil habitantes.*

Esta consulta seria definida através do comando:

```
SELECT d.nome, d.populacao
      FROM d Cidade, c Cidade
      WHERE c.nome = "Belo Horizonte"
            and distance(d,c) < 50
            and d.populacao > 50.000
```

Esta consulta exemplifica uma seleção por região, definida por um círculo de raio de 50km, cujo centro está na localização geográfica da

cidade de Belo Horizonte. Há essencialmente três planos de execução mais plausíveis para esta consulta (entre outros possíveis), discutidos a seguir.

Considere inicialmente o plano de execução P_1 :

- P_{11} . Determine a posição b de Belo Horizonte;
- P_{12} . Determine o conjunto C' de todas as cidades a 50km de b ;
- P_{13} . Determine o subconjunto C de C' de cidades com mais de 50 mil habitantes.

As subconsultas P_{11} e P_{13} não envolvem nenhum operador espacial e, portanto, são subconsultas convencionais de Q para este plano. Já a subconsulta P_{12} envolve o operador de distância e é a única subconsulta espacial de Q em P_1 .

O processamento de P_1 começa com a execução da subconsulta convencional P_{11} , que retorna a localização da cidade de Belo Horizonte, o ponto b .

Suponha que o banco de dados inclua um índice espacial sobre CIDADES, que chamaremos de ID-CIDADES, invisível aos usuários, que facilite identificar todos os objetos de CIDADES cuja localização esteja a uma certa distância de um dado ponto. Suponha ainda que o índice espacial seja impreciso para este caso, no sentido de também retornar objetos que não estão à distância especificada do ponto. Esta imprecisão é típica de métodos de indexação espacial, e contrasta com os métodos de indexação tradicionais, que são precisos.

O processamento da subconsulta P_{12} segue em três passos:

- P_{121} . Através de ID-CIDADES, determine que objetos em CIDADES podem estar a menos de 50km de b , criando o conjunto I .
- P_{122} . Leia da memória secundária para a memória principal a representação interna dos objetos indicados por I , criando o conjunto C'' .
- P_{123} . Determine que elementos de C'' estão de fato a menos de 50km de b , criando o conjunto C' .

Os elementos de I são apontadores para a localização em memória secundária da representação interna de objetos em CIDADES. Além

disto, como supomos que o índice não é exato, I poderá apontar para objetos a mais de 50km de b. Já os elementos de C' e C'' são representações internas de objetos em CIDADES, armazenadas em memória principal. Assim, o passo P_{121} corresponde à fase de filtragem P_{122} e P_{123} , à fase de leitura e refinamento.

O processamento da subconsulta P_{13} é agora simples, bastando extrair de cada elemento de C' os valores dos atributos de interesse e eliminando aqueles que não possuem mais de 50 mil habitantes, criando assim o conjunto C, que é a resposta da consulta Q. Os elementos de C serão pares de valores, correspondendo ao nome e à população de uma cidade.

O processamento de P_{121} , P_{122} , P_{123} e P_3 poderá ser concatenado em *pipeline*, ou seja, cada elemento identificado em P_{121} poderá ser imediatamente processado por P_{122} e filtrado por P_{123} , eventualmente gerando um par em C, ao ser tratado em P_{13} . Isto evita a criação explícita dos resultados intermediários I, C' e C'', levando a uma razoável economia no processamento global da consulta.

O segundo plano, P_2 , inverte a ordem de execução das restrições de Q:

- P_{21} . Determine a posição b de Belo Horizonte;
- P_{22} . Determine o conjunto C' de todas as cidades com mais de 50 mil habitantes;
- P_{23} . Determine o subconjunto C de C' de cidades a 50km de b.

Os elementos de C' serão triplas em memória principal indicando o nome, população e localização de objetos em CIDADES com mais de 50 mil habitantes. O processamento da subconsulta convencional P_{22} será particularmente eficiente se partirmos do pressuposto que o banco de dados inclui um índice convencional sobre CIDADES, que chamaremos de ID-POP, invisível aos usuários, que permita identificar todos os objetos de CIDADES cuja população esteja em uma faixa de valores. Tal índice será provavelmente preciso, no sentido de retornar exatamente o subconjunto de cidades cuja população está na faixa de valores dada. Porém, note que o conjunto C' corresponderá a todas as cidades brasileiras com mais de 50 mil habitantes. Neste caso, a execução de P_{22} dá-se da seguinte forma:

- P_{221} . Através de ID-POP, determine que objetos em CIDADES possuem população com mais de 50 mil habitantes, criando o conjunto J;
- P_{222} . Leia da memória secundária para a memória principal a representação interna dos objetos indicados por J, selecionando seu nome, população e localização e criando o conjunto C'.

O processamento da subconsulta P_{23} é então simplificado, bastando selecionar do conjunto C' (suposto já em memória principal) aqueles elementos que estão de fato a menos de 50km de b, criando o conjunto C. Novamente, o processamento de P_{221} , P_{222} e P_{23} poderá ser sincronizado, evitando a criação de J e C'.

O terceiro plano, P_3 , constrói dois conjuntos independentemente para, em seguida, computar a sua interseção:

- P_{31} . Determine a posição b de Belo Horizonte;
- P_{32} . Determine o conjunto C' de todas as cidades com mais de 50 mil habitantes;
- P_{33} . Determine o conjunto C" de cidades a 50km de b;
- P_{34} . Determine a interseção C de C' e C".

A execução detalhada deste plano é uma combinação da discutida para os dois planos anteriores.

Até este ponto, descrevemos apenas o detalhamento de três planos plausíveis para execução da consulta. A estimativa do custo de cada plano depende essencialmente de estimar o número de objetos identificados pelo índice ID-CIDADES e pelo índice ID-POP e está fora do escopo deste texto. Por exemplo, o plano P_1 será mais vantajoso do que P_2 se ID-CIDADES for razoavelmente eficiente, ou seja, não retornar muitas cidades que não estão a menos de 50km de Belo Horizonte, e se o número de cidades brasileiras com mais de 50 mil habitantes for muito grande. Ou seja, se a construção da resposta intermediária registrada no conjunto I for mais barata do que a registrada no conjunto J.

7.3.2 Exemplo de processamento de junção espacial

Suponha agora que o banco de dados geográfico contém também um conjunto de geo-objetos, AEROPORTOS, tal que, para cada aeroporto

brasileiro, há um objeto em AEROPORTOS cujos atributos convencionais correspondem aos dados convencionais do aeroporto e cuja localização é dada novamente pela latitude e longitude do aeroporto. Suponha ainda que o banco inclua um índice espacial sobre AEROPORTOS, chamado ID-AEROPORTOS, que permita identificar todos os aeroportos cuja localização esteja a uma certa distância de um dado ponto.

Considere a seguinte consulta "Selecione todas as cidades que são sede de municípios e que são atendidas por aeroportos pavimentados". Esta consulta exemplifica uma junção espacial. Para torná-la mais precisa, suponha que uma cidade é atendida por um aeroporto se a cidade dista menos de 50km do aeroporto. A consulta reformulada seria então:

Q. *Selecione o nome de cada cidade e o nome de cada aeroporto tais que a cidade é sede de município, o aeroporto é pavimentado e a cidade dista menos de 50km do aeroporto.*

Novamente há vários planos de execução possíveis para esta consulta, sendo os principais discutidos a seguir. Todos os planos, sob certo aspecto, reduzem o processamento da junção espacial ao processamento de várias seleções por região.

Considere o plano P_1 inicialmente:

- P_{11} . Determine o conjunto R contendo o nome n e localização l de cada aeroporto pavimentado;
- P_{12} . Inicialize o conjunto C como vazio;
- P_{13} . Para cada elemento $(n,l) \in R$, faça:
 - P_{131} . Determine o conjunto dos nomes c de cidades que são sede de município e que estão a menos de 50km de l, acrescentando o par (c,n) a C.

O processamento de P_1 começa com a execução da subconsulta convencional P_{11} , que retorna o nome e a localização de todos os aeroportos pavimentados. Para cada um destes, executa-se uma seleção por região para determinar as sedes dos municípios que estão a menos de 50km do aeroporto. O processamento destas seleções segue então de acordo com uma das estratégias discutidas no exemplo anterior.

O segundo plano, P_2 , é simétrico a este, recuperando primeiro o conjunto de todas as cidades que são sede de município para depois determinar quais aeroportos pavimentados estão a menos de 50km de cada cidade. Como supusemos que há um índice espacial sobre o conjunto de aeroportos, este segundo plano compete com o primeiro. A determinação de qual dos dois planos é mais vantajoso depende de qual conjunto for menos numeroso: os aeroportos pavimentados ou as cidades que são sede de município.

O terceiro plano, P_3 , primeiro constrói de forma independente dois conjuntos para, em seguida, computar a sua interseção:

- P_{31} . Determine o conjunto R contendo o nome n e localização l de cada aeroporto pavimentado;
- P_{32} . Determine o conjunto S contendo o nome c e localização s de cada sede de município;
- P_{33} . Determine o conjunto C tal que $(c,l) \in C$ se e somente se $(n,l) \in R$ e $(c,s) \in S$ e a distância de l a s é menor do que 50km.

O passo P_{33} esconde uma iteração sobre os elementos de R e sobre os elementos de S. Em um caso, por exemplo, a cada elemento de R, todos os elementos de S seriam visitados para determinar quais satisfazem a condição imposta. Logo, o plano P_3 nada mais é do que uma reformulação do plano P_1 .

É possível, no entanto, implementar outras formas de computar o passo P_{33} que tornam o plano P_3 diferente dos demais. Discutiremos a seguir, brevemente, como aplicar o *método de varredura do espaço* para computar este passo. Intuitivamente, este método consiste em varrer o espaço em uma dada direção processando todos os objetos encontrados. No caso abaixo, usaremos um meridiano, varrendo o plano em ordem crescente de valor de longitude:

- P_{331} . Ordene inicialmente os elementos de R e S em valor crescente da longitude das suas localizações;
- P_{332} . Para cada elemento de menor longitude ainda não processado, faça:
 - P_{3321} . Suponha que este elemento seja um par $(n,l) \in R$ (se for um par $(c,s) \in S$, o processamento é inteiramente semelhante);

- P_{3322} . Determine todos os pares $(c,s) \in S$ tais que a distância de l a s é menor do que 50km;
- P_{3323} . Considere o par (n,l) e todos os pares (c,s) encontrados como processados.

Novamente a escolha entre os planos depende de uma estimativa do custo de cada um, discussão que está fora do escopo deste texto.

7.3.3 Exemplo de processamento de superposição de geo-campos

Considere agora que o banco de dados geográfico contém dois geo-campos temáticos, SOLOS e VEGETAÇÃO, indicando os tipos de solo e de vegetação do Estado de São Paulo. Suponha que os geo-campos possuam representações por subdivisão planar. Suponha ainda, por simplicidade, que as duas representações utilizem a mesma escala e projeção de tal forma que seja possível operar sobre elas sem conversão.

Considere agora a consulta Q , formulada como:

Q . Crie um geo-campo temático com o remanescente de mata atlântica em latossolo roxo na região B

onde B é uma *janela*, ou seja, um retângulo definido por um par (ie,sd) , no mesmo sistema de coordenadas cartográficas das representações, indicando o canto inferior esquerdo e superior direito do retângulo. Esta consulta difere das anteriores por criar um campo temático a partir de dois outros armazenados no banco de dados, tendo uma restrição dada por um retângulo.

Discutiremos a execução de apenas um plano, P_1 , para esta consulta:

- P_{11} . Utilizando SOLOS, crie o campo temático SOLOS-B indicando os tipos de solo em B ;
- P_{12} . Utilizando VEGETAÇÃO, crie o campo temático VEGETAÇÃO-B indicando os tipos de vegetação em B ;
- P_{13} . Crie o campo temático VEGET-REM, resposta da consulta, aplicando uma variante da operação de superposição de campos a SOLOS-B e VEGETAÇÃO-B;

Ao contrário dos exemplos anteriores em que a otimização do plano dependia da existência de índices espaciais, o processamento será facilitado aqui se supusermos a existência de estruturas de

armazenamento eficientes para as representações que facilitem recuperar as parcelas das representações que cobrem o retângulo B. Suponha então que as representações envolvidas sejam armazenadas em estruturas que permitam trazer para memória principal apenas as páginas que contêm os dados de vegetação ou solos na região B ou, pelo menos, em uma região B' que cubra B e seja significativamente menor do que a região abrangida pelas representações.

Em uma implementação mais simples, o subsistema de armazenamento recuperará as páginas apropriadas, filtrando-as se necessário, e materializará as representações vetoriais dos campos SOLOS-B e VEGETAÇÃO-B. Sobre estes campos intermediários, o processador de consultas aplicará então a operação de superposição de representações apropriada para criar o campo temático VEGET-REM, resposta da consulta.

Já em uma implementação mais sofisticada, o subsistema de armazenamento tratará os campos SOLOS-B e VEGETAÇÃO-B como *visões* não materializadas dos campos originais. Mais explicitamente, quando o processador de consultas executar a operação de superposição de representações, o subsistema de armazenamento recuperará, para memória principal, as páginas contendo os dados de vegetação ou solos à medida que se tornem necessárias para a operação, sendo de fato criada apenas a representação do campo temático VEGET-REM.

7.4 Computação de operações básicas

Conforme discutido brevemente na seção 7.2, dada uma consulta Q, o processador de consultas é responsável por preparar um plano de execução para Q, consistindo de operações de mais baixo nível, implementadas pelo subsistema de armazenamento.

Esta seção aborda então implementações alternativas para duas das principais operações que um subsistema de armazenamento deve oferecer, que correspondem diretamente à seleção e à junção espaciais. As implementações dependem da utilização ou não de índices espaciais e, conseqüentemente, exibem desempenho bastante diferenciado.

7.4.1 Computação de seleções espaciais

Uma *seleção espacial* é uma expressão da forma $\sigma[B(x)]$, onde $B(x)$, chamado de *predicado de seleção*, é uma expressão booleana com apenas uma variável livre, x , tal que $B(x)$ envolve operações espaciais. Dado um conjunto S de geo-objetos, $\sigma[B(x)](S) = S'$ se e somente se S' é o conjunto de todos os objetos $s \in S$ tais que $B(s)$ é verdadeira.

Um exemplo de seleção espacial é a expressão $\sigma[\text{distancia}(x,p) < 100]$, onde x é a variável livre e p é uma constante do tipo ponto.

Seja S um conjunto de geo-objetos no que se segue. Usaremos r.e.m. para abreviar o termo “retângulo envolvente mínimo”.

A estratégia mais simples para computar $\sigma[B(x)](S)$, denominada *seleção por pesquisa exaustiva*, é apresentada em pseudo-código na Figura 7.2. Consiste em trazer a representação geométrica de cada objeto $s \in S$ para memória principal e testar se $B(s)$ é verdadeira. Possui custo proporcional à cardinalidade de S , se os objetos de S não estiverem agrupados em memória secundária, ou proporcional ao número de páginas ocupadas por S , em caso contrário. Esta estratégia é adotada quando não há índices para S , ou quando a cardinalidade de S é suficientemente pequena para não justificar estratégias mais sofisticadas.

```

SELEÇÃO_POR_PESQUISA_EXAUSTIVA(S,B;R)
S - conjunto de geo-objetos
B - expressão booleana envolvendo comparações
    espaciais com apenas uma variável livre x
R - subconjunto de S dos objetos satisfazendo B
Início
  R = ∅
  Para cada s em S faça
    Início Recupere s para memória principal
      Se s satisfaz B, acrescente s a R
    Fim
  Fim
Fim

```

Figura 7.2 – Seleção por pesquisa exaustiva.

Conforme discutido no Capítulo 6, quando a cardinalidade esperada de S é suficientemente grande e a frequência de acesso a S é alta, costuma-se definir um índice para acelerar o acesso aos objetos de S .

A estratégia para computar $\sigma[B(x)](S)$, denominada *seleção por índice*, é apresentada em pseudo-código na Figura 7.3 e possui duas etapas. A *etapa de filtragem* utiliza um índice para selecionar parte dos objetos de S , criando um conjunto P de apontadores para objetos em S . A *etapa de refinamento* recupera para memória principal a representação geométrica de cada objeto s apontado por P , acrescentando s à resposta, se $B(s)$ for verdadeira.

```

SELEÇÃO_POR_INDICE(S,B,I,C;R)
S - conjunto de geo-objetos
B - predicado de seleção
I - índice sobre S
C - expressão booleana envolvendo comparações
    espaciais com apenas uma variável livre
    que pode ser resolvida por I
R - conjunto dos objetos de S que satisfazem B
Início
  R, P = ∅
  FILTRO(I,C;P)                % Etapa 1: Filtragem
  Para cada p em P faça        % Etapa 2: Refinamento
  Início
    Recupere o objeto s de S apontado por p
    Se s satisfizer B, acrescente s a R
  Fim
Fim
FILTRO(I,C;P)
Início
  P = ∅
  Acrescente a P todos os apontadores para
    objetos de S que satisfazem C, usando I
Fim

```

Figura 7.3 – Seleção por índice.

Normalmente, o índice não resolve B diretamente, mas uma outra expressão booleana C , também com uma variável livre, tal que:

(1) para todo geo-objeto s , se $C(s) = \text{falso}$ então $B(s) = \text{falso}$

Assim, se um geo-objeto s não é selecionado pelo índice (ou seja, $C(s)$ é falso), então s não pertence à resposta da seleção (ou seja, $B(s)$ também é falso). Porém, o índice pode selecionar um objeto s que não pertence à resposta (ou seja, pode existir um geo-objeto s tal que $C(s) \wedge \neg B(s)$).

O resto desta seção refina a função auxiliar FILTRO para árvores-R.

Dados dois inteiros $m > 1$ e $n > 1$, recorde que uma árvore-R de ordem (m,n) para S é uma árvore de busca balanceada tal que (veja exemplo na Figura 7.47.4):

- todas as folhas estão no mesmo nível;
- cada folha contém entre $\lceil n/2 \rceil$ e n entradas;
- cada nó interior contém entre $\lceil m/2 \rceil$ e m filhos, exceto a raiz, que contém entre 2 e m filhos;
- para cada objeto $s \in S$, existe uma (única) entrada em uma (única) folha M , consistindo de um ponteiro para s em conjunto com o r.e.m. de s ; dizemos que s é *apontado* por M ;
- cada nó interior N contém uma entrada para cada filho M de N , consistindo de um ponteiro para M e o r.e.m. de M , ou seja, o r.e.m. de todos os retângulos armazenados em M .

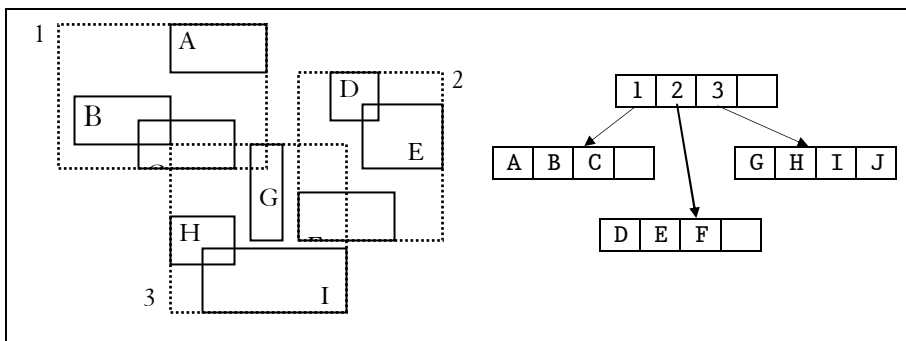


Figura 7.4 – Exemplo de uma árvore-R.

A computação de $\sigma[B(x)](S)$ é guiada pelos retângulos armazenados na árvore-R, dependendo evidentemente da expressão booleana B .

Considere, por exemplo, a computação da seleção $\sigma[\text{contem}(x,p)](S)$, onde p é uma constante do tipo ponto.

Voltando à **Figura 7.47.4**, para determinar quais objetos $s \in S$ contêm p , proceda da seguinte forma. Observe inicialmente que os retângulos 1 e 3, armazenados na raiz, contêm p ; estes retângulos correspondem ao primeiro e ao segundo filho da raiz. Em seguida, repita o processo descendo por estes nós. Assim, descendo pelo primeiro filho da raiz, observe que o retângulo C contém p e acrescente o ponteiro a ele associado a uma lista temporária L. Descendo pelo terceiro filho da raiz, observe que nenhum retângulo aí armazenado contém p . Como último passo, recupere para memória principal o objeto s apontado pelo (único) ponteiro em L e teste se s de fato contém p ; se contiver, $\sigma[\text{contem}(x, p)](S) = \{s\}$; caso contrário, $\sigma[\text{contem}(x, p)](S) = \emptyset$.

De forma geral, o uso de árvores-R como índices coloca um desafio adicional pois as folhas da árvore armazenam retângulos aproximando a geometria dos objetos, e não a geometria em si. Além disto, cada nó interior N contém uma entrada para cada filho M de N, consistindo de um ponteiro para M e o r.e.m. dos retângulos armazenados em M.

Portanto, para que árvores-R possam ser utilizadas na etapa de filtragem, devemos definir uma segunda expressão booleana C' , que chamaremos de *companheira* de C, tal que:

- (2) para todo geo-objeto s , para todo retângulo r tal que r é igual ou contém o r.e.m. de s , se $C'(r) = \text{falso}$ então $C(s) = \text{falso}$

Dizemos que uma expressão booleana $C(x)$, com uma variável livre x varrendo geo-objetos, *pode ser filtrada por árvores-R* se somente se é possível definir uma expressão booleana *companheira* para C.

Note que C' não é a expressão booleana C da Figura 7.3. Porém, de (1) e (2) podemos deduzir diretamente que

- (3) para todo geo-objeto s , para todo retângulo r tal que r é igual ou contém o r.e.m. de s , se $C'(r) = \text{falso}$ então $B(s) = \text{falso}$

Ou seja, apesar de trabalhar com retângulos, através da noção de expressão *companheira*, as árvores-R podem ser utilizadas como filtros.

Freqüentemente, C' é a própria expressão C. Por exemplo, se tivermos $C(x) = \text{contem}(x, k)$, onde x é uma variável varrendo geo-objetos e k uma constante espacial, então $C(r) = \text{contem}(r, k)$, para todo retângulo que contém ou é igual ao r.e.m. de x .

Porém, isto nem sempre é verdade. Tome, por exemplo, $C(x) = \text{toca}(x,k)$, onde x é uma variável varrendo geo-objetos e k uma constante espacial. Neste caso, $C(x)$ não implica em $C(r)$, para todo retângulo que contém ou é igual ao r.e.m. de x . Por exemplo, teríamos que definir C' como:

$$C'(r) = \text{superpõe}(r,k) \vee \text{toca}(r,k)$$

Em outras situações, não é realmente possível definir C' de tal forma a tornar árvores-R úteis para filtrar $C(x)$. Por exemplo, tomemos $C(x) = \text{disjunto}(x,k)$, onde x é uma variável varrendo geo-objetos e k uma constante espacial. Para este operador, teríamos que definir C' como:

$$C'(r) = \text{disjunto}(r,k) \vee \text{superpõe}(r,k)$$

o que seria inútil como filtro.

De posse destas noções, o refinamento da função auxiliar FILTRO para árvores-R é imediato e mostrado na Figura 7.5. Note que esta função continua a receber a expressão booleana C sobre objetos de S , por compatibilidade com FILTRO, e internamente determina (por um processo não especificado) a expressão C' companheira de C , que supõe-se existir.

O pseudo-código apresentado consiste de um caminhamento em amplitude na árvore, com o auxílio da variável Q , eliminando os nós cujos r.e.m. não satisfazem C' , e retornando os apontadores para objetos, contidos nas folhas, cujos r.e.m.'s satisfazem C' . Portanto, consistentemente com a estratégia de seleção por índice, esta função não acessa a geometria dos objetos.

Por fim, simplificadamente, o custo de uma pesquisa por índice, utilizando árvores-R, é proporcional ao número de nós da árvore-R lidos, somado ao número de objetos apontados pela árvore-R cujo retângulo envolvente satisfaz C' . Uma análise detalhada pode ser encontrada em (Aboulnaga e Naughton, 2000; Kriegel et al., 1990; Ciferri et al., 2000).

FILTRO_ARVORE_R(A,C;P)

- A - apontador não nulo para raiz de uma árvore-R sobre um conjunto de geo-objetos S.
Cada nó N é uma estrutura da forma:
lt(N) - lista de entradas;
rem(N) - r.e.m. das entradas em lt(N)
Cada entrada E é uma estrutura da forma:
Nó interior:
pt(E) - apontador para a raiz de uma sub-árvore de A
rem(E) - retângulo envolvente mínimo da sub-árvore apontada por E
Folha:
pt(E) - apontador para um objeto em S
rem(E) - retângulo envolvente mínimo do objeto apontado por E
- C - expressão booleana envolvendo comparações espaciais com apenas uma variável livre que pode ser filtrada por A
- P - conjunto de apontadores para objetos de S

Início

Determine a expressão C' companheira de C

P = \emptyset

Q = {A}

Enquanto Q $\neq \emptyset$ faça:

Início

Selecione q em Q, removendo-o de Q

Se q for uma folha,

Para cada entrada E em ls(q),

Se C'(rem(E)), acrescente pt(E) a P

Senão

Para cada entrada E em ls(q),

Se C'(rem(E)), acrescente pt(E) a Q

Fim

Fim

Figura 7.5 – Filtro por árvores-R.

7.4.2 Computação de junções espaciais

Uma *junção espacial* é uma expressão da forma $\chi[J(x,y)]$, onde $J(x,y)$, chamado de *predicado de junção*, é uma expressão booleana envolvendo apenas comparações espaciais com duas variáveis livres x e y . Dados dois conjuntos S e T de geo-objetos, $\chi[J(x,y)](S,T)=R$ se e somente se R é o conjunto de todos os pares de objetos $(s,t) \in S \times T$ tais que $J(s,t)$ é verdadeira.

Um exemplo de junção espacial é a expressão $\chi[\text{distancia}(x,y) < 100]$, onde x e y são variáveis. O predicado de junção define o conjunto de pares de geo-objetos, x e y , que estão a menos de 100 e a mais de 50 (metros) um do outro.

Computar $\chi[J(x,y)](S,T)$ é mais complexo do que computar junções não espaciais pois não há uma ordem total entre geo-objetos que preserve proximidade espacial. Ou seja, não há uma função que mapeie qualquer par (S,T) de conjuntos de geo-objetos em uma seqüência Q de geo-objetos tal que todo objeto em $S \cup T$ ocorre uma única vez em Q e, para todo $(s,t) \in S \times T$, se s e t estão espacialmente próximos, então s e t estão próximos em Q . Esta característica dos geo-objetos afeta as estratégias tradicionais para computar junção, tornando-as inaplicáveis ou menos eficientes.

Esta seção discute algumas estratégias para computar junção espacial, seguindo basicamente (Brinkhoff et al., 1993; Brinkhoff et al., 1994; Gunther et al., 1993; Patel e DeWitt, 1996).

Considere inicialmente a estratégia de *junção por pesquisa exaustiva* para computar $\chi[J(x,y)](S,T)$, apresentada na Figura 7.6. Esta estratégia consiste em duas iterações aninhadas varrendo todos os pares de objetos (s,t) em $S \times T$, testando $J(s,t)$ para cada um deles. Ou seja, consiste de uma pesquisa exaustiva sobre $S \times T$, o espaço de busca de $\chi[J(x,y)](S,T)$.

Esta estratégia levanta dois problemas básicos relativos à computação de uma junção espacial. Primeiro, a transferência dos geo-objetos da memória secundária para memória principal pode ser uma operação de alto custo, quando as geometrias dos objetos são extensas. Esta transferência pode, de fato, forçar a liberação de geo-objetos de T para permitir a reutilização de espaço na área de trabalho em memória

principal. Desta forma, o mesmo objeto em T poderá ser transferido várias vezes para memória principal. No pior caso, cada objeto em T será lido n vezes, onde n é a cardinalidade de S . Assim, o próprio aninhamento das iterações torna-se ainda mais ineficiente.

Segundo, o teste do predicado de junção pode também ser uma operação de alto custo, principalmente se a geometria dos objetos consistir de um grande número de pontos. O problema é agravado pelo fato de que este teste será realizado para todos os pares $(s,t) \in S \times T$.

```

JUNÇÃO_POR_PESQUISA_EXAUSTIVA(S,T,J;R)

S,T - conjuntos de objetos espaciais
J   - expressão booleana envolvendo comparações
      espaciais com duas variáveis livres
R   - conjunto de pares de objetos em  $S \times T$  que
      satisfazem J

Início
  R =  $\emptyset$ 
  Para cada s em S faça
    Início
      Recupere s de memória secundária
              para memória principal
      Para cada t em T faça
        Início
          Se t não está em memória principal,
            recupere t de memória secundária
                    para memória principal
          Se J(s,t),
            acrescente (s,t) a R
        Fim
      Fim
    Fim
  Fim

```

Figura 7.6 – Junção por pesquisa exaustiva.

Uma segunda estratégia para computar $\chi[J(x,y)](S,T)$, chamada de *junção por particionamento e intercalação* (Patel e DeWitt, 1996), opera em duas etapas, filtragem e refinamento, sem recorrer a índices. De forma semelhante à seleção espacial, assumiremos que a etapa de

filtragem aplica-se a um predicado $K(x,y)$ tal que, para todo x, y , se $K(x,y)$ é verdadeiro então $J(x,y)$ também é verdadeiro. Além disto, o predicado K deve ser tal que:

- (4) para todo par (s_1, s_2) de geo-objetos, com r.e.m.'s r_1 e r_2 respectivamente, se r_1 e r_2 não se superpõem então $K(s_1, s_2) = \text{falso}$

O passo inicial da etapa de filtragem cria e armazena em memória secundária dois conjuntos temporários, S^0 e T^0 , da seguinte forma. Cada geo-objeto $s \in S$ é lido para memória principal, gerando um objeto $s^0 \in S^0$ da forma $s^0 = (p,r)$, onde p é o identificador de s em memória secundária, denotado $id(s^0)$, e r é o r.e.m. de s , denotado $rem(s^0)$. O mesmo processo é aplicado a T , gerando T^0 .

O próximo passo desta etapa determina todos os pares de objetos $(s^0, t^0) \in S^0 \times T^0$ tais que $rem(s^0)$ e $rem(t^0)$ se superpõem. Para cada par (s^0, t^0) que passar no teste, o par de identificadores $(id(s^0), id(t^0))$ é acrescentado à resposta desta fase.

Se os conjuntos temporários S^0 e T^0 puderem ser simultaneamente trazidos para memória principal, uma técnica de varredura do plano (Preparata e Shamos, 1988) poderá ser adotada para determinar quais retângulos se superpõem. Por exemplo, um conjunto de retângulos pode ser varrido em ordem crescente da coordenada do canto inferior esquerdo, como ilustra a Figura 7.7. Um algoritmo, baseado em uma técnica de varredura, para determinar quais pares de retângulos $(r, s) \in A_1 \times A_2$ se superpõem é apresentado na Figura 7.8.

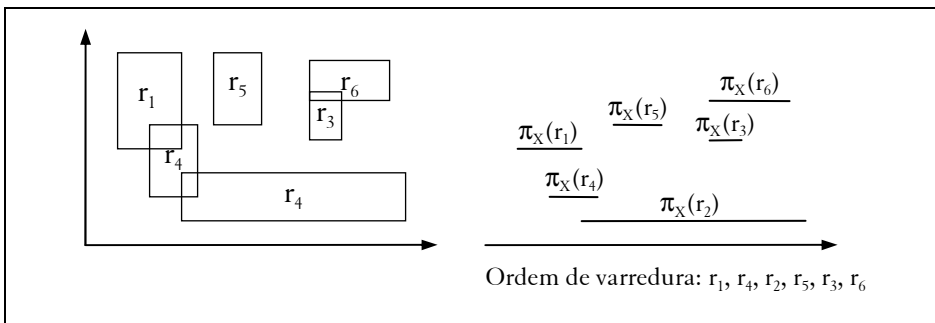


Figura 7.7 – Exemplo de varredura do plano para retângulos.

```

VARREDURA_DO_PLANO(A1,A2;B)
A1,A2 - conjuntos de retângulos com lados paralelos
        aos eixos x e y;
        xe(r) e xd(r) denotam as coord. x
        dos vértices à esquerda e à direita de r
B      - (r,s)∈A1×A2 tal que r e s se sobrepõem
Início
  R = ∅
  C = A1 ∪ A2
  Ordene C crescentemente por xe(r)
  Enquanto C ≠ ∅ faça
  Início
    Seja r∈C com menor valor de xe(r)
    Remova r de C e suponha que r∈Ai
    % A(i+1) : soma módulo 2
    Para cada s∈A(i+1) até que xe(s)>xd(r) faça
    % xe(r)≤xe(s)≤xd(r): r e s se superpõem no eixo x
      Se r e s também se superpõem no eixo y,
        acrescente (r,s) a B
  Fim
Fim

```

Figura 7.8 – Técnica de varredura para determinar superposição de retângulos.

Se os conjuntos não puderem ser trazidos para memória principal, então cada conjunto deve ser particionado em p conjuntos não necessariamente disjuntos S^1, \dots, S^p e T^1, \dots, T^p , respectivamente. Estes novos conjuntos são tais que S^k e T^k , para $k=1, \dots, p$, podem ser simultaneamente trazidos para memória e, para todo $(s^0, t^0) \in S^0 \times T^0$ tais que $\text{rem}(s^0)$ e $\text{rem}(t^0)$ se superpõem, se $s^0 \in S^k$ então $t^0 \in T^k$.

A criação destas partições segue a seguinte tática (Patel e DeWitt, 1996):

- (a) Ao criar S^0 e T^0 , determine R , o r.e.m. de todos os retângulos em S^0 e T^0 .
- (b) Considerando o tamanho da área de trabalho disponível em memória principal e o tamanho de cada elemento em S^0 e T^0 ,

determine o número p de partes em que R deve ser dividido, criando retângulos R^1, \dots, R^p (ver **Figura 7.9**).

(c) Os conjuntos S^1, \dots, S^p e T^1, \dots, T^p são criados da seguinte forma:

para cada $k \in [1, p]$, para cada $s^0 \in S^0$,
 $s^0 \in S^k$ se e somente se $\text{rem}(s^0)$ e R^k se superpõem

para cada $k \in [1, p]$, para cada $t^0 \in T^0$,
 $t^0 \in T^k$ se e somente se $\text{rem}(t^0)$ e R^k se superpõem

O número p de partes em que R deve ser dividido pode ser estimado da seguinte forma.

Sejam $|S^0|$ e $|T^0|$ as cardinalidades de S^0 e T^0 , respectivamente. Seja M o espaço disponível em memória principal e N o tamanho de cada objeto em S^0 e T^0 . Como é necessário manter S^k e T^k simultaneamente em memória principal, temos que:

$$p = \lceil (|S^0| + |T^0|) * N / M \rceil$$

Após o particionamento, para cada $k \in [1, p]$, os conjuntos S^k e T^k são suficientemente pequenos para serem simultaneamente trazidos para memória principal.

Novamente a técnica de varredura do plano pode ser adotada para determinar todos os pares de objetos $(s^k, t^k) \in S^k \times T^k$ tais que $\text{rem}(s^k)$ e $\text{rem}(t^k)$ se superpõem. Para cada par (s^k, t^k) que passar no teste, $(\text{id}(s^k), \text{id}(t^k))$ é acrescentado à resposta desta fase. Isto conclui a etapa de filtragem. O resultado é um conjunto F de pares de identificadores de objetos em $S^0 \times T^0$.

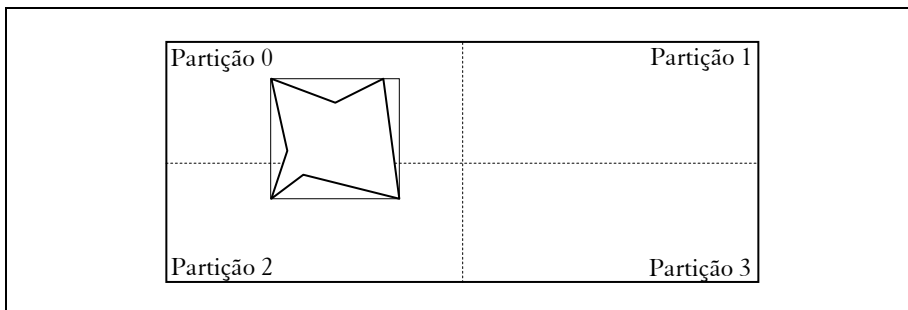


Figura 7.9 – Divisão dos conjuntos originais contendo os r.e.m.'s.

A etapa de refinamento procede da seguinte forma. Para evitar acesso aleatório aos objetos de S e T , os pares no resultado F da etapa de filtragem são ordenados tendo como chave a primeira coordenada seguida da segunda coordenada. Duplicatas são eliminadas neste processo, gerando o conjunto G . Em seguida, para cada par $(d,e) \in G$, o objeto $s \in S$ tal que $d = \text{id}(s)$ é trazido para a área de trabalho, se já lá não estiver. Para cada par $(f,g) \in G$ tal que $f = d$, o objeto $t \in T$ tal que $g = \text{id}(t)$ também é trazido para a área de trabalho. Se $J(s,t)$ for satisfeito, então o par (s,t) é finalmente adicionado à resposta de $\chi[J(x,y)](S,T)$.

Como na seleção espacial, uma outra forma de reduzir o custo de computar uma junção espacial $\chi[J(x,y)](S,T)$ consiste em utilizar índices.

Uma estratégia genérica, denominada *junção por índice*, também possui duas etapas, exceto que a estratégia utiliza um índice para a etapa de filtragem. A **Figura 7.10** apresenta a estratégia em pseudo-código e a **Figura 7.11** ilustra a estratégia esquematicamente.

De acordo com (Brinkhoff et al., 1994), a Etapa 2 é a de maior custo, tanto em termos de acesso a memória secundária para recuperar a geometria exata dos objetos, quando em termos de tempo de processamento para computar o predicado de junção.

Note que a estratégia, como apresentada na **Figura 7.10**, é apenas um *framework*, que deve ser refinado com implementações específicas em cada etapa. Para a etapa de filtragem, por exemplo, (Brinkhoff et al., 1993; Brinkhoff et al., 1994) usam árvores-R* e (Patel e DeWitt, 1996; Lo e Ravishankar, 1996) propõem uma estrutura baseada em *hash*. Já para a etapa de refinamento, (Brinkhoff et al., 1994) adota um algoritmo de varredura do plano, conforme anteriormente apresentado.

O resto desta seção discute o refinamento da etapa de filtragem para o caso em que os índices são árvores-R. (Brinkhoff et al., 1993; Brinkhoff et al., 1994; Gunther et al., 1993).

Seja $K(x,y)$ o predicado a ser filtrado pelas árvores-R. Novamente, para que árvores-R possam ser utilizadas, devemos definir uma segunda expressão booleana K' , que chamaremos de *companheira* de K , tal que:

- (5) para todo par (s_1, s_2) de geo-objetos, para todo par (r_1, r_2) de retângulos tais que r_i é igual ou contém o r.e.m. de s_i , para $i=1,2$, se $K'(r_1, r_2) = \text{falso}$ então $K(s_1, s_2) = \text{falso}$

O refinamento proposto, apresentado como *filtro de junção por árvore-R* na Figura 7.12, consiste de uma dupla pesquisa por amplitude nas duas árvore-R. A variável Q contém apenas os pares de retângulos (r_1, r_2) tais que $K'(r_1, r_2) = \text{verdadeiro}$. Pela propriedade acima de K' e pela construção das árvores-R, para todo par $(s, t) \in S \times T$, se $K(s, t) = \text{verdadeiro}$, então $(pt(s), pt(t)) \in P$, onde P é a resposta devolvida pelo filtro (mas o converso é falso, ou seja, poderá existir $(s, t) \in S \times T$ tal que $K(s, t) = \text{falso}$ e $(pt(s), pt(t)) \in P$ pelo próprio fato das árvores-R trabalharem com aproximações, ou seja, não serem filtros exatos.

```
JUNÇÃO_POR_INDICES(S,T,J,U,V,K;R)

S,T - conjuntos de objetos espaciais
J    - expressão booleana envolvendo comparações
      espaciais com duas variáveis livres
U,V  - índices sobre S e T, respectivamente
K    - expressão booleana envolvendo comparações
      espaciais com duas variáveis livres
      a ser filtrada por U e V
R    - conjunto de pares de objetos em S×T
      que satisfazem J

Inicio
  R = ∅
  % Etapa 1: Filtragem por índice
  - Use os índices espaciais U e V sobre S e T
    para computar o conjunto R1 de pares de retângulos, u e
    v, tais que K(u,v)
  % Etapa 2: Refinamento
  - Para cada entrada (u,v) em R1, acesse as geometrias do
    par de objetos (s,t) correspondente a (u,v)
  - se J(s,t), acrescente (s,t) à resposta R
Fim
```

Figura 7.10 – Junção por índices.

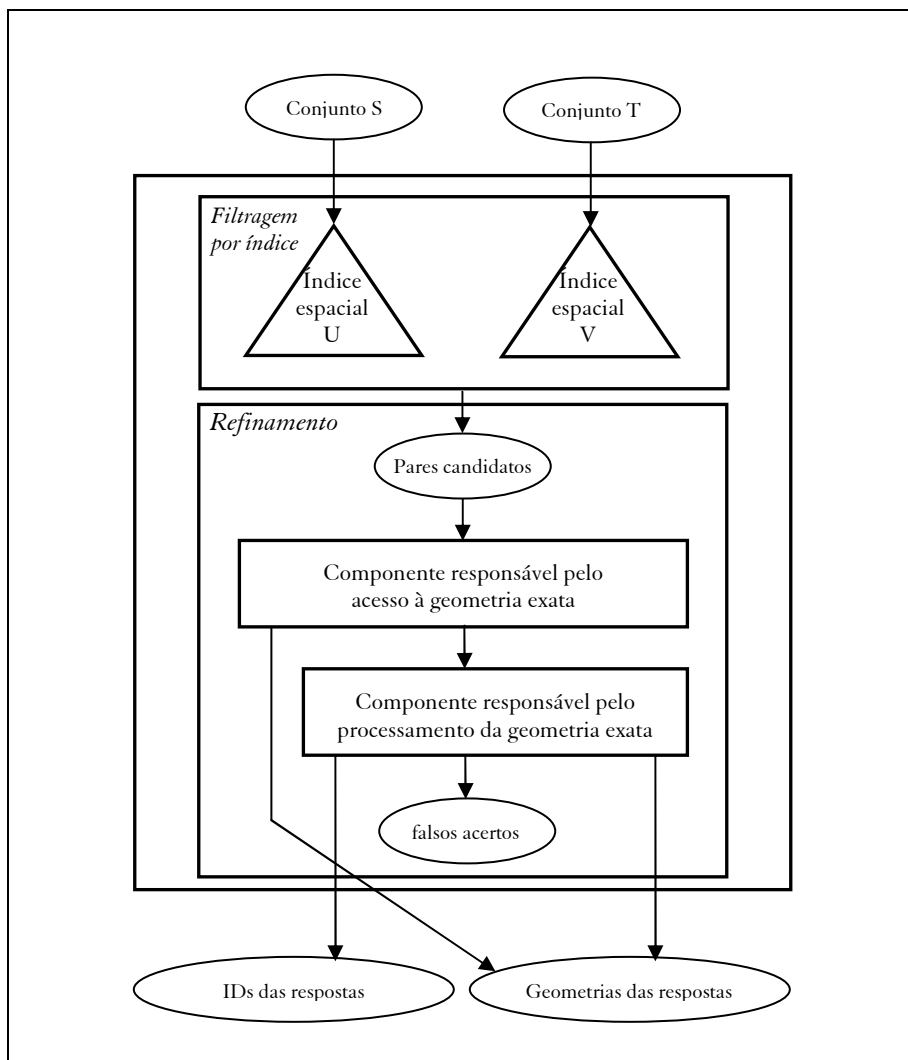


Figura 7.11 – Esquema da junção por índices. Fonte: adaptado de Kriegel et al., (1993).

FILTRO_DE_JUNCAO_POR_ARVORE_R(A,B,K;P)

- A,B- apontadores para as raízes de duas árvores-R sobre conjuntos de geo-objetos S e T (com a estrutura indicada na Figura 7.5).
 Supõe-se que A e B não sejam nulos.
 K - expressão booleana envolvendo comparações espaciais com duas variáveis livres que pode ser filtrada por árvores-R
 P - conjunto de pares de apontadores para objetos de S e T

Início

Determine a expressão K' companheira de K

P = \emptyset

Q = {(A,B)}

Enquanto Q $\neq \emptyset$ faça:

Início

Selecione (p,q) em Q, removendo-o de Q

Se p e q forem folhas,

Para cada entrada E em ls(p),

Para cada entrada F em ls(q),

Se K'(rem(E),rem(F)),

acrescente (pt(E),pt(F)) a P

Se p for folha e q for nó interior,

Para cada entrada F em ls(q),

Se K'(rem(p),rem(F)),

acrescente (p,pt(F)) a Q

Se q for folha e p for nó interior,

Para cada entrada E em ls(p),

Se K'(rem(E),rem(q)),

acrescente (pt(E),q) a Q

Fim

Fim

Figura 7.12 – Filtro de junção por árvore-R.

7.4.3 Computação de superposições de mapas temáticos

Um polígono é *simples* se e somente se não há um par de arestas não-consecutivas compartilhando um ponto. Um *polígono simples com furos* é um polígono simples ou um par (p,F) onde p é um polígono simples e F é um conjunto não vazio de polígonos simples com furos tais que, para todo $q \in F$, q está contido no interior de p e, para todo $q, q' \in F$, q e q' são disjuntos.



Figura 7.13 – Exemplos de polígonos (Kriegel et al., 1992).

Seja \wp o conjunto dos polígonos simples com furos. Um *conjunto de temas* é qualquer conjunto finito não vazio. Um *mapa temático vetorial* sobre um conjunto de temas T é uma função parcial $M: \wp \rightarrow T$ tal que, para todo $p, q \in \text{dom}(M)$, p e q são disjuntos. Denotaremos por $\mathfrak{S}[T]$ o conjunto dos mapas temáticos sobre um conjunto de temas T .

Esquemas de armazenamento, em memória secundária, para mapas temáticos vetoriais são variantes dos esquemas de armazenamento de polígonos onde cada polígono possui um atributo indicando o seu tema. Desta forma, índices para conjuntos de polígonos podem ser utilizados diretamente para indexar os domínios dos mapas temáticos.

Sejam T, U e V conjuntos de temas e $f: T \times U \rightarrow V$ uma função total. A *superposição de mapas temáticos induzida por f* é a função $\theta[f]: \mathfrak{S}[T] \times \mathfrak{S}[U] \rightarrow \mathfrak{S}[V]$ definida como $\theta[f](M, N) = O$ se e somente se

- $\text{dom}(O)$ é o conjunto dos polígonos com furos obtidos pela interseção dois-a-dois de um polígono em $\text{dom}(M)$ com um polígono em $\text{dom}(N)$;
- $O(p) = f(q, r)$ se e somente se p faz parte da interseção de q e r

Há três questões que devem ser levadas em consideração na computação da superposição $\theta[f](M,N)=O$ de dois mapas temáticos vetoriais: (1) M e N são normalmente grandes e, portanto, devem ser trazidos progressivamente para memória principal; (2) além da geometria, é necessário armazenar e recuperar o tema de cada polígono nos domínios de M e N; (3) a etapa de refinamento deve ser seguida por um pós-processamento para computar os polígonos no domínio do mapa resultante O e o valor associado a cada um destes polígonos.

Apesar destas questões, estratégias para computar $\theta[f](M,N)$ podem ser obtidas modificando-se as estratégias para junção espacial apresentadas anteriormente, considerando-se como predicado de junção a superposição de polígonos, e acrescentando-se uma etapa de pós-processamento como indicado acima. De fato, variantes da estratégia para computar $\theta[f](M,N)$ denominada *particionamento por varredura do plano* (Kriegel et al., 1992) podem ser derivadas da junção por particionamento e intercalação e da junção por índice.

7.5 Gerência da área de trabalho

7.5.1 Principais questões na gerência da área de trabalho

A definição de uma estratégia para gerência da área de trabalho em memória principal suscita algumas questões importantes, discutidas nesta seção, para que efetivamente contribua para um melhor desempenho do processador de consultas.

No que segue, usaremos o termo objeto para designar indistintamente, páginas físicas, objetos lógicos ou conjuntos destes.

Em primeiro lugar, há a questão do tipo de objeto – físico ou lógico - que será mantido na área de trabalho. Normalmente, os sistemas de gerência de banco de dados objeto-relacionais trabalham com páginas físicas na área de trabalho, enquanto que os sistemas orientados-a-objeto estritos podem trabalhar com páginas físicas, objetos lógicos ou conjuntos de objetos. Uma estratégia de gerência da área de trabalho é chamada de *semântica* quando trabalha com objetos lógicos ou conjuntos de objetos e utiliza características destes objetos ou conjuntos para implementar seus algoritmos.

Associada a esta primeira questão, está a decisão de que nível de granularidade a estratégia de gerência adota. A estratégia pode controlar páginas físicas ou objetos lógicos individualmente, ou trabalhar com *segmentos físicos*, compostos por páginas físicas e *segmentos semânticos*, compostos por objetos lógicos.

A terceira questão é a política de substituição dos objetos mantidos na área de trabalho. A maioria dos sistemas usa uma variante da estratégia usualmente chamada *LRU – Least Recently Used*, que propõe substituir o objeto que está há mais tempo na área de trabalho. Uma variante desta estratégia será discutida na seção seguinte.

Além destas, há a questão de criar políticas de antecipação que tragam objetos para a área de trabalho antes de serem solicitados por uma consulta. Naturalmente, uma boa política de antecipação pode acelerar o processamento de consultas, mas depende de um bom entrosamento com o processador de consultas.

7.5.2 Área de trabalho utilizando segmentos semânticos

Esta seção discute as características básicas de estratégias para gerência da área de trabalho que a organizem em segmentos lógicos, seguindo (Ren et al., 2003).

Ao processar uma nova consulta *Q*, o relacionamento entre o seu resultado e os segmentos lógicos na área de trabalho recai nos 4 casos mostrados na Figura 7.14. No caso 4, o resultado de *Q* está inteiramente contido na área de trabalho e nenhum novo objeto precisa ser trazido de memória secundária. Porém, informação adicional mantida junto aos segmentos lógicos poderá ser atualizada para refletir o processamento de *Q*. Em todos os outros casos, o resultado de *Q* deverá ser, em parte ou na sua totalidade, adicionado à área de trabalho e, novamente, informação adicional será gerada ou atualizada.

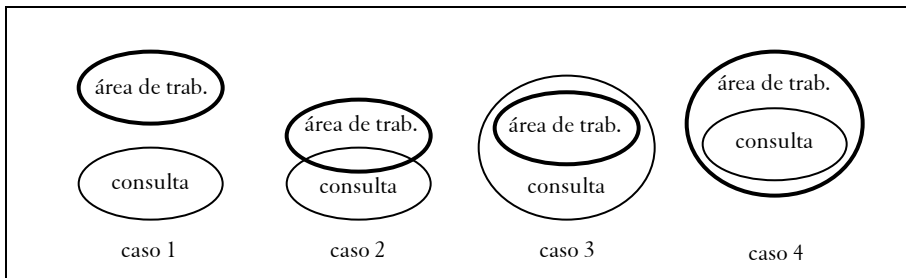


Figura 7.14 – Relacionamentos entre o resultado de uma nova consulta e a área de trabalho (Ren et al., 2003).

Mais precisamente, suponha que o conteúdo da área de trabalho possa ser representado pelo predicado C e seja R o resultado de uma consulta Q , com predicado de consulta Q_p .

A *política de inclusão* determina como tratar R em presença de C :

1. $(\neg Q_p \Rightarrow C)$: o resultado de Q deve ser *inteiramente incluído* na área de trabalho (Caso 1 da Figura 7.14).
2. $(Q_p \Rightarrow C)$: o resultado de Q deverá ser *ignorado* pois está inteiramente contido na área de trabalho (Caso 4 da Figura 7.14).
3. Nenhum dos casos acima: o resultado de Q deverá ser *parcialmente incluído* na área de trabalho (Casos 2 e 3 da Figura 7.14).

Usualmente, as implementações da política de inclusão testam o resultado de uma consulta contra cada segmento semântico em separado e tentam chegar a uma conclusão.

Suponha que R , o resultado da consulta, se superpõe a um segmento semântico S .

A *política de colapsamento* determina como tratar este tipo de superposição, escolhendo entre 3 opções:

1. *Colapsar totalmente* R e S , criando um novo segmento semântico $S_1 = R \cup S$.
2. *Colapsar parcialmente* R e S , criando dois novos segmentos semânticos, $S_1 = R$ e $S_2 = S - R$.
3. *Não colapsar* R e S , criando três novos segmentos semânticos, $S_1 = R - S$, $S_2 = S - R$ e $S_3 = S \cap R$.

Colapsar totalmente R e S reduz o número de segmentos lógicos na área de trabalho, o que simplifica a sua gerência e reduz o tempo de processamento do critério de inclusão. Por outro lado, esta política tende a criar segmentos lógicos muito grandes, embora só uma parte dos objetos no segmento seja efetivamente usada. Não colapsar R e S produz o efeito inverso pois tende a gerar segmentos lógicos com uma granularidade muito fina. Colapsamento parcial é normalmente a melhor escolha pois abre a possibilidade de balancear o tamanho dos segmentos lógicos.

A *política de substituição* determina qual segmento semântico deve ser removido da área de trabalho quando o resultado de uma nova consulta não puder ser acomodado.

Assumindo a política de colapsamento parcial, a política de substituição tradicional, LRU, de remover o segmento lógico que está há mais tempo na área de trabalho não é adequada pois os objetos em um segmento resultam das respostas de várias consultas e, portanto, possuem diferentes tempos de residência na área de trabalho. Neste caso, a política LRU pode ser substituída por uma versão mais sofisticada, chamada de *LRU dinâmica*, ou D-LRU (Ren et al., 2003). Brevemente, seja R novamente o resultado de uma consulta e S um segmento semântico que se sobrepõe a R. Suponha que S seja decomposto em $S_1=R-S$, $S_2=S-R$ e $S_3=S \cap R$. A política D-LRU mantém a data de S para o segmento S_2 e cria S_1 e S_3 com a data de execução de Q. Se a área de trabalho não pode acomodar o resultado R da consulta, os segmentos mais antigos são descartados, como na política LRU tradicional, até que haja espaço livre suficiente.

Há estratégias mais sofisticadas que utilizam uma noção de distância semântica para selecionar o objeto a substituir, como a apresentada em (Dar et al., 1996).

7.5.3 Exemplo de estratégia de gerência da área de trabalho

Esta seção exemplifica o uso dos conceitos apresentados na seção anterior para construir uma estratégia de gerência da área de trabalho que melhore o desempenho de aplicações de exploração visual dos dados,

típicas de sistemas de informação geográfico. Esta seção segue basicamente (Doshi et al., 2003).

Suponha que, na aplicação de exploração visual dos dados, o usuário interaja diretamente com uma *janela de visualização* através das operações usuais de *aproximação, afastamento, deslocamento e enquadramento (zoom-in, zoom-out, panning, fitting)*. Estas operações geram então consultas aos dados.

A área de trabalho armazena segmentos semânticos, definidos por conjuntos de geo-objetos. Cada segmento semântico está associado ao r.e.m. dos objetos nele contidos.

O resto desta seção discute, em detalhe, políticas de antecipação de acesso aos geo-objetos. Em uma aplicação de exploração visual dos dados, uma política de antecipação é especialmente interessante por duas razões: (1) o usuário passa considerável tempo analisando os dados visualizados, deixando o processador e o disco ociosos; logo, é possível utilizar o processador e o disco para pré-carregar dados sem afetar a exploração dos dados; (2) dadas as características das operações, é possível prever qual será o próximo movimento do usuário.

Em geral, uma política de antecipação deve trabalhar gradativamente, à medida que descobre novas características do comportamento do usuário ou dos dados acessados. No início do processo de exploração visual, pouca ou nenhuma informação está disponível. Com o tempo, mais características do comportamento do usuário ou dos dados acessados tornam-se evidentes, melhorando a qualidade dos dados trazidos antecipadamente para a área de trabalho.

Mais especificamente, para aplicações de exploração visual dos dados, assumimos que a política de antecipação deve detectar se o usuário: (1) tende a manter a direção da operação de deslocamento (da janela de visualização) ou alterá-la; (2) se os dados sob análise visual possuem regiões de interesse às quais o usuário tende a voltar. Baseadas nestas suposições, podemos definir as seguintes políticas de antecipação (Doshi et al., 2003): P1 – aleatória; P2 – direção de deslocamento; P3 – foco do usuário.

A política de antecipação aleatória, ilustrada na **Figura 7.157.15 (a)**, escolhe aleatoriamente a direção do deslocamento da janela de visualização, dando pesos iguais às quatro possíveis direções. Esta política naturalmente se aplica quando não é possível detectar características úteis no comportamento do usuário ou do acesso aos dados.

A política de antecipação baseada na direção de deslocamento, ilustrada na **Figura 7.15(b)**, é semelhante à anterior, exceto que dá maior peso a uma direção, se o usuário moveu a janela de visualização duas vezes seguidas naquela direção.

Já a política de antecipação baseada no foco do usuário é mais sofisticada e utiliza indicações sobre o próximo deslocamento e sobre regiões de interesse do usuário. Uma região é considerada como uma *região de interesse* em uma sessão de trabalho quando o usuário a visita mais do que k vezes, onde k é um parâmetro da política. Uma região de interesse é descrita por uma data e um retângulo. Enquanto nenhuma região de interesse estiver próxima da janela de visualização, esta política trabalha de forma semelhante à política baseada na direção de deslocamento. Quando alguma região de interesse torna-se próxima à janela de visualização, esta política antecipa a carga dos objetos na direção da região de interesse, e não na direção dos últimos deslocamentos. Esta política reflete então a intuição de que o usuário provavelmente governa a visualização em direção à região de interesse e lá permanecerá por algum tempo.

Por fim, a política de substituição combina a estratégia de D_LRU com a política de antecipação. Assim, os segmentos semânticos mais antigos e fora da direção indicada pela política de antecipação devem ser descartados primeiro, até que seja liberada memória suficiente para acomodar os novos a serem trazidos para memória principal.

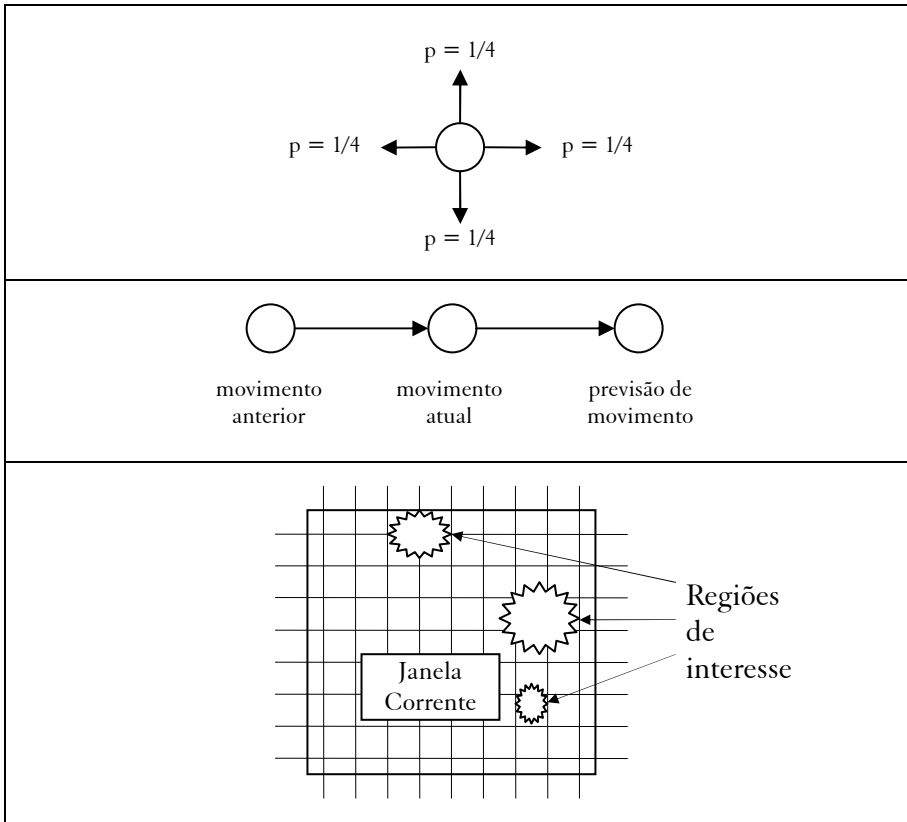


Figura 7.15 – Relacionamentos entre o resultado de uma nova consulta e a área de trabalho (Ren et al., 2003).

7.6 Gerência de transações

7.6.1 Transações em bancos de dados geográficos

Em aplicações convencionais de banco de dados, um usuário modifica os dados através de uma seqüência de operações elementares que devem ser executadas como um todo. As operações são tipicamente curtas e envolvem um volume pequeno de dados. Por exemplo, uma transferência de fundos é implementada através de duas atualizações, em contas bancárias distintas, que devem ser ambas executadas ou ambas canceladas, para evitar erros.

Esta situação configura uma *transação*, ou seja, uma seqüência de operações que o sistema de gerência de banco de dados deve processar até o fim, ou garantir que o banco de dados não reflita nenhuma delas, desfazendo aquelas que tenham sido total ou parcialmente executadas. Além disto, o sistema deve processar as operações sem interferência de outras transações e garantir que, se as operações começam em um estado consistente do banco de dados, elas terminam em um estado também consistente. Mecanismos para garantir estas propriedades foram amplamente investigados no contexto de aplicações convencionais.

Os usuários de bancos de dados geográficos também executam transações desta natureza, principalmente ao modificar os atributos convencionais de objetos. Porém, eles podem apresentar um padrão de comportamento muito diferente, mais próximo do encontrado em ambientes de desenvolvimento de software, projeto de circuitos VLSI e criação de documentos hipermídia, entre outros. Nestes ambientes, uma equipe de usuários trabalha cooperativamente para produzir uma nova configuração dos objetos. Cada usuário cria uma parte de um todo em várias etapas, possivelmente trabalhando sobre uma situação já existente, que não pode ser congelada. Além disto, a equipe pode criar configurações alternativas dos objetos, ou seja, modificações que não necessariamente serão efetivadas.

O conjunto de modificações criado pela equipe de usuários configura uma *transação longa e aninhada*. Neste contexto, o termo transação é freqüentemente utilizado como sinônimo de *sessão de trabalho*. A

transação é longa pois normalmente o trabalho da equipe se desenvolve por muito tempo, possivelmente semanas. Ela é cooperativa porque envolve modificações criadas por usuários distintos, de forma coordenada. Finalmente, a transação pode conter coleções completas de modificações organizadas em *subtransações aninhadas*. Informalmente, uma transação T contém uma subtransação aninhada T' quando um parte das operações de T é agrupada e tratada como uma transação T' de tal forma que as operações em T' possam ser desfeitas ou refeitas pelo sistema como um todo, ou tornadas condicionalmente visíveis a outros usuários.

Por exemplo, considere um banco de dados geográfico contendo dados sobre topografia, vegetação, áreas de preservação ambiental e rede viária de uma região. Suponha que uma equipe de dois engenheiros esteja desenvolvendo o traçado de um novo oleoduto, cada um trabalhando em trechos diferentes e contíguos do oleoduto. Este projeto envolve então trabalho cooperativo, de longa duração, sobre os objetos no banco, configurado como uma única transação longa, contendo duas subtransações aninhadas, uma para cada membro da equipe. Este exemplo será elaborado de forma um pouco diferente na Seção 7.6.3.

7.6.2 Mecanismos para implementação de transações

Esta seção discute alguns mecanismos genéricos propostos para implementação de transações com as características descritas na seção anterior, incluindo tratamento de versões. A descrição segue (Dias et al., 1995; Soares et al., 1993). Esta seção é opcional, podendo o leitor passar diretamente para a Seção 7.6.3.

Paralelamente aos mecanismos sofisticados para manipulação de dados, descritos abaixo, o usuário poderá ainda manipular diretamente tais dados, para cobrir casos simples como, por exemplo, atualização do valor de um atributo convencional.

Em geral, um ambiente para trabalho cooperativo deve permitir tanto compartilhamento de conjuntos de dados entre usuários quanto a definição de conjuntos privativos a um grupo de usuários. Para tanto, supomos que os dados estão organizados em uma hierarquia de bancos de dados e que cada dado só pertence a um destes bancos. Se B' é filho de

B na hierarquia, dizemos que B' é um banco *subordinado* a B. (Note que a hierarquia possui um número arbitrário de níveis). Cada banco está associado a um grupo de usuários com permissão para realizar um certo conjunto de operações sobre os dados no banco ou em bancos subordinados a ele. Esta hierarquia de bancos reflete o fato de transações conterem outras transações aninhadas. Assim, um grupo de usuários poderá criar um banco de dados para conter os dados com que irá trabalhar e, recursivamente, criar outros bancos subordinados a este, associando-os a transações aninhadas conduzidas por subgrupos de usuários.

Ortogonalmente a estes conceitos, consideramos que um dado pode estar em um de três estados: *pronto*, *trabalho* ou *obsoleto* (respectivamente, *committed*, *uncommitted* ou *obsolete*). Um dado d pode ainda estar associado a um conjunto de outros dados, tratados como suas *versões*, e organizados sob forma de uma árvore, onde d é a raiz. Esta árvore é mantida de tal forma que, se uma versão está pronta, todos os seus ancestrais na árvore também estão prontos; e, se uma versão é obsoleta, todos os seus descendentes também são obsoletos. Os dados são manipulados através do elenco de operações descritas a seguir.

A operação *update* permite atualizar um dado em trabalho, sem alterar o seu estado.

A operação *commit* transforma em pronto um dado em trabalho.

A operação *delete* remove um dado d de um banco B. Se d for um dado em trabalho, ele é efetivamente destruído; se d for um dado pronto, ele é tornado obsoleto. O usuário também pode remover um banco B, provocando a remoção de todos os dados e bancos subordinados a B, recursivamente. O sistema se encarregará de manter dados obsoletos apenas enquanto eles forem referenciados por outros dados.

Há três operações para criação de dados: *create* cria um dado d' completamente novo em um banco B'; *create-version* cria uma nova versão d' de um dado d pronto ou em trabalho no mesmo banco de dados B' que contém d; *check-out* cria um novo dado d' em B' como uma versão de um dado pronto d existente em B, onde B' é um banco subordinado a B. O usuário também poderá utilizar esta operação para criar em bloco versões de um conjunto de dados. Note que o usuário não pode mover d

de B para B', mas apenas usar a operação *check-out* para criar uma versão d' de d em B'. Em todos estes três casos, d' é considerado um dado em trabalho após a sua criação.

A operação *check-in* permite mover um dado pronto d' de um banco B' para o banco B a que B' se subordina. O usuário também poderá utilizar esta operação para mover em bloco um conjunto de dados de um banco para outro. Há duas variantes desta operação. A primeira delas, *check-in de adição*, de fato move d' de B' para B, exceto se d' for uma versão de um dado d existente em B e d' não tiver sido modificado, caso em que d' é descartado, evitando assim a criação de uma réplica de d em B. A segunda, *check-in por substituição*, move d' de B' para B, se d' for um novo dado, ou move d' de B' para B e remove o dado d existente em B (através da operação *delete*, se d' for uma versão de d e d' tiver sido modificado). Esta última operação corresponde portanto à tradicional operação de atualização.

Estas operações mantêm as árvores de versões consistentes através de ações colaterais. Assim, a operação *commit* aplicada a um dado se propaga recursivamente para todos os seus ancestrais na árvore, transformando-os também de dados em trabalho para prontos. A operação *delete* aplicada a um dado se propaga recursivamente para todos os seus descendentes na árvore, destruindo-os ou tornando-os obsoletos. Este efeito pode ser provocado também por uma operação de *check-in por substituição* ao chamar implicitamente uma operação *delete*.

Note que as árvores de versões cruzam a hierarquia de bancos de dados. Portanto, quando um usuário executa uma operação de *commit* sobre um dado em um banco B, ele poderá afetar um dado que pertence ao banco a que B se subordina, e assim recursivamente. Assim, o usuário deverá possuir permissão para executar *commit* também sobre dados nestes outros bancos. Porém, a operação *delete* não causa problemas. Por definição, se o usuário possui permissão para aplicar a operação de *delete* sobre dados de B, ele também possui permissão para aplicar *delete* a dados em bancos subordinados a B, recursivamente. A mesma observação se aplica à operação de *check-in por substituição*. Assim, como efeito colateral de uma operação sobre um dado d, um usuário poderá interferir com bancos de dados de outros usuários que contêm versões de d.

Para minimizar este problema, podemos lançar mão de mecanismos de controle de acesso, permitindo o bloqueio de dados em três modos: *compartilhado*, *exclusivo* e *versão-exclusivo*. Quando um usuário (ou grupo de usuários) bloqueia um dado em modo compartilhado, ele indica uma potencial intenção de substituí-lo por outra versão; em modo exclusivo, ele proíbe outros usuários de substituí-lo por outra versão; em modo versão-exclusivo, ele proíbe outros usuários de criar versões do dado.

Este recurso se integra às operações descritas anteriormente. Em uma operação de *check-out* que cria um novo dado d' em B' como uma versão de um dado d em B ; d é bloqueado em modo compartilhado para o usuário que executa a operação. Ao executar um *check-in por substituição* sobre d' , se d' tiver sido modificado, o bloqueio de d é escalonado para o modo exclusivo e os outros usuários que possuam bloqueios compartilhados sobre d têm seus bloqueios descartados. Isto os impede de retornar versões de d , forçando-os a refazer o *check-out*, agora sobre d' (dado que substituiu d).

Esta discussão é simplificada, pois não explora em detalhe as referências cruzadas entre dados, quer seja porque um dado é uma componente de outro, quer seja porque há um relacionamento explícito entre ambos. Também não considera a criação de versões ou o bloqueio de partes de um conjunto de dados. Esta extensão é particularmente importante pois geo-objetos ou geo-campos freqüentemente cobrem áreas geográficas maiores do que a região de interesse do usuário. A seção seguinte apresenta um exemplo de modificação de um banco de dados geográfico que utiliza extensões das operações acima descritas cobrindo este ponto.

7.6.3 Exemplo de transação sobre banco de dados geográfico

Considere um banco de dados geográfico B contendo, para o Estado do Rio de Janeiro:

- um geo-campo T capturando a sua topografia;
- uma coleção de geo-objetos chamada de **ÁREAS-DE-PRODUÇÃO**, cujos elementos descrevem as áreas de proteção, descritas por uma representação por subdivisão planar P ; e

- duas coleções de geo-objetos, REDE-VIÁRIA e OLEODUTOS, com a descrição das estradas e oleodutos, cujas localizações encontram-se em uma representação complexa V.

Suponha agora que uma equipe tenha sido designada para elaborar uma proposta para um oleoduto do terminal da Petrobrás no Município de Angra dos Reis para o Porto de Sepetiba, sem cruzar áreas de proteção ambiental e aproveitando ao máximo os cortes já feitos para as estradas. O trabalho da equipe configura uma transação longa e cooperativa, construída interativamente com descrito a seguir.

Inicialmente a equipe define uma janela R cobrindo toda a região de trabalho. Em seguida, define o banco de dados privado B' e cria, através da operação *check-out*, versões T_R e V_R dos objetos T e V apenas para a região R e uma versão OL de OLEODUTOS. Suponha que a equipe tenha permissão apenas para consultar as coleções ÁREAS DE PROTEÇÃO e REDE-VIÁRIA e a representação P.

Em seguida, a equipe define o oleoduto, inserindo um novo geo-objeto O em OL, e alterando V_R para incluir a representação de O.

A equipe também modifica o geo-campo T_R para indicar os novos cortes necessários à passagem do oleoduto (mas não modifica P, por restrições de projeto).

Ao término, a equipe introduz o resultado do projeto no banco original B através da operação de *check-in por substituição*, estendida para tratar de versões que se referem apenas a uma parte da área geográfica coberta pelos objetos.

Em paralelo, uma segunda equipe poderá estar trabalhando no traçado de uma nova estrada no Município de Angra dos Reis, com restrições semelhantes às anteriores: a estrada não poderá cruzar áreas de proteção ambiental e deverá aproveitar ao máximo os cortes já feitos para oleodutos.

Novamente, a equipe define uma janela S cobrindo sua região de trabalho e define o banco de dados privado E, criando, através da operação *check-out*, versões T_S e V_S dos objetos T e V apenas para a região S e uma versão RV de REDE-VIÁRIA.

De forma semelhante, esta equipe define a estrada, inserindo um novo geo-objeto E na coleção RV, alterando $\$V_S\$$ para incluir a representação de E, e modificando o geo-campo \bar{T}_S para indicar novos cortes necessários à passagem da nova estrada.

Ao término, a equipe tentará introduzir o resultado deste segundo projeto no banco original B, através da operação de *check-in por substituição*.

Dois situações poderão ocorrer. Primeiro, se as regiões R e S forem disjuntas, então o trabalho de uma equipe não afeta diretamente o da outra e a alteração do banco B ocorrerá normalmente. Porém, a implementação da operação de *check-in por substituição* deverá ser suficientemente sofisticada para alterar V e T apenas para a região S de tal forma que não destrua as modificações efetuadas nestes campos pela equipe que criou o oleoduto.

Segundo, se as regiões R e S não forem disjuntas, o trabalho de uma equipe potencialmente afeta o da outra. À guisa de ilustração, façamos uma análise mais detalhada desta situação. A primeira equipe altera V acrescentando a representação de O e a segunda acrescentando a representação de E. Portanto, se o sistema versionar as componentes de V, e não V como um todo, será simples implementar a operação de *check-in por substituição* de tal forma que o trabalho de uma equipe não interfira com o da outra do ponto de vista de V: basta reconhecer que ambas as equipes simplesmente criaram novas componentes de V. Argumento semelhante se aplica a T, exceto que, neste caso, como T é um geo-campo, será necessário analisar a representação utilizada para T. Por exemplo, se a representação for por isolinhas, será necessário versionar as isolinhas em si; assim uma equipe não interferirá com a outra se alterarem isolinhas distintas.

Naturalmente, toda esta discussão sobre a operação de *check-in por substituição* é afetada pelos mecanismos de controle de acesso, que também deverão ser revistos para acomodar os refinamentos baseados no uso de regiões ou de componentes.

7.7 Leituras suplementares

Resumos dos principais aspectos de implementação de sistema de gerência de bancos de dados geográficos podem ser encontrados em referências gerais (Guting, 1994; Medeiros e Pires, 1994; Shekhar et al., 1999; Rigaux et al, 2001; Shekhar, 2002). Descrições de extensões dos principais sistema de gerência de bancos de dados para tratar dados geográficos encontram-se em (Ravada e Sharma, 1999; David, 2001).

Políticas específicas para otimização de consultas espaciais podem ser encontradas em referências seminais sobre o assunto (Aref e Samet, 1991; Kriegel et al., 1993; Brinkhoff et al., 1993; Brinkhoff et al., 1994)). Em particular, a idéia básica de processar uma consulta espacial em vários passos, através de aproximações da geometria dos objetos, foi introduzida em (Kriegel et al., 1993). Propostas de *benchmarks* para consultas espaciais podem ser encontrados em (Stonebraker et al. 1993; Cifferi, 1995). Uma biblioteca de algoritmos para processamento de consultas espaciais é apresentada em (Bercken et al. 2000). A questão do processamento de consultas para objetos representados em alta resolução é discutida em (Kriegel et al., 2003).

Há bem poucas referências sobre gerência de transações para o caso específico de sistemas de gerência de bancos de dados geográficos. Porém, o padrão de transações para estas aplicações é semelhante às chamadas *transações longas e aninhadas* (Lynch, 1983; Weikum, 1991), típicas de ambientes de desenvolvimento de software, ambientes de projeto de VLSI e hipertexto, entre outros. Um modelo semântico, genérico para transações, que se aplica também a bancos de dados geográficos, foi proposto em (Brayner et al., 1999). Um modelo de transações recente e específico para bancos de dados geográficos encontra-se (Kadri-Dahmani e Osmani, 2003).

A discussão sobre processamento de consultas espaciais apresentada neste capítulo aborda apenas aplicações convencionais envolvendo dados geográficos. Há, no entanto, uma vasta gama de aplicações de outra natureza, que necessitam revisar a semântica das consultas e, conseqüentemente, as técnicas de processamento de consultas.

Podemos apontar aplicações que necessitam: tratar objetos espaço-temporais (Kim et al., 2002); trabalhar com dados geográficos

representando redes, onde a noção de espaço Euclidiano não é adequada (Papadias et al., 2003); considerar consultas dependentes de localização em ambientes para computação móvel (Ayse et al., 2001; Zhang et al., 2003; Manical et al., 2004); e processar objetos móveis (Porkaew et al., 2001 ; Chon et al., 2002).

Referências

- ABOULNAGA, A.; NAUGHTON, J. F. Accurate estimation of the cost of spatial selections. In: 16th International Conference on Data Engineering. San Diego, CA, USA, 2000. p. 123-134.
- AREF, W.; SAMET, H. Optimization for Spatial Query Processing. In: 17th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., Barcelona, Spain, 1991. p. 81-90.
- AYSE, Y.; DUNHAM, H. M.; KUMAR, V. M. Location dependent query processing. In: 2nd ACM international workshop on Data engineering for wireless and mobile access. Santa Barbara, CA, USA, 2001. p. 47-53.
- BERCKEN, J.; BLOHSFELD, B.; DITTRICH, J.-P.; KRAMER, J.; SCHAFER, T.; SCHNEIDER, M.; SEEGER, B. XXL - A Library Approach to Supporting Efficient Implementations of Advanced Database Queries. In: 27th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001. p. 39-48.
- BRAYNER, A.; HARDER, T.; RITTER, N. Semantic Serializability: A Correctness Criterion for Processing Transactions. **Data & Knowledge Engineering**, v. 31, n.1, p. 1-24, 1999.
- BRINKHOFF, T.; HORN, H.; KRIEGEL, H.-P.; SCHNEIDER, H. A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems. In: Third International Symposium on Advances in Spatial Databases. **Lecture Notes In Computer Science**. Springer-Verlag London, UK, 1993a. p. 357-376.
- BRINKHOFF, T.; KRIEGEL, H.-P.; SCHNEIDER, H.; SEEGER, B. GENESYS: A System for Efficient Spatial Query Processing. In: 1994 ACM SIGMOD International Conference on Management of Data. Minneapolis, MI, USA, 1994a. p. 519.
- BRINKHOFF, T.; KRIEGEL, H.-P.; SCHNEIDER, R.; SEEGER, B. Multi-step Processing of Spatial Joins. In: 1994 ACM SIGMOD International Conference on Management of Data. Minneapolis, USA, 1994b. p. 197-208.

- BRINKHOFF, T.; KRIEGEL, H.-P.; SEEGER, B. Efficient Processing of Spatial Joins. In: 1993 ACM SIGMOD International Conference on Management of Data. Washington, DC, USA, 1993b. p. 237-246.
- CHON, D. H.; AGRAWAL, D.; EL ABBADI, A. Query Processing for Moving Objects with Space-Time Grid Storage Model. In: Third International Conference on Mobile Computing. 2002. p. 121.
- CIFFERI, R. Um Benchmark Voltado a Análise de Desempenho de Sistemas de Informações Geográficas. Campinas, SP, Brasil: UNICAMP, 1995. Departamento de Ciência da Computação, 1995.
- DAR, S.; FRANKLIN, M. J.; JONSSON, B. T.; SRIVATAVA, D.; TAN, M. Semantic Data Caching and Replacement. In: 22th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., 1996. p. 330-341.
- DAVID, W. A. DB2 Spatial Extender - Spatial data within the RDBMS. In: 27th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001. p. 687-690.
- DIAS, E.; GRANADO, S.; MAGALHÃES, G. Uso de Versões na Garantia de Consistência em Ambientes Mistos de Projetos e Operação. In: Simpósio Brasileiro de Banco de Dados. 1995. p. 321-334.
- DOSHI, P. R.; RUNDENSTEINER, E. A.; WARD, M. O. Prefetching for Visual Data Exploration. In: Eighth International Conference on Database Systems for Advanced Applications. 2003. p. 195.
- GUNTHER, O.; BECKER, L.; HINRICHS, K.; FINKE, U. Efficient computation of spatial joins. In: 9th International Conference on Data Engineering. Vienna, Austria, 1993. p. 50-59.
- GUTING, R. An Introduction to Spatial Database Systems. **VLDB Journal**, v. 3, n.4, p. 357-399, 1994.
- KADRI-DAHMANI, H.; OSMANI, A. Updating Data in GIS: How to Maintain Database Consistency? In: Enterprise Information Systems IV. Kluwer Academic Publishers, Ciudad Real, Spain, 2003. p. 163-169.
- KIM, H. D.; RYU, H. K.; PARK, H. C. Design and implementation of spatiotemporal database query processing system. **Journal of Systems and Software**, v. 60, n.1, p. 37-49, 2002.
- KRIEGEL, H.-P.; BRINKHOFF, T.; SCHNEIDER, R. An Efficient Map Overlay Algorithm Based on Spatial Access Methods and Computational Geometry. In: **Geographic Database Management Systems**. Springer-Verlag, Capri, 1992. p. 194-211.

- KRIEGEL, H.-P.; BRINKHOFF, T.; SCHNEIDER, R. Efficient Spatial Query Processing in Geographic Database Systems. **Data Engineering Bulletin**, v. 16, n.3, p. 10-15, 1993.
- KRIEGEL, H.-P.; PFEIFLE, M.; POTKE, M.; SEIDL, T. Spatial Query Processing for High Resolutions. In: 8th International Conference on Database Systems for Advanced Applications (DASFAA'03). Kyoto, Japan, 2003. p. 17.
- KRIEGEL, H.-P.; SCHIWIETZ, M.; SCHNEIDER, R.; SEEGER, B. Performance comparison of point and spatial access methods. In: First Symposium on Design and Implementation of Large Spatial Databases. Lecture Notes in Computer Science. Springer-Verlag, Santa Barbara, CA, USA, 1990. p. 89-114.
- LO, M.-L.; RAVISHANKAR, C. V. Spatial hash-joins. In: 1996 ACM SIGMOD International Conference on Management of Data. Montreal, Quebec, Canada, 1996. p. 247-258.
- LYNCH, N. A. A New Correctness Criterion for Database Concurrency Control. **ACM Transactions on Database Systems**, v. 8, n.4, p. 484-502, 1983.
- MANICAL, H.; CAMARGO, S. M.; CIFERRI, A. D. C.; CIFERRI, R. R. Processamento de Consultas Espaciais Baseado em Cache Semântico Dependente de Localização. In: VI Simpósio Brasileiro de Geoinformática – GeoInfo 2004. Campos de Jordão, SP, 2004. p.
- MEDEIROS, C. B.; PIRES, F. Databases for GIS. **ACM SIGMOD Record**, v. 23, n.1, p. 107-115, 1994.
- PAPADIAS, D.; ZHANG, J.; MAMOULIS, N.; Y., T. Query Processing in Spatial Network Databases. In: 29th Very Large Data Base Conference. Berlin, Germany, 2003. p.
- PATEL, J. M.; DEWITT, D. J. Partition based spatial-merge join. In: 1996 ACM SIGMOD International Conference on Management of Data. Montreal, Quebec, Canada, 1996. p. 259-270.
- PORKAEW, K.; LAZARIDIS, I.; MEHROTRA, S. Querying Mobile Objects in Spatio-Temporal Databases. In: 7th International Symposium on Advances in Spatial and Temporal Databases. Lecture Notes In Computer Science. Springer-Verlag, Redondo Beach, CA, USA, 2001. p. 59-78.
- PREPARATA, F. P.; SHAMOS, M. I., eds., 1985, **Computational Geometry: An Introduction**: New York, NY, Springer-Verlag.

- RAVADA, S.; SHARMA, J. Oracle8i Spatial: Experiences with Extensible Databases. In: 6th International Symposium on Advances in Spatial Databases. Springer-Verlag, London, UK, 1999. p. 355-359.
- REN, Q.; DUNHAM, M. H.; KUMAR, V. Semantic Caching and Query Processing. **IEEE Transactions on Knowledge and Data Engineering**, v. 15, n.1, p. 192-210, 2003.
- RIGAUX, P.; SCHOLL, M.; VOISARD, A. **Spatial Databases with Application to GIS**. San Francisco: Morgan Kaufmann Publishers Inc, 2001.
- SHEKHAR, S.; CHAWLA, S.; RAVADA, S.; FETTERER, A.; LIU, X.; LIU, C. T. Spatial Databases: Accomplishments and Research Needs. **IEEE Transactions on Knowledge and Data Engineering**, v. 11, n.1, p. 45-55, 1999.
- SOARES, L.; CASANOVA, M.; RODRIGUES, N. Um Modelo Conceitual Hipermídia com Nós de Composição e Controle de Versões. In: VII Simpósio Brasileiro de Engenharia de Software. 1993. p. 365-381.
- STONEBRAKER, M.; FREW, J.; GARDELS, K.; MEREDITH, J. The Sequoia 2000 Benchmark. In: 1993 ACM SIGMOD International Conference on Management of Data. Washington, D.C., USA, 1993. p. 2-11.
- WEIKUM, G. Principles and Realization Strategies of Multilevel Transaction Management. **ACM Transactions on Database Systems**, v. 16, n.1, p. 132-180, 1991.
- ZHANG, J.; ZHU, M.; PAPADIAS, D.; TAO, Y.; LEE, D. L. Location-based spatial queries. In: 2003 ACM SIGMOD International Conference on Management of Data. San Diego, CA, USA, 2003. p. 443-454.