Métodos de acesso espaciais e indexação espacial

Clodoveu Davis

Métodos de acesso

- *Métodos de acesso* são os procedimentos empregados pelo gerenciador de banco de dados com o objetivo de acelerar a localização e a recuperação de algum dado

Métodos de acesso

- Em bancos de dados convencionais, existem diversos tipos de métodos de acesso
- O SGBD utiliza aquele método que parece ser o mais adequado para a solução de cada operação
  - *Otimização de consultas*

Métodos de acesso

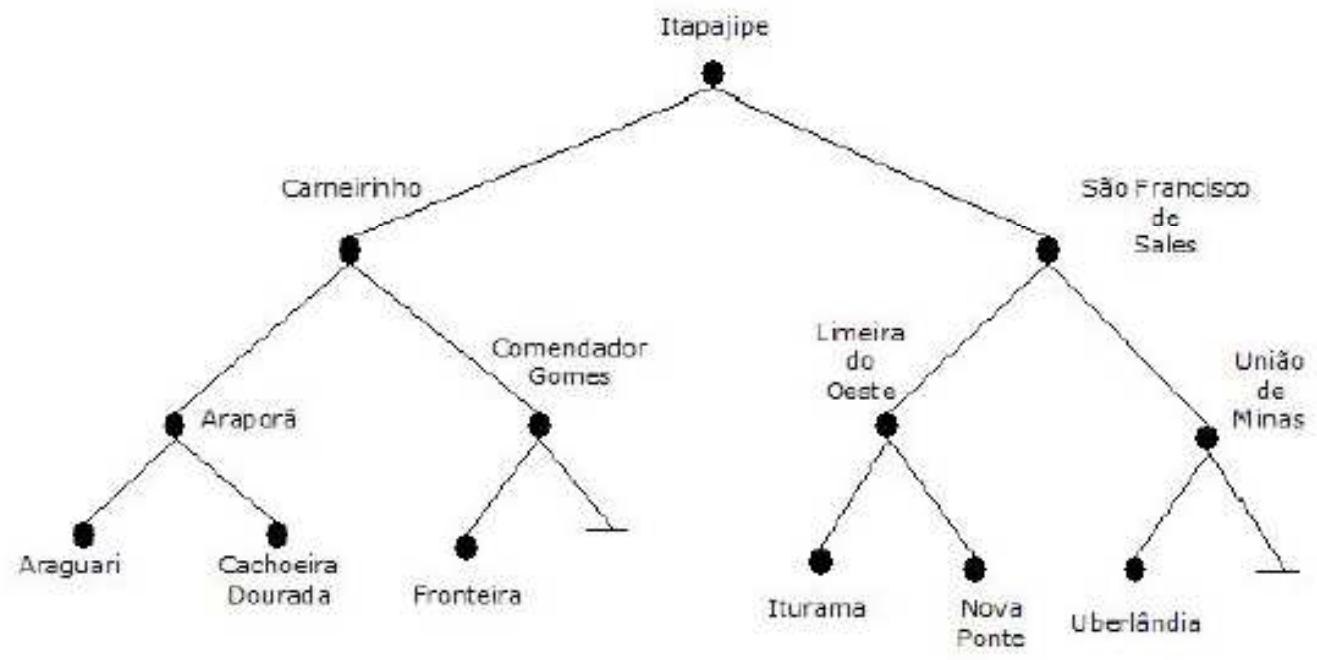
- A escolha do método é feita com base em
  - Características estruturais da tabela e de índices porventura existentes: *otimização heurística*
  - Cálculos aproximados do custo computacional de processamento da consulta: *otimização baseada em custos*

Métodos de acesso

- Em BD convencionais, a criação de *índices* é opcional, ficando a cargo do administrador do banco de dados
- Cada índice provê uma maneira rápida de buscar dados dentro de uma tabela
  - Existem diferentes tipos de índices, cada qual adequado a um tipo de atributo ou situação

Métodos de acesso

- Exemplo: árvore binária balanceada



Fonte: Casanova et al 2005, fig 6.1

Métodos de acesso

- Exemplo: árvore B+

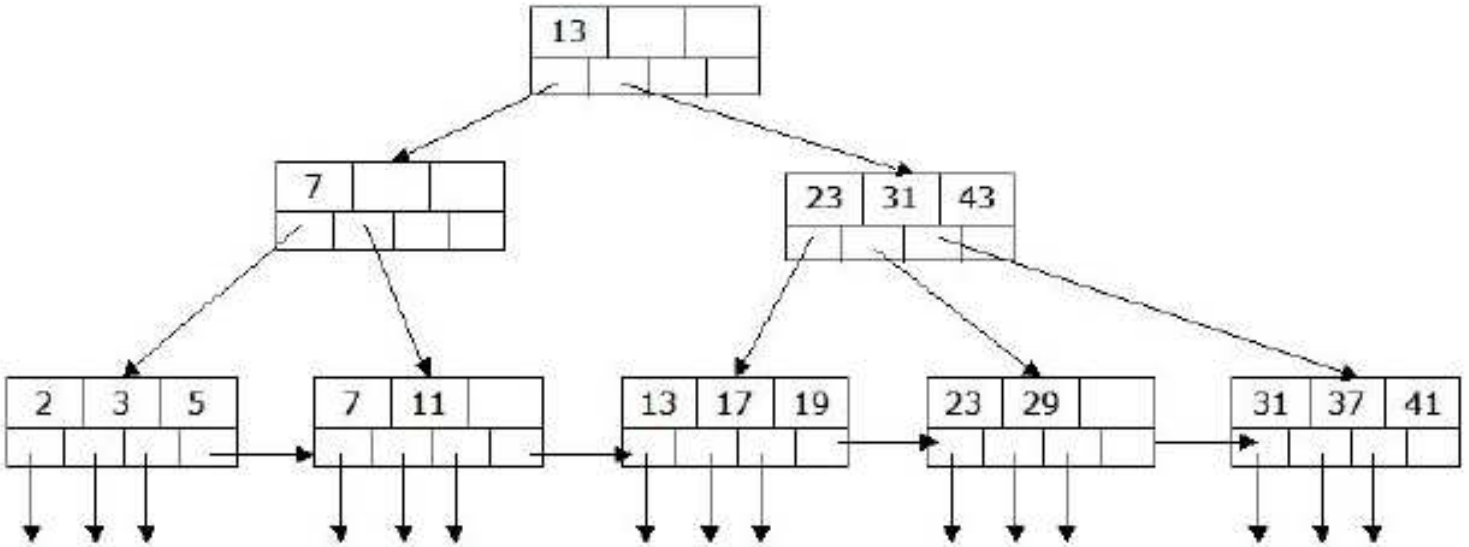


Figura 6.2 – Índice unidimensional B-Tree. Fonte: adaptada de Garcia-Molina et al., (2001).

Métodos de acesso espaciais

- Os métodos tradicionais se aplicam a chaves alfabéticas ou numérica simples, unidimensionais
- Para recuperar dados espaciais, seria necessário ter *métodos de acesso multidimensionais*

Métodos de acesso multidimensionais

- Ex: encontrar empregados nascidos entre 1950 e 1955 cujo salário está entre R$3000 e R$5000 mensais
  - Equivale ao problema geométrico de encontrar um ponto em um retângulo

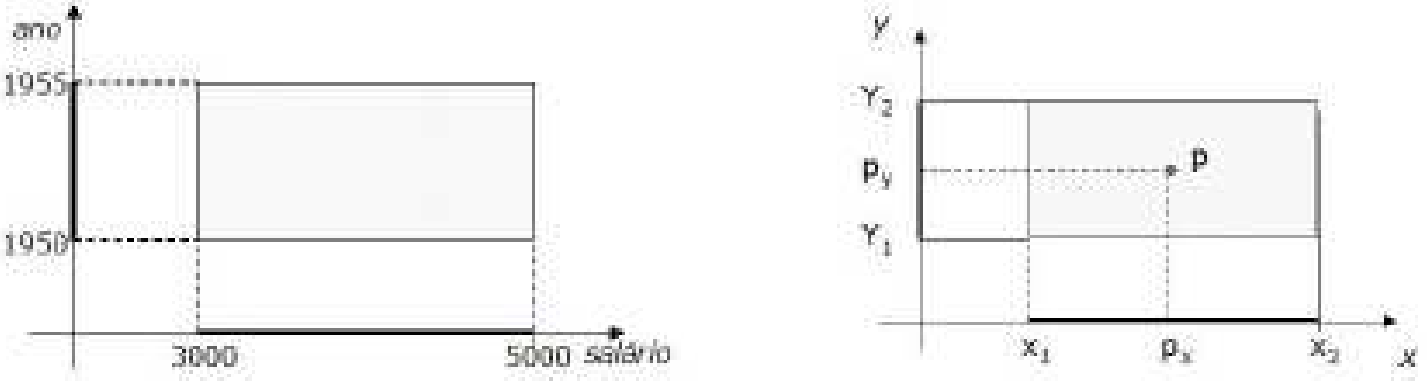


Figura 6.4 – Busca em dois atributos (a) Interpretação geométrica da busca (b).

Métodos de acesso espaciais

- Em BD espaciais, o SGBD precisa contar com métodos de acesso especificamente voltados para as características dos dados de representação
- Os métodos tradicionais são também usados, mas apenas sobre os dados alfanuméricos

Métodos de acesso espaciais

- Métodos de acesso espaciais são estruturas de dados auxiliares, porém essenciais para o processamento de consultas e para a execução de procedimentos de análise espacial com eficiência
- Também são chamados de *índices espaciais*
  - Ao contrário dos índices convencionais, os espaciais são de uso obrigatório, para que o desempenho seja minimamente aceitável em BDs de tamanho razoável

Métodos de acesso espaciais

- Em geral, uma consulta envolve apenas uma pequena parcela do BD
- Percorrer todo o BD procurando pelos dados relevantes para a consulta é em geral muito ineficiente
- O método de acesso estabelece um *plano de execução* para a consulta

Métodos de acesso espaciais

- O plano de execução realiza uma *filtragem*, para determinar um subconjunto dos objetos do BD que *podem* atender às especificações da consulta
- Essa filtragem precisa ser executada com muita rapidez, e portanto é realizada sobre uma *aproximação* da forma geométrica de cada objeto

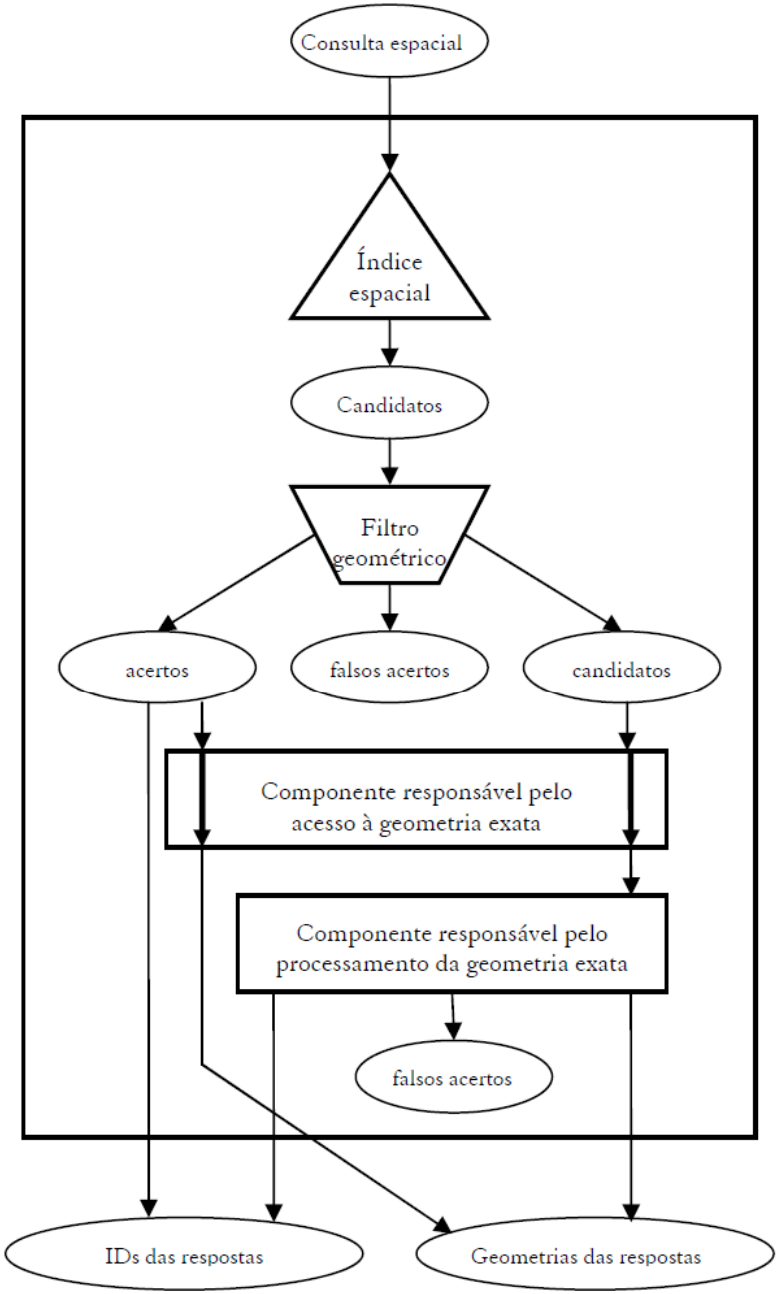
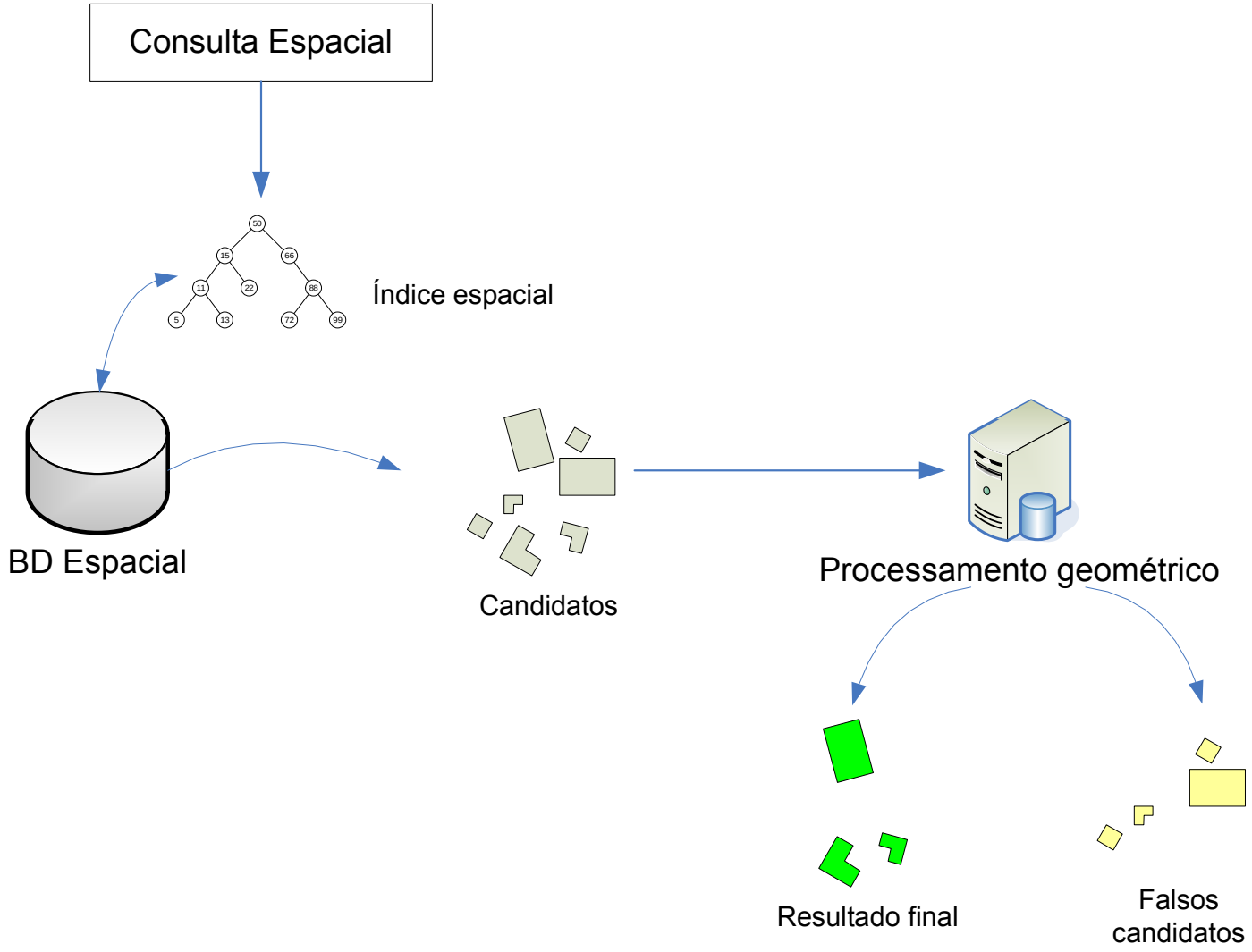


Figura 7.1 – Processamento de consultas espaciais (Kriegel et al., 1993).

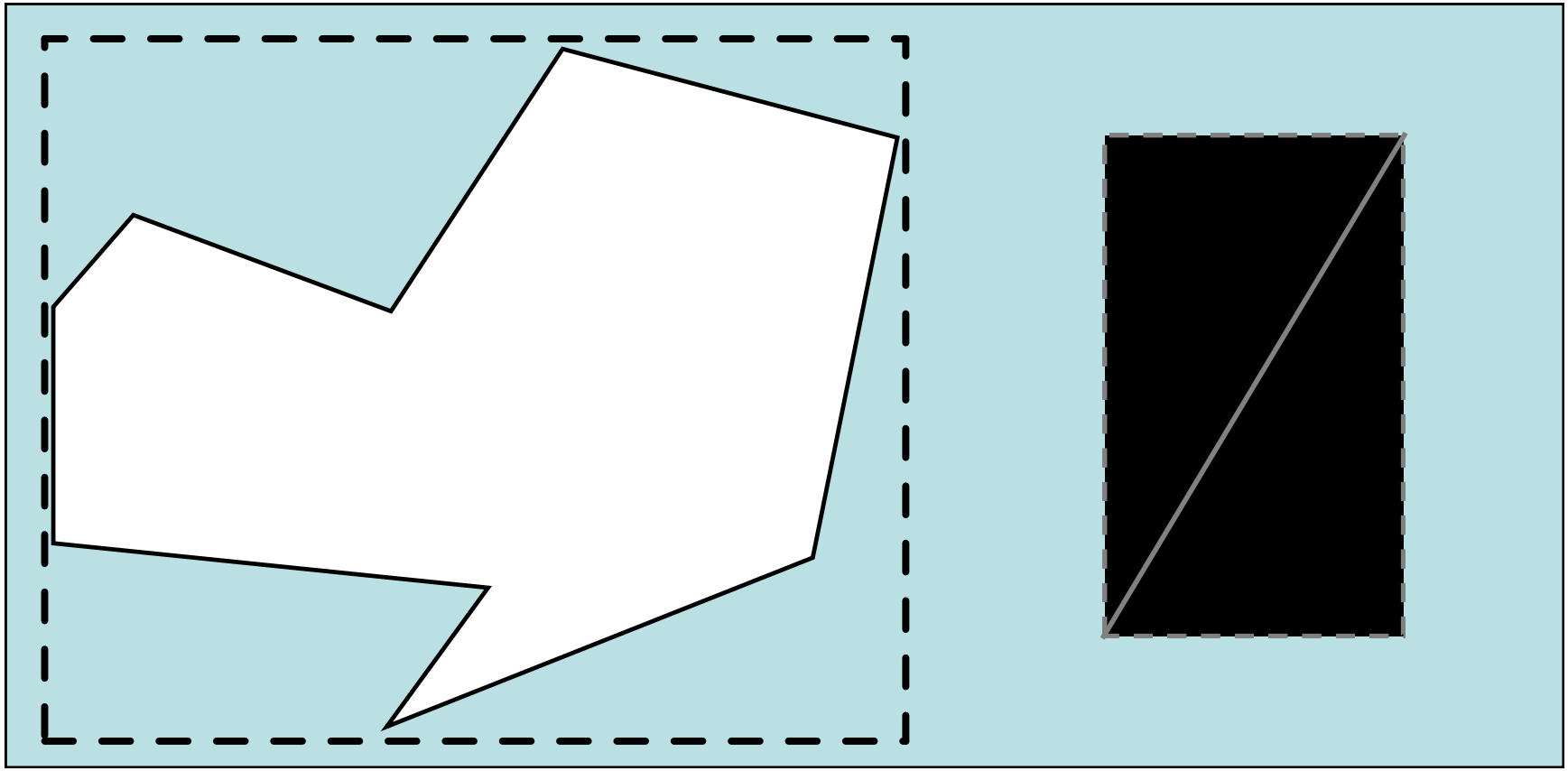
Processamento de consulta espacial



Filtragem:

Retângulo envolvente mínimo de um objeto

- Menor retângulo com lados paralelos aos eixos coordenados que contém totalmente o objeto



Uso de índices espaciais

- Seleção de objetos para visualização
  - Objetos contidos no retângulo do zoom
- Localização de objetos selecionados por apontamento
  - Objetos cujas fronteiras contêm ou se aproximam do ponto indicado na tela
- Consultas topológicas
  - Encontrar objetos relacionados topologicamente a um objeto espacial dado (contido em, contém, adjacente a, cruzando, etc.)

Índices espaciais

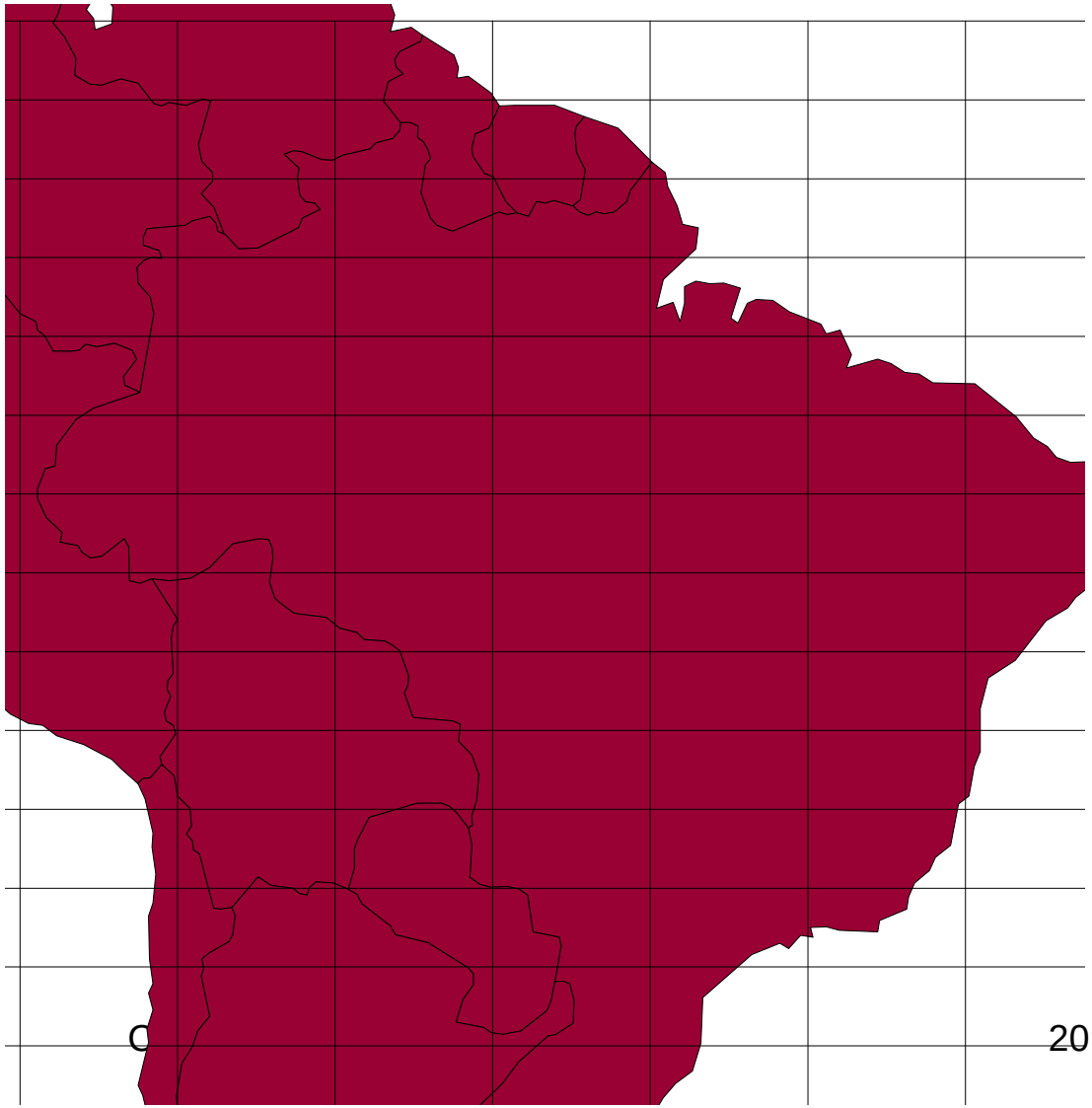
- Tipos encontrados em SIG
  - Sem indexação
  - *Grid file*
  - *K-d tree*
  - Quad-tree
  - Tiling
  - R-tree

SIG sem Indexação Espacial

- Sistemas baseados em CAD
  - Organização seqüencial
  - Baixo desempenho para arquivos grandes
  - Sistemas manuais de indexação
  - Mapa chave
  - Diversos arquivos

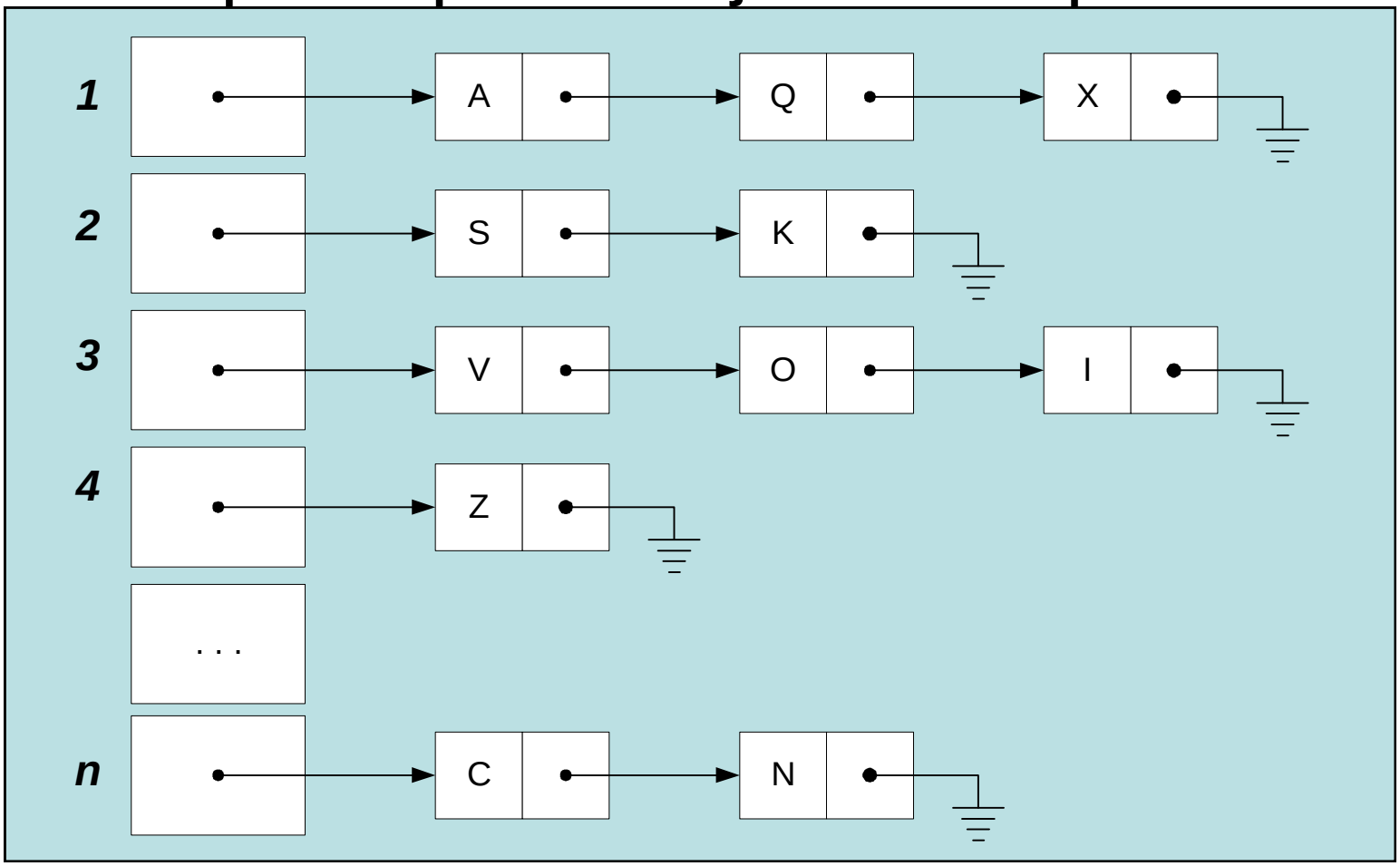
SIG sem indexação espacial

- Mapa chave
- Um arquivo seqüencial para cada folha de mapa
- Manutenção por conta do usuário



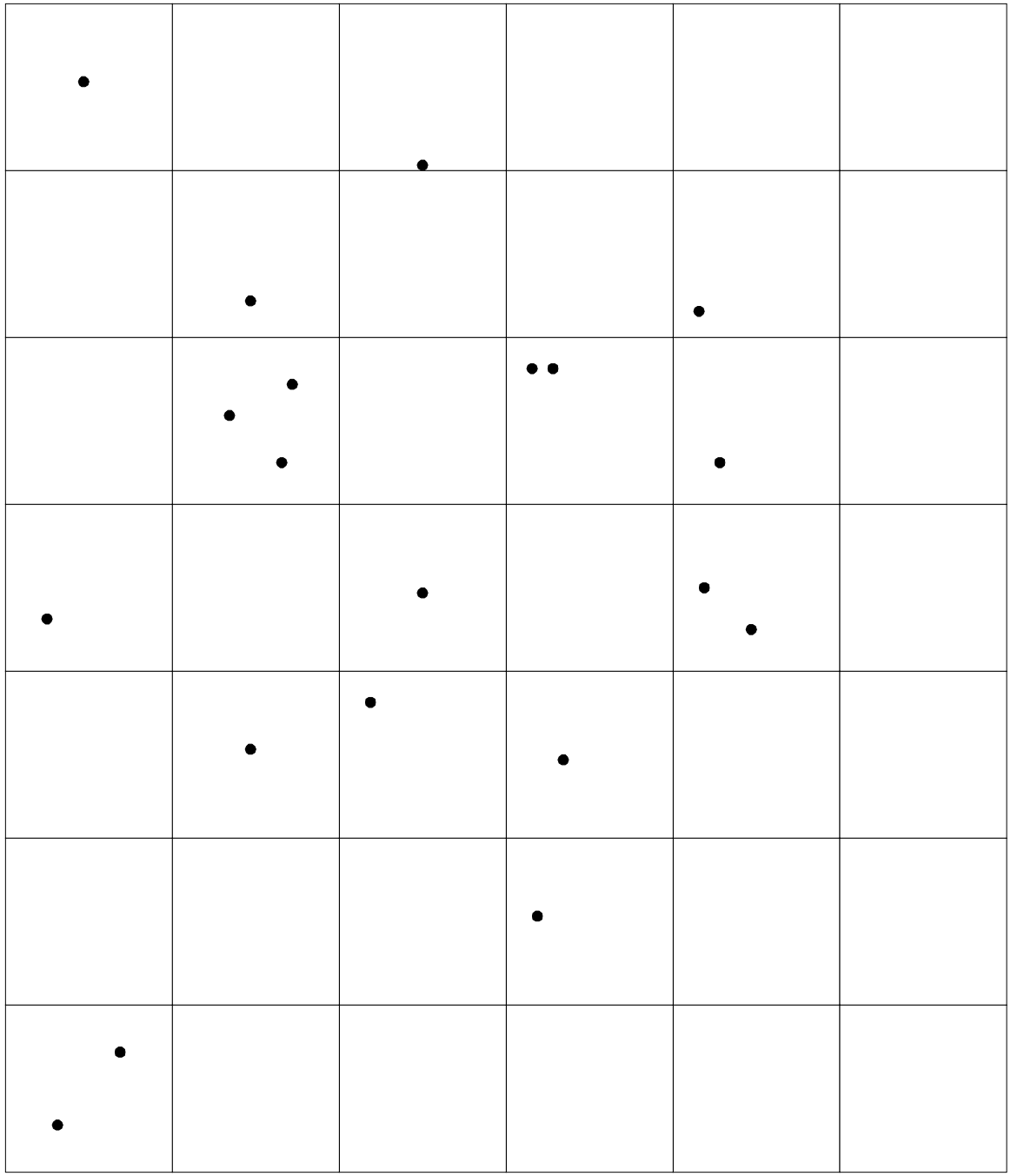
Grid file

- Baseado em *hashing*
- Adequado para conjuntos de pontos



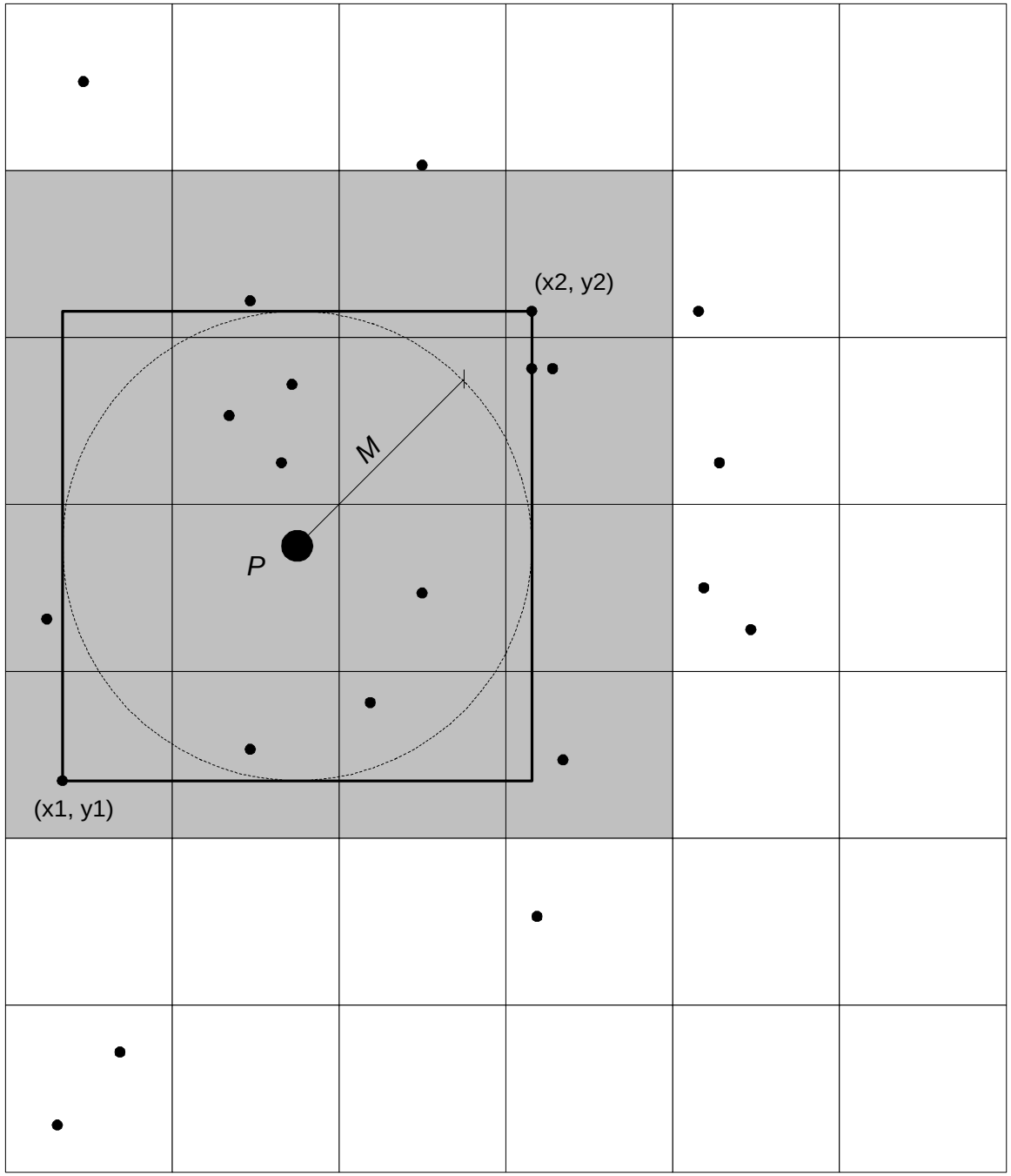
Grid file

- *Hashing* em duas dimensões, usando uma grade regular
- A escolha das dimensões dos elementos da grade é orientada pela aplicação



Grid file

- Pesquisa: pontos a menos de M metros do ponto P
- Apenas os *buckets* em cinza são lidos



K-d Tree

- Indexa chaves formadas por *k* atributos (geometricamente, dimensões)
- Cada nível da árvore corresponde a uma das dimensões
- As dimensões ocorrem ciclicamente pelos níveis da árvore

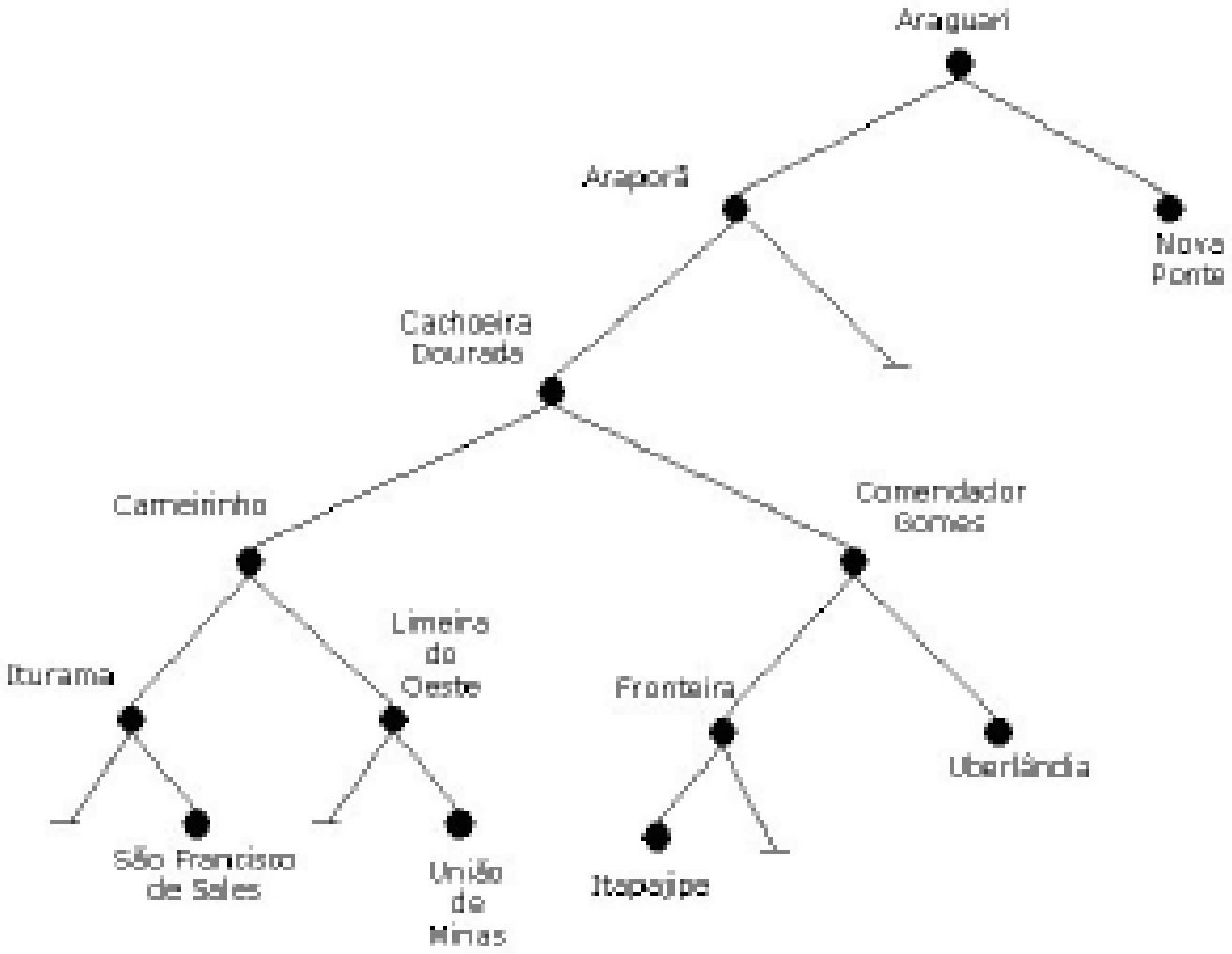
K-d Tree

- Exemplo
  - Duas dimensões: *x* e *y*
  - Níveis pares indexam o *x* e níveis ímpares indexam o *y*

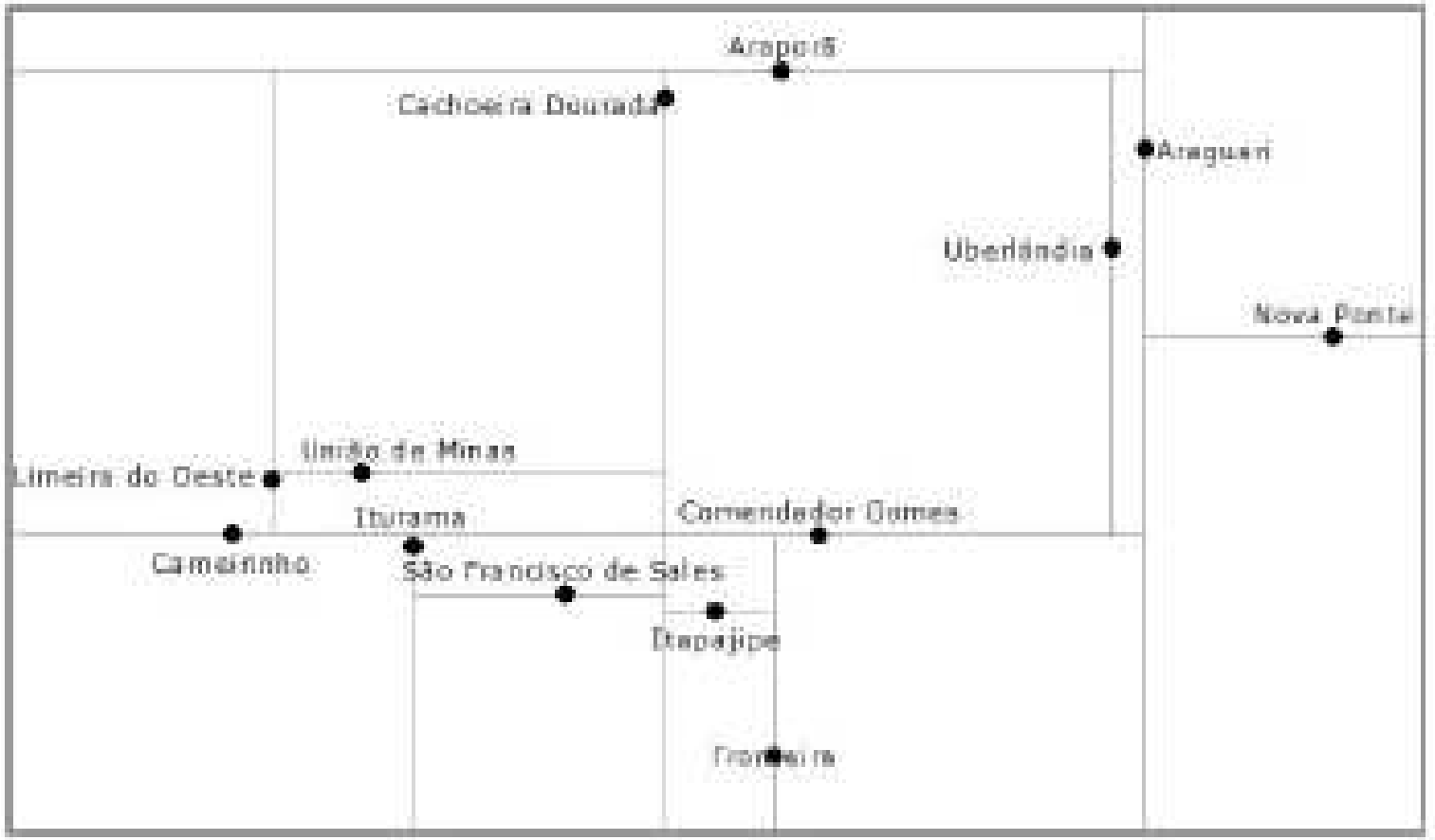
K-d Tree

Figura 6.5 – Sedes de alguns municípios do Estado de Minas Gerais.

K-d Tree


Araguari
Araporã
Nova Ponte
Cachoeira Dourada
Carneirinho
Comendador Gomes
Iturama
Limeira do Oeste
Fronteira
Uberlândia
São Francisco de Sales
União de Minas
Itapajipe

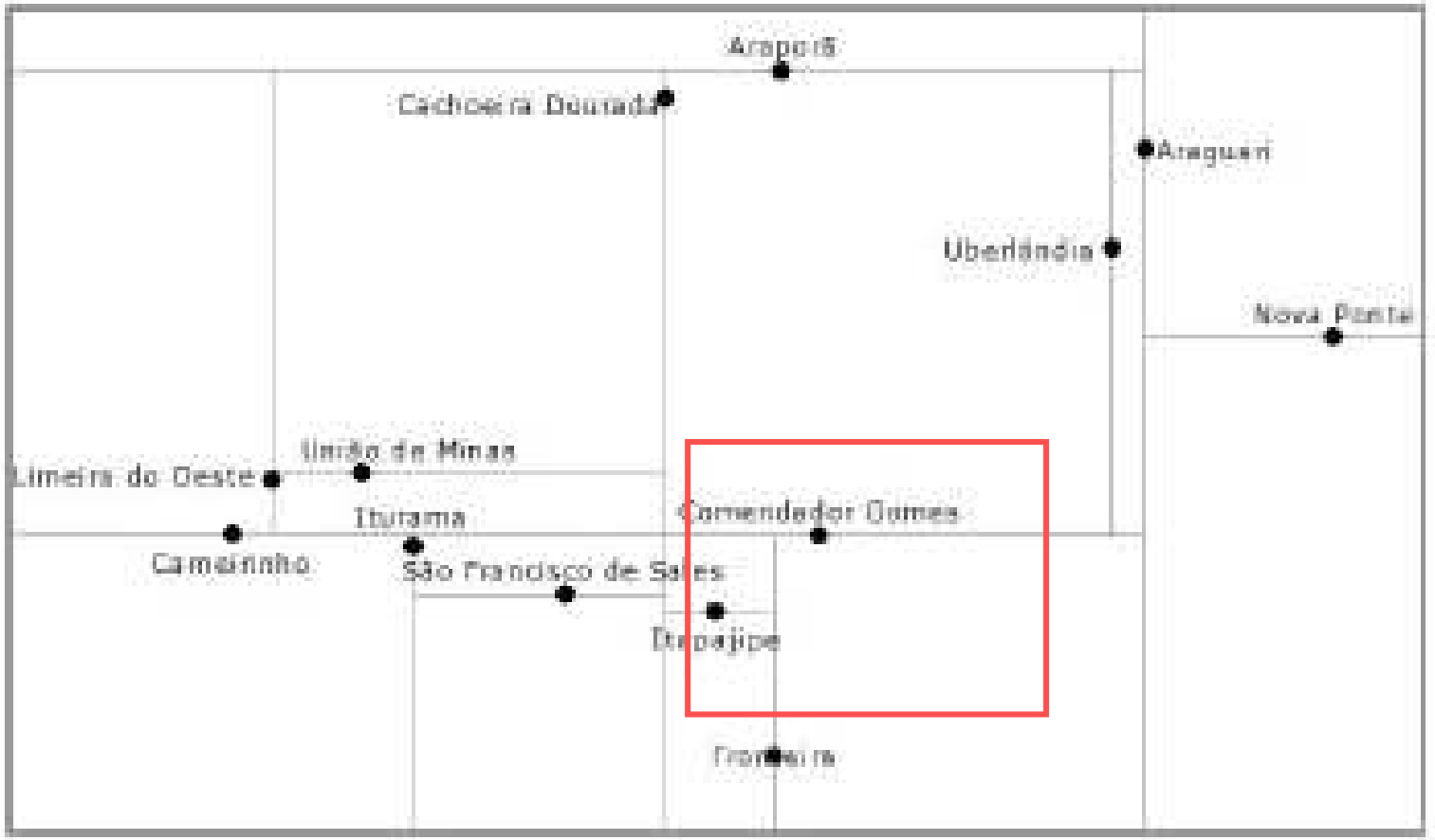
K-d Tree


Araporã
Cachoeira Dourada
Araguari
Uberlândia
Nova Ponte
União de Minas
Limeira do Oeste
Iturama
Comendador Gomes
Campinorte
São Francisco de Sales
Itapajipe
Fronteira

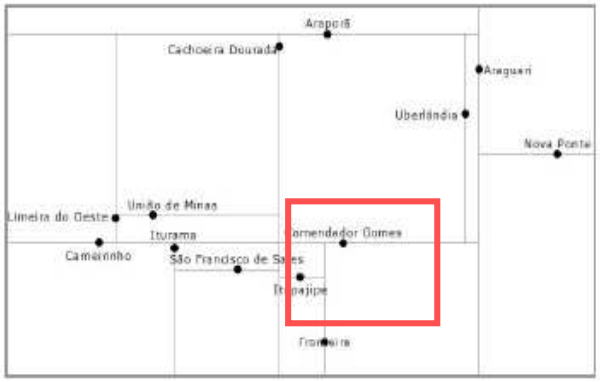
K-d Tree

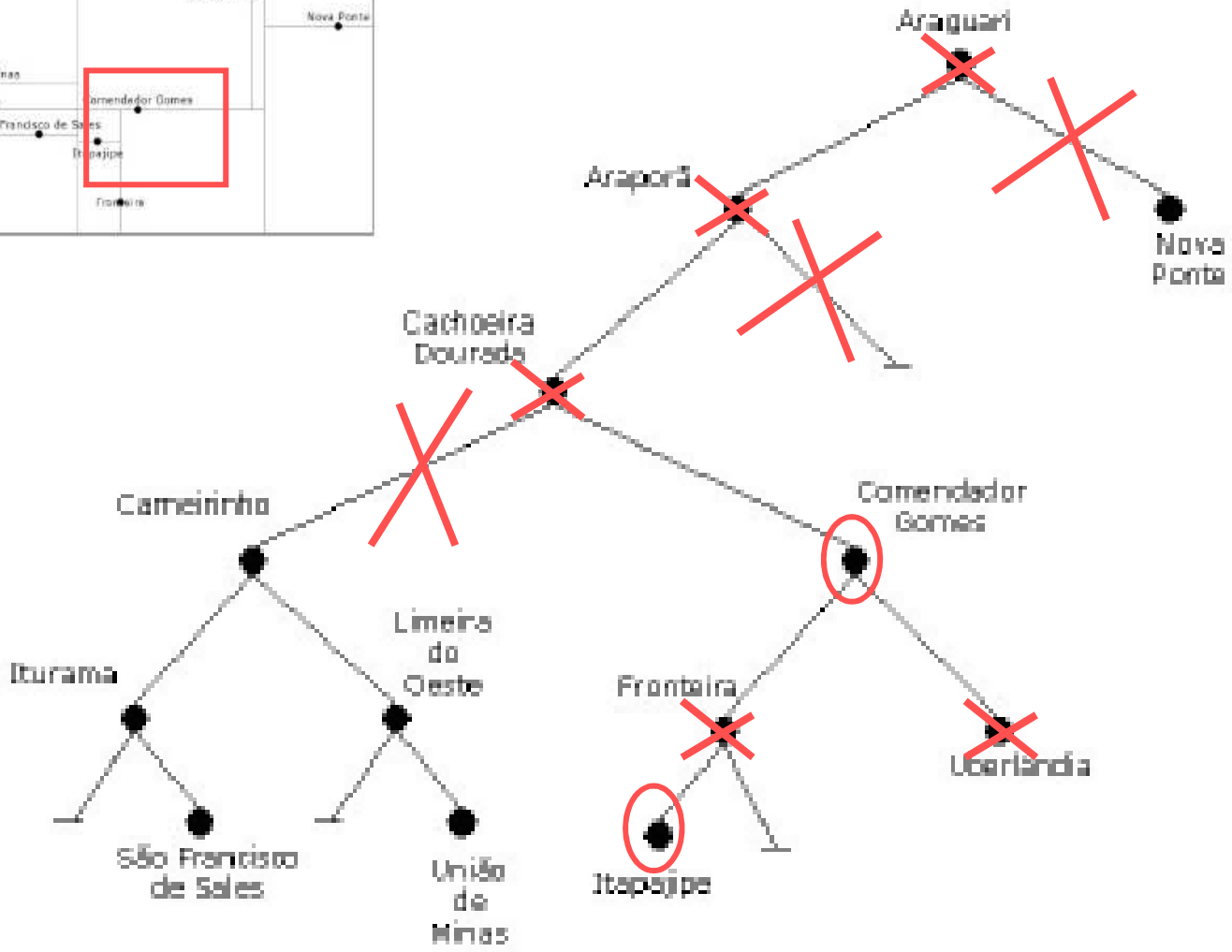
- Pesquisa na k-d tree, partindo de um retângulo de coordenadas dadas
  - Verificar se o nó corrente encontra-se no intervalo, se for ele é incluído na saída
  - Se o nível for par
    - Se o x1 do retângulo for menor que o x do nó, caminhar pela sub-árvore à esquerda e desprezar o restante
    - Se o x2 do retângulo for maior que o x do nó, caminhar pela sub-árvore à direita e desprezar o restante
  - Se o nível for ímpar
    - Idem, considerando y1, y2 e y do nó.

K-d Tree


Araporã
Cachoeira Dourada
Araguari
Uberlândia
Nova Ponte
União de Minas
Limeira do Oeste
Iturama
Comendador Gomes
Campinorte
São Francisco de Sales
Itapajipe
Fronteira

K-d Tree

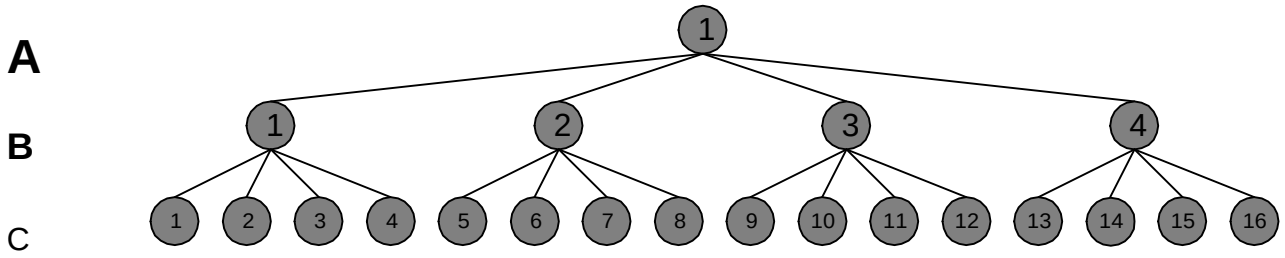

Araporã
Cachoeira Dourada
Araguari
Uberlândia
Nova Ponte
União de Minas
Limeira do Oeste
Ituiutaba
Comendador Gomes
Cameirinho
São Francisco de Sales
Itapajipe
Fronteira

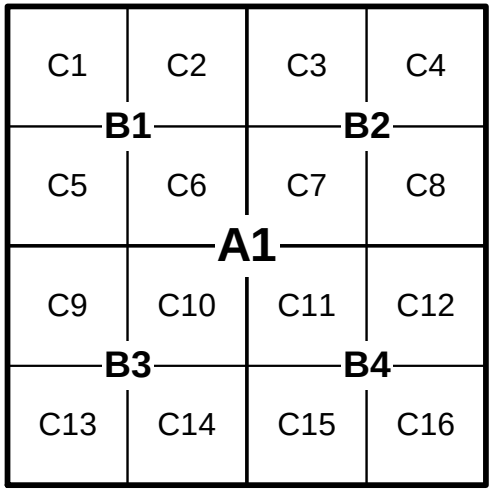

Araguari
Araporã
Nova Ponte
Cachoeira Dourada
Cameirinho
Comendador Gomes
Limeira do Oeste
Ituiutaba
Fronteira
Uberlândia
São Francisco de Sales
União de Minas
Itapajipe

Quad-tree

- Espécie de árvore em que cada nó possui sempre quatro “folhas”
- A cada nó corresponde uma região quadrada do espaço
- Os objetos são relacionados ao menor quadrado que contém seu retângulo envolvente

Quad-tree

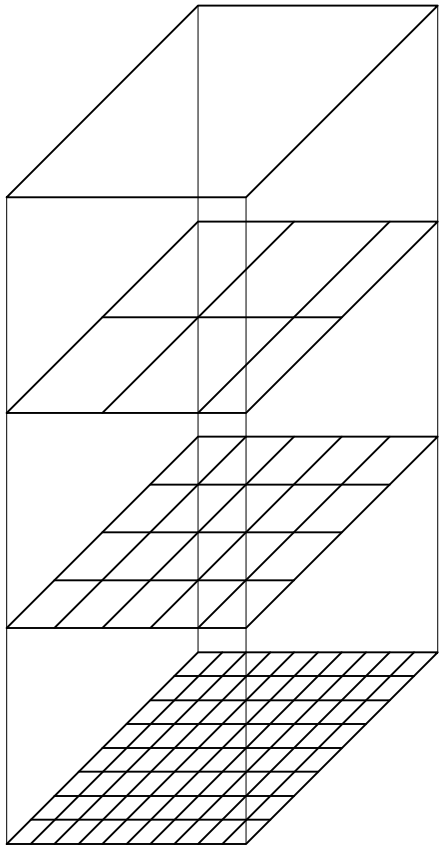

A
B
C
1
1
2
3
4
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


C1
C2
C3
C4
B1
B2
C5
C6
C7
C8
A1
C9
C10
C11
C12
B3
B4
C13
C14
C15
C16

Animação do funcionamento de quad-trees (PR-quadtree)

- http://njord.umiacs.umd.edu:1601/users/brabec/quadtree/points/pointquad.html

Tiling



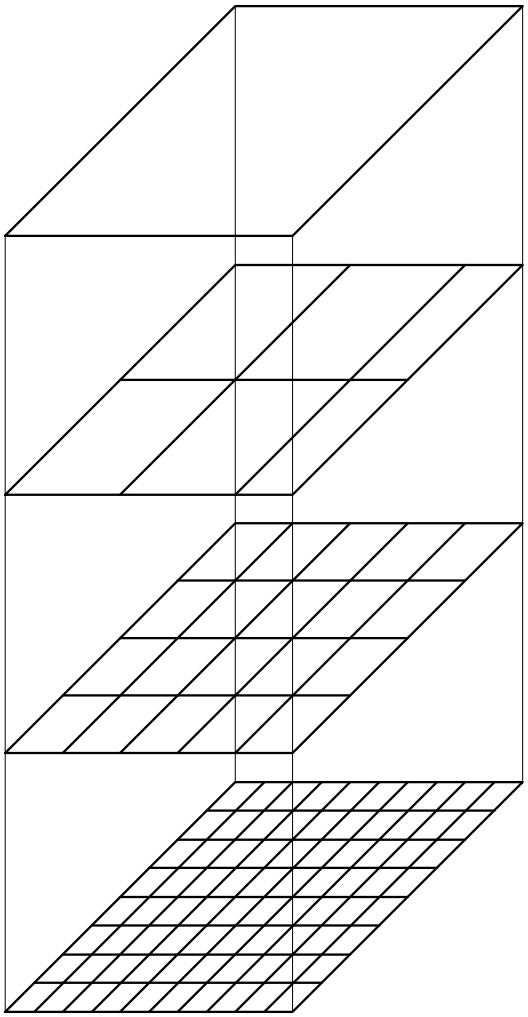
Camada 0: toda a base de dados

Camada 1: largura 10000

Camada 2: largura 5000

Camada 3: largura 1000

- Tiles, ladrilhos, quadrados
- Dimensões fixas
- Tamanho adequado aos objetos geográficos
- Número fixo de “quadrados”
- Limites conhecidos dos “quadrados”

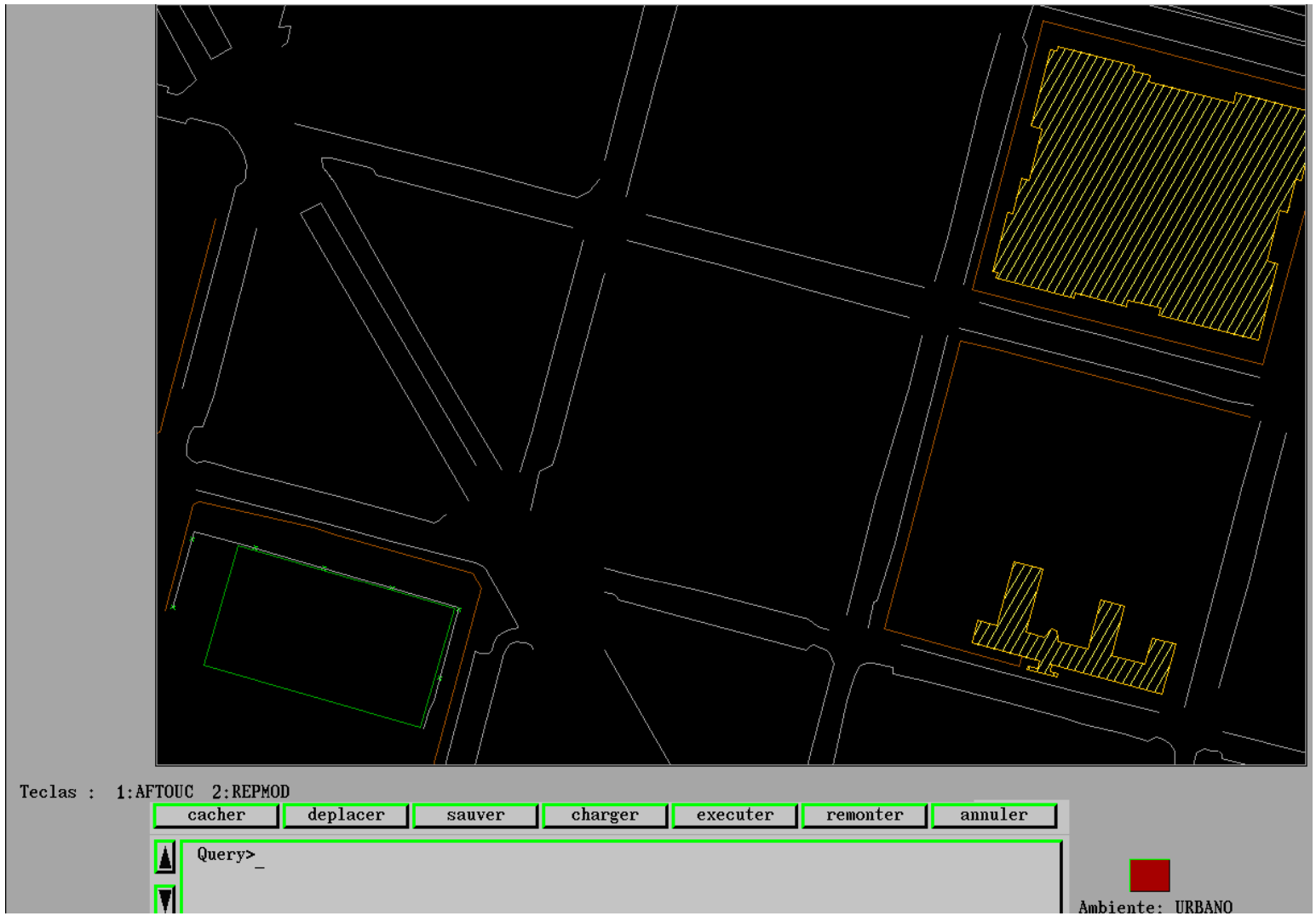


Camada 0: toda a base de dados

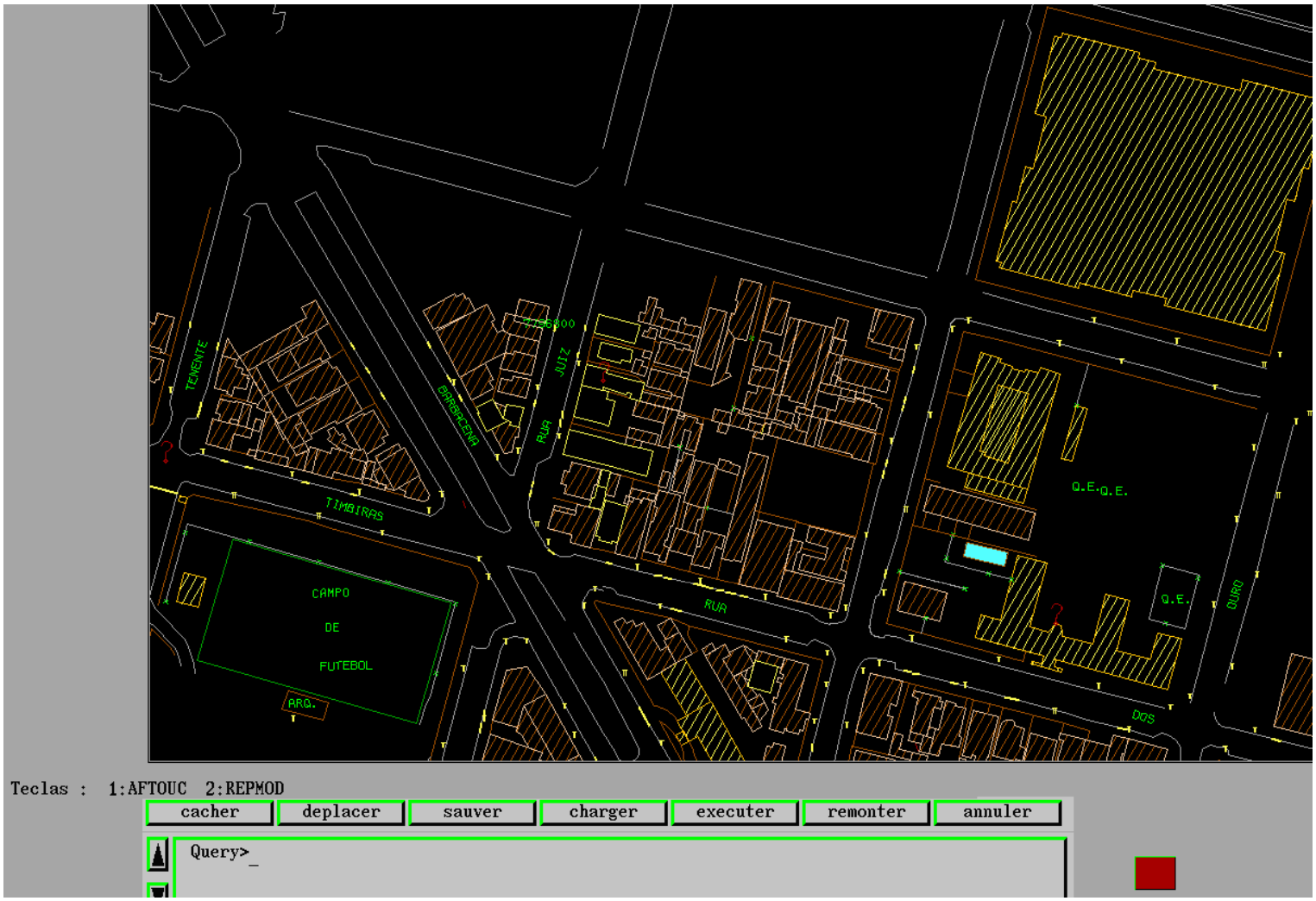
Camada 1: largura 10000

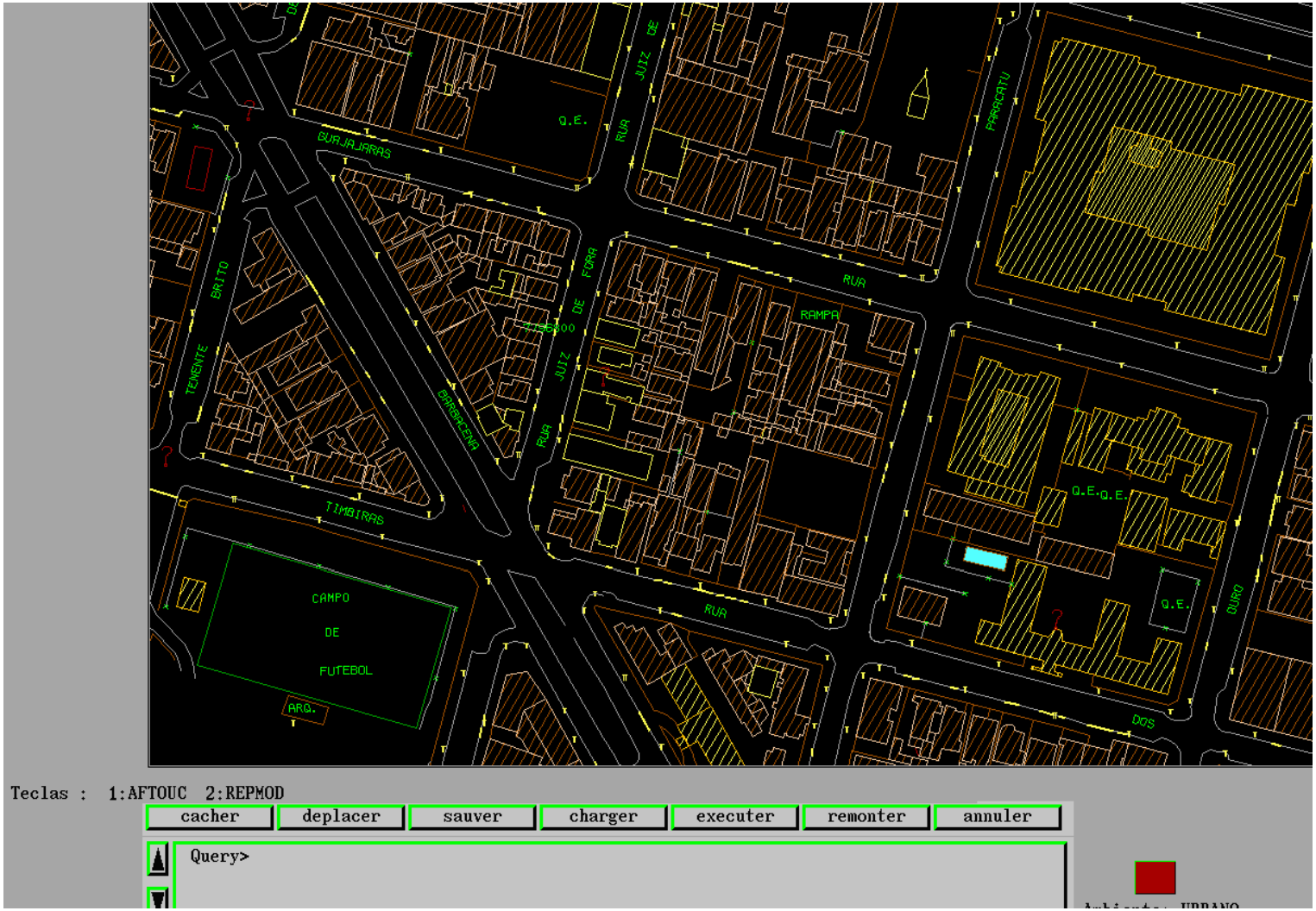
Camada 2: largura 5000

Camada 3: largura 1000


Teclas : 1:AFTOUC 2:REPMOD
cacher
deplacer
sauver
charger
executer
remonter
annuler
Query>_
Ambiente: URBANO


7796800
TENENTE
CARNEIRO
RUA
JUIZ
TIMBIRAS
CAMPO
DE
FUTEBOL
ARQ.
Teclas : 1:AFTOUC 2:REPMOD
cacher
deplacer
sauver
charger
executer
remonter
annuler


TENENTE
BARBACENA
RUA
JUIZ
TIMBIRAS
CAMPO
DE
FUTEBOL
ARQ.
RUA
Q.E.
Q.E.
Q.E.
OURO
DOS
Teclas : 1:AFTOUC 2:REPMOD
cacher
deplacer
sauver
charger
executer
remonter
annuler
Query>_

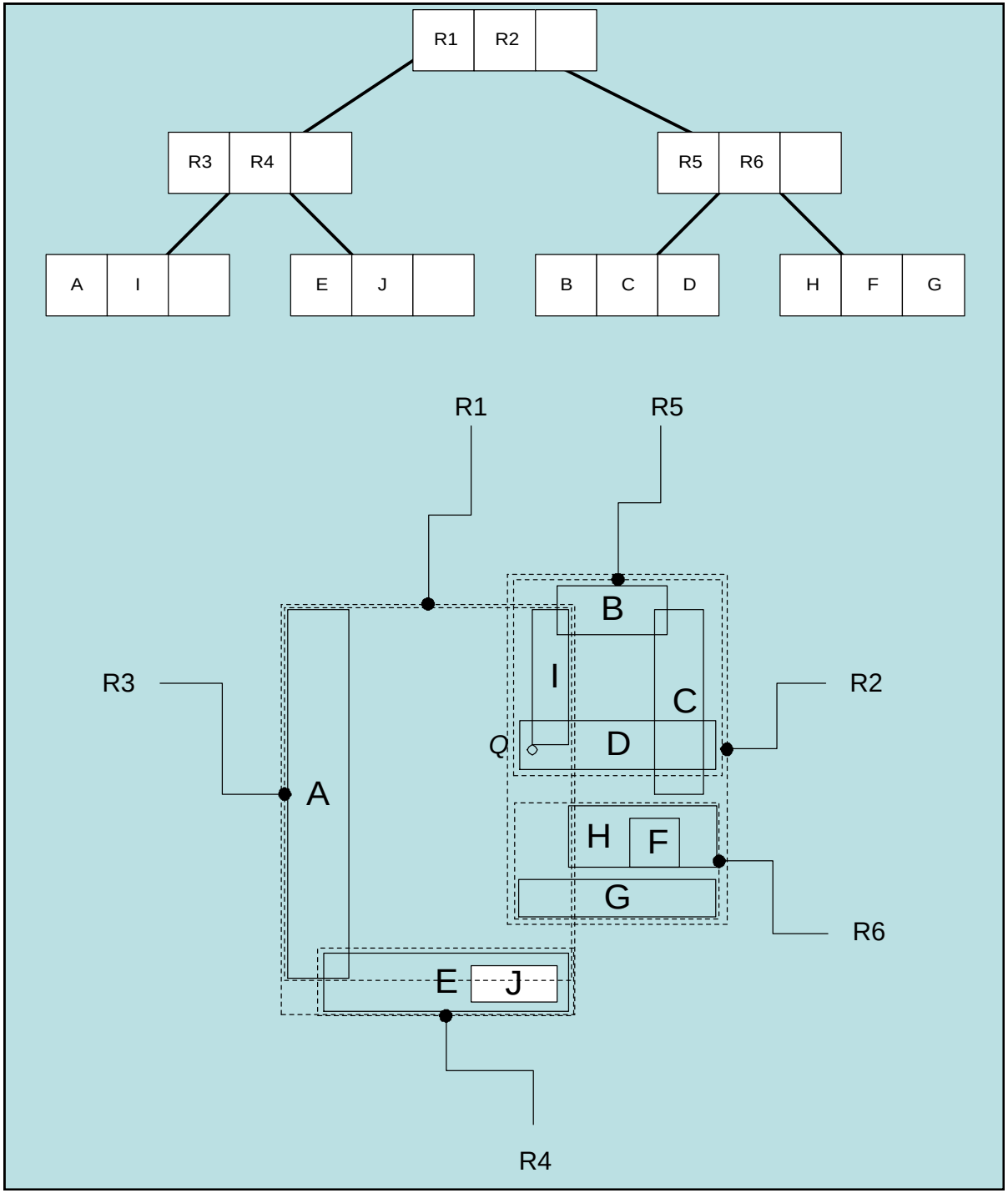

Q.E.
RUA
JUIZ DE
PARACATU
GUAJAJARAS
BRITO
TENENTE
FORA
JUIZ DE
RUA
RAMPA
BARBACENA
RUA
TIMBIRAS
CAMPO
DE
FUTEBOL
ARQ.
RUA
Q.E.
Q.E.
Q.E.
OURO
DOS
Teclas : 1:AFTOUC 2:REPMOD
cacher
deplacer
sauver
charger
executer
remonter
annuler
Query>

R-tree

- Indexa objetos pelo seu retângulo envolvente mínimo
- Pontos também podem ser indexados (retângulo envolvente nulo)
- Cada bloco de armazenamento pode conter um número variável de retângulos
- O aumento do número de objetos causa subdivisões nos blocos, e a redução provoca fusões de blocos

R-tree

- Exemplo: que objetos contêm o ponto *Q*?



Algoritmos da R-Tree

- Search
- Insert
- Delete
- Update
- Node splitting
  - Exhaustive
  - Quadratic
  - Linear

Search

Encontrar todas as entradas cujo REM sobrepõe um retângulo de busca S, começando pelo nodo T

- S1. Se o nó T não é folha, verificar cada entrada E para determinar se E.ret sobrepõe S; caso positivo, chamar *search* recursivamente a partir de cada $E.p_i$ (filhos)
- S2. Se T é um nodo folha, verificar todas as entradas E para determinar se E.ret sobrepõe S; caso positivo, E vai para a saída

Insert

Inserir a entrada E a partir de T

- I1. Chamar *ChooseLeaf* para selecionar um nodo folha L no qual E vai ser posicionado
- I2. Se L tiver espaço para mais uma entrada, acomodar E; caso contrário, chamar *SplitNode* para obter L e LL, que conterão L e todas as entradas atuais
- I3. Chamar AdjustTree a partir de L, passando também LL se um *split* foi realizado
- I4. Se o *split* foi propagado até a raiz, causando seu particionamento, criar uma nova raiz

ChooseLeaf

- CL1. Seja T o nodo raiz
- CL2. Se T for nodo folha, retornar T
- CL3. Caso contrário, seja F a entrada de T cujo retângulo precisaria ser *menos* expandido (largura ou altura) para acomodar E.ret; caso ocorra empate, resolver pela área do retângulo
- CL4. Faça T o nodo filho correspondente a F e faça uma chamada recursiva a *ChooseLeaf*

AdjustTree

Ascender da folha L até a raiz, ajustando os retângulos de cobertura e propagando os splits se necessário

- AT1. Inicialize, fazendo N = L; se L foi particionado anteriormente, inicializar também NN a partir de LL
- AT2. Se N for a raiz, termine
- AT3. Ajustar o retângulo de cobertura no nodo acima de N
- AT4. Se existe NN, criar uma nova entrada ENN, com um retângulo de cobertura adequado, e adicionar ENN no nodo acima de N se houver espaço; caso não haja, chamar *SplitNode* sobre P com NN, produzindo P e PP
- AT5. Faça N = P e NN = PPs se tiver ocorrido um *split*, e iterar em AT2

Delete

Excluir o registro E da árvore

- D1. Chamar *FindLeaf* para encontrar o nodo folha L que contém E; se não for encontrado, terminar
- D2. Remover E de L
- D3. Chamar *CondenseTree* passando L
- D4. Se a raiz tiver apenas um filho após o ajuste, transformar esse filho na raiz

FindLeaf

Encontrar a entrada E a partir da raiz T

- FL1. Se T não for um nodo folha, verificar cada entrada F em T para determinar se F.ret contém E.ret; para cada entrada com resposta positiva, chamar *FindLeaf* recursivamente
- FL2. Se T for um nodo folha, verificar cada entrada para encontrar E; se E for encontrado, retornar T

CondenseTree

Dado um nodo do qual foi retirada uma entrada, eliminar o nodo se necessário e realocar suas entradas

- CT1. Inicialize, fazendo N = L, e criar Q = { } para ser o conjunto de nodos eliminados
- CT2. Se N for a raiz, pular para CT6; caso contrário, P é o pai de N e EN é a entrada de N em P
- CT3. Se N tiver menos de *m* entradas, deletar EN de P e adicionar N ao conjunto Q
- CT4. Se N não for eliminado, reajustar EN.ret para conter todas as entradas de N
- CT5. Fazer N = P e iterar a partir de CT2
- CT6. Reinserir na árvore as entradas que restarem em Q; entradas provenientes de níveis superiores devem ficar mais altas na árvore, para equilibrá-la na altura

SplitNode

- Para criar um novo nodo com capacidade para *M* entradas, é necessário dividir as *M+1* entradas entre dois nodos
- A divisão deve ser feita de modo a minimizar a possibilidade de ambas as subárvores terem que ser verificadas no momento de uma consulta
  - = minimizar a sobreposição entre os retângulos
  - = minimizar a área total dos dois retângulos
- O mesmo critério é aplicado em *ChooseLeaf*

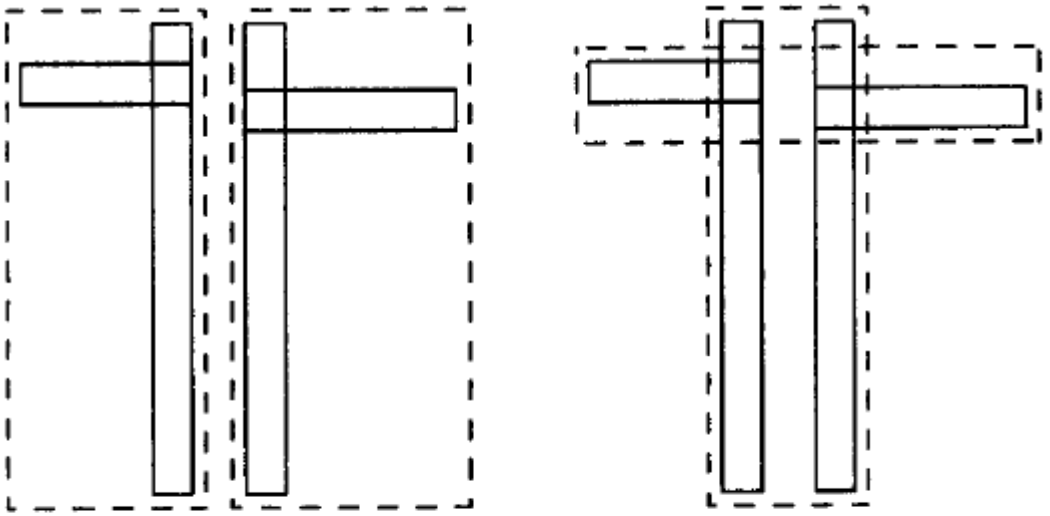


Figure 3 1

Fonte: Guttman (1984)

SplitNode: escolha exaustiva

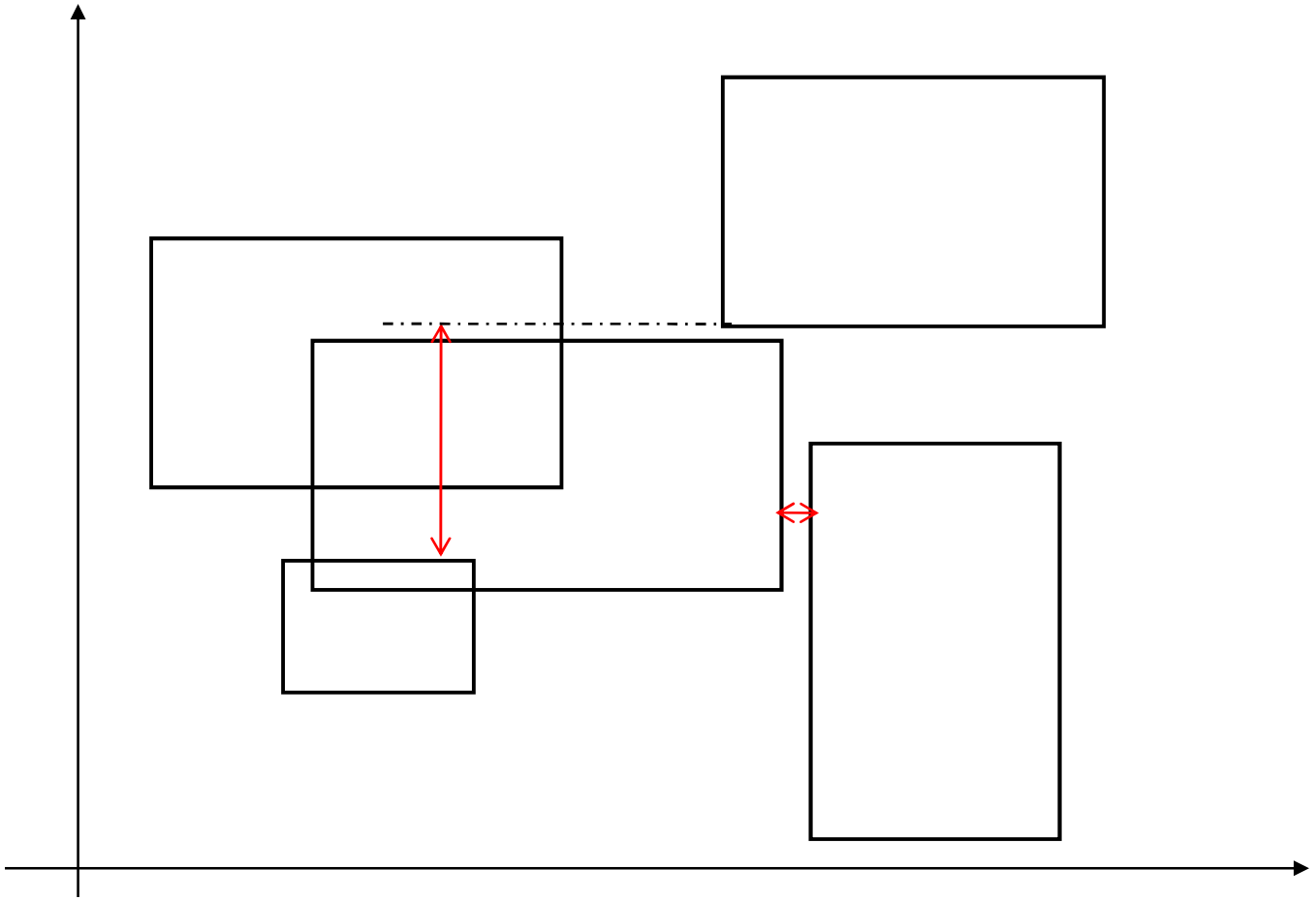
- Analisar todas as combinações de agrupamentos de retângulos seria o ideal
- Porém, se considerarmos blocos de 1024 bytes, *M* pode chegar facilmente a 50 entradas, e portanto gerando 2^{50-1} combinações, o que é inviável

SplitNode: algoritmo quadrático

- Não garante a solução ótima
- Custo quadrático em M
- O algoritmo escolhe duas das M+1 entradas para ser os primeiros elementos (sementes) dos dois novos nodos
  - Critério: maior área combinada do retângulo para cobrir a ambos
- A cada passo, calcula-se o acréscimo de área correspondente a cada entrada para cada um dos grupos
  - A entrada que tiver a maior diferença no acréscimo de área será então associada ao grupo sobre o qual causou o menor impacto

SplitNode: algoritmo linear

- A diferença para o quadrático está no processo de escolha das sementes
- Ao longo de cada dimensão, encontrar o retângulo que tem a maior coordenada inferior, e o retângulo que tem a menor coordenada superior
  - Ex.: em x, encontrar quem tem o maior x1 e o menor x2
- Armazenar a distância de separação entre esses retângulos
  - Ex: diferença entre o menor x2 e o maior x1
- Normalizar usando como denominador a largura total segundo a dimensão
- Escolher as entradas com a maior separação dentre as duas dimensões



R-tree

- Existem diversas variações na literatura
  - R^+-Tree
  - R*-Tree
  - Hilbert R-Tree
  - X-Tree
  - ...
- O mais usual é encontrar implementações da R-tree original nos produtos

R^+-Tree

- A R^+-tree é um refinamento da R-tree em que não se permitem retângulos superpostos em nós que não sejam folhas
  - A regra de não-superposição faz com que a árvore fique mais esparsa, e portanto menos eficiente no disco; algoritmos mais sofisticados de inserção e exclusão são necessários para garantir que a árvore fique pelo menos “meio-cheia”
  - Obtém bons resultados para consultas baseadas em pontos e em retângulos, mas tem custo maior na inserção e exclusão

R*-Tree

- Modifica a estratégia de divisão de nodos
- Introduz a idéia de reinserção de objetos na árvore em caso de overflow no nodo
- Os algoritmos de consulta e exclusão são os mesmos da R-Tree original

Animação do funcionamento de R-trees

- http://gis.umb.no/gis/applets/rtree2/jdk1.1/
  - Visualização da árvore
- http://donar.umiacs.umd.edu/quadtree/
  - Vários tipos de índices espaciais
- http://research.att.com/~marioh/spatialindex/
  - Biblioteca spatialindex e visualizador de R-Tree 3D

Otimização de consultas espaciais

- Seleções com funções geográficas

```
SELECT d.nome, d.populacao
FROM cidade d, cidade c
WHERE c.nome = ‘Belo Horizonte’
AND ST_DISTANCE(c.geom, d.geom) < 50000
AND d.populacao > 50000
```

- Junções com critério geográfico

```
SELECT c.nome, a.nomeaerop
FROM cidade c, aeroporto a
WHERE a.pavimento = ‘Asfalto’
AND ST_DISTANCE(c.geom, a.geom) < 50000
```

Seleção: plano de execução

- P11. Determine a posição de ‘Belo Horizonte’ (p)
- P12. Determine o conjunto C’ de cidades a menos de 50 km da posição p
- P13. Determine o subconjunto C de C’ de cidades com mais de 50 mil habitantes

Seleção: plano de execução canônico

- P11. Determine a posição de ‘Belo Horizonte’ (p)
  - Convencional: busca exaustiva pelo atributo nome ou uso de índice convencional, se houver
- P12. Determine o conjunto C’ de cidades a menos de 50 km da posição p
  - Espacial: depende do processamento da função de distância; acelerado na presença de índice espacial na etapa de filtragem
- P13. Determine o subconjunto C de C’ de cidades com mais de 50 mil habitantes
  - Convencional: busca exaustiva em C’

- P12. Determine o conjunto C’ de cidades a menos de 50 km da posição p
  - P121. Usando o índice espacial, determine as cidades que *podem estar* a menos de 50km de p, criando o conjunto I
  - P122. Leia a geometria dos objetos de I
  - P123. Verifique quais objetos de I estão de fato a menos de 50km de p

Seleção: plano de execução alternativo

- P21. Determine a posição de ‘Belo Horizonte’ (p)
- P12. Determine o conjunto C’ de cidades a com mais de 50 mil habitantes
- P13. Determine o subconjunto C de C’ de cidades a menos de 50 km da posição p

Seleção: plano de execução alternativo

- P21. Determine a posição de ‘Belo Horizonte’ (p)
  - Idem ao anterior
- P22. Determine o conjunto C’ de cidades a com mais de 50 mil habitantes
  - Eficiente se existir um índice sobre a população
- P23. Determine o subconjunto C de C’ de cidades a menos de 50 km da posição p
  - Teste exaustivo sobre o resultado de P12; pode ser prejudicado se retornarem muitas cidades de P22

Seleção: outro plano de execução alternativo

- P31. Determine a posição de ‘Belo Horizonte’ (p)
- P32. Determine o conjunto C’ de cidades a com mais de 50 mil habitantes
- P33. Determine o conjunto C’’ de cidades a menos de 50 km da posição p
- P34. Determine a interseção de C’ e C’’

Seleção: outro plano de execução alternativo

- P31. Determine a posição de ‘Belo Horizonte’ (p)
  - Idem aos anteriores
- P32. Determine o conjunto C’ de cidades a com mais de 50 mil habitantes
  - Idem ao anterior
- P33. Determine o conjunto C’’ de cidades a menos de 50 km da posição p
  - Idem ao primeiro plano
- P34. Determine a interseção de C’ e C’’

Junção: plano de execução

- P11. Determine o conjunto R, contendo nome n e localização l de cada aeroporto pavimentado
  - Convencional
- P12. Inicialize uma lista de saída vazia
- P13. Para cada elemento (n, l) de R, faça
  - P131. Determine o conjunto C de cidades que estão a menos de 50km de l, e acrescente a cidade à lista de saída

Junção: plano de execução 2

- P21. Determine o conjunto C, contendo nome n e localização l de cada cidade
  - Convencional
- P22. Inicialize uma lista de saída vazia
- P23. Para cada elemento (n, l) de R, faça
  - P231. Determine o conjunto A de aeroportos que estão a menos de 50km de l, e acrescente o aeroporto à lista de saída

Junção: plano de execução 3

- P31. Determine o conjunto A, contendo a localização l e nome n de cada aeroporto pavimentado
- P32. Determine o conjunto C de cidades, contendo localização cl e nome cn;
- P33. Determine o conjunto R de resultados contendo apenas pares (cn, an) quando a distância entre l e cl for menor que 50km
  - Loops aninhados

Junção: um possível quarto plano de execução

- Ordenar os elementos de A e C pela coordenada x (uma única lista)
- Para o primeiro elemento da lista, verificar nos subsequentes quem está a uma distância inferior a 50km do primeiro, e é de classe diferente da dele
- Considere todos os elementos encontrados como processados, e retire-os da lista
- Retire o primeiro elemento da lista, e reinicie

Leitura complementar

- Davis Jr., C. A. e Queiroz, G. R. *Métodos de Acesso para Dados Espaciais*. Capítulos 6 e 7 de Casanova, M. A., Câmara, G., Davis Jr., C. A., Vinhas, L., Queiroz, G. R. (editores) *Bancos de Dados Geográficos*. Ed. MundoGeo, 2005.
- Guttman, A. R-Trees: a dynamic index structure for spatial searching. ACM SIGMOD, 47-57, 1984.
- Elmasri, R. and Navathe, S. *Fundamentals of Database Systems*, 4th ed. Pearson, 2002. [métodos de acesso para BD convencionais]