MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-15251-TDI/1338

# COMPARAÇÃO DO DESEMPENHO DOS ÍNDICES R-TREE, GRADES FIXAS, E CURVAS DE HILBERT PARA CONSULTAS ESPACIAIS EM BANCOS DE DADOS GEOGRÁFICOS

Frederico Augusto Bedê Teotônio

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Gilberto Câmara e Antônio Miguel Vieira Monteiro, aprovada em 27 de março de 2008.

O original deste documento está disponível em:
<http://urlib.net/sid.inpe.br/mtc-m17@80/2008/02.12.12.07>

INPE
São José dos Campos
2008

**PUBLICADO POR:**

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6911/6923
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-15251-TDI/1338

# COMPARAÇÃO DO DESEMPENHO DOS ÍNDICES R-TREE, GRADES FIXAS, E CURVAS DE HILBERT PARA CONSULTAS ESPACIAIS EM BANCOS DE DADOS GEOGRÁFICOS

Frederico Augusto Bedê Teotônio

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Gilberto Câmara e Antônio Miguel Vieira Monteiro, aprovada em 27 de março de 2008.

O original deste documento está disponível em:
<http://urlib.net/sid.inpe.br/mtc-m17@80/2008/02.12.12.07>

INPE
São José dos Campos
2008

Dados Internacionais de Catalogação na Publicação (CIP)

T265c Teotônio, Frederico Augusto Bedê.

Comparação do desempenho dos Índices R-Tree, Grades Fixas, e Curvas de Hilbert para consultas espaciais em Bancos de Dados Geográficos/ Frederico Augusto Bedê Teotônio. – São José dos Campos: INPE, 2008.

72p. ; (INPE-15251-TDI/1338)

1. Índices. 2. Eficiência. 3. Distribuição espacial. 4. Espaço de Hilbert. 5. Desempenho de sistemas computacionais. I. Título.

CDU 004.424.43

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de Mestre em Computação Aplicada

Dr. Rafael Duarte Coelho dos Santos

Presidente / INPE / SJCampos - SP

Dr. Gilberto Câmara

Orientador(a) / INPE / SJCampos - SP

Dr. Antonio Miguel Vieira Monteiro

Orientador(a) / INPE / SJCampos - SP

Dra. Lúbia Vinhas

Membro da Banca / INPE / São José dos Campos - SP

Dr. Clodoveu Davis Júnior

Aluno (a): Frederico Augusto Bede Teotonio

São José dos Campos, 27 de Março de 2008

*"Nenhum problema pode ser resolvido pelo mesmo estado de consciência que o criou. É preciso ir mais longe. Eu penso 99 vezes e nada descubro. Deixo de pensar, mergulho num grande silêncio e a verdade me é revelada".*

ALBERT EINSTEIN

*A meus pais*

## AGRADECIMENTOS

Agradeço às pessoas que me ajudaram a vencer mais esta etapa da vida.

Ao Instituto Nacional de Pesquisas Espaciais – INPE, pela oportunidade de estudos e utilização de suas instalações.

Aos professores do INPE pelo conhecimento compartilhado.

Aos meus orientadores Prof. Dr. Gilberto Câmara e Prof. Dr. Antonio Miguel Vieira Monteiro, pelo conhecimento passado, e pela orientação e apoio na realização deste trabalho.

À Prof. Dr. Lúbia Vinhas e ao Prof. Gilberto Ribeiro, pela orientação, apoio na realização deste trabalho e principalmente pelas pessoas especiais que demonstraram ser. Também não poderia deixar de agradecer as horas de conversa que tivemos.

Ao amigo Gilberto Ribeiro de Queiroz pela amizade, paciência e toda ajuda prestada. Um amigo de pouco tempo, que considero como a um irmão.

Aos meus amigos e simpatizantes da República: Sidney (Sidão), Henrique (Calango) e Geraldo (o Caçapa) por me receberem e conviverem comigo por grande parte desta jornada.

À minha futura esposa Flavinha, pelo amor e compreensão, mesmo nos tempos de difícil compreensão. Aos pais dela, meus futuros sogros, que se tornaram a minha família.

Aos meus pais e à toda minha família por sempre acreditarem na importância do estudo, apoiarem as minhas decisões e pela presença, apesar da grande distância que nos separa.

## RESUMO

O desenvolvimento de bancos de dados espaciais tem sido influenciado pelas pesquisas sobre indexação espacial. Várias pesquisas de índices multidimensionais podem ser encontradas na literatura, como as R-trees e suas variantes. O aumento no desempenho das operações espaciais com a utilização de índices espaciais tem levado os desenvolvedores a incluir o suporte à R-tree em seus produtos. Apesar desses avanços, existem Sistemas Gerenciadores de Bancos de Dados (SGBD) que não fornecem suporte a dados e operações espaciais. Usuários desses tipos de SGBD podem querer o desenvolvimento de aplicativos espaciais sobre estas plataformas, mas podem ser desencorajados por restrições de desempenho. Uma solução óbvia é implementar o mecanismo de indexação espacial no núcleo do SGBD. No entanto, esta solução pode não ser viável, devido a restrições de tempo e custo ou por falta de acesso aos códigos fontes dos SGBDs (caso de sistemas proprietários). Este trabalho investiga uma forma para incluir índices espaciais no SGBD, sem a necessidade de alterações nos SGBDs. Serão considerados índices como as curvas de Hilbert e as Grades Fixas construídos como uma camada sobre a TerraLib. Os desempenhos desses índices serão comparados com o desempenho dos mecanismos nativos oferecidos pelos SGBDs estudados como as B-Trees e R-Trees.

# COMPARISON OF THE R-TREE, GRID FILES AND HILBERT SPACE FILLING CURVES PERFORMANCES FOR SPATIAL QUERIES ON GEOGRAPHICAL DATABASES


## ABSTRACT

The development of spatial databases has been influenced by the research on spatial indexes. There has been a considerable amount of research on multidimensional indexes such as R-trees and its variants. The performance improvements when using spatial indexes for spatial operations led database developers to include R-tree support on their products. Despite these advances, there are database management systems (DBMS) that do not provide spatial indexing in their internal core. Users of this and similar DBMS may want to develop spatial applications, but may be deterred because of performance drawbacks. One obvious solution would be to implement spatial indexes in the DBMS kernel. However, such solution may not be feasible, either because of lack of access to the source code or due to time and cost constraints. This work investigates an alternative way of including spatial indexes in DBMS. We consider indexes such as Hilbert space-filling curves and Fixed Grid developed on top of TerraLib GIS Library. Our main goal is to provide, through this library, the functionalities that are not available in a DBMS that do not include spatial index support and verify the performance between the indexes and the native mechanisms of the SGBDs like B-Trees and R-Trees.

**SUMÁRIO**

Pág.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE SIGLAS E ABREVIATURAS

| | |
|---|---|
| MBR | - Minimum Bounding Rectangle (Retângulo Envolvente Mínimo) |
| BLOB | - Binary Large Object (Campo Binário Longo) |
| RSFC | - Recursive Space-Filling Curves (Curvas de Preenchimento Espacial Recursivas) |
| SIG | - Sistemas de Informação Geográfica |
| SFC | - Space-Filling Curves (Curvas de Preenchimento Espacial) |
| SGBD | - Sistemas Gerenciadores de Bancos de Dados |
| SQL | - Structured Query Language (Linguagem de Consulta Estruturada) |

# CAPÍTULO 1

# INTRODUÇÃO

O desenvolvimento da tecnologia de bancos de dados espaciais tem sido influenciado pelas pesquisas na área de indexação espacial. Vários trabalhos sobre índices multidimensionais podem ser encontrados na literatura como as R-Trees (GUTTMAN, 1984), e suas variantes (TIMOS et al., 1997). Análises comparativas entre diferentes estruturas de indexação podem ser encontradas em (GAEDE E GÜNTHER, 1998) e (SAMET, 1984). A significativa melhora no desempenho de operações espaciais, com a utilização de índices espaciais, tem levado os desenvolvedores de SGBDs a incluir esses mecanismos em seus produtos. Exemplos de produtos bem sucedidos que possuem suporte a dados espaciais são o Oracle Spatial® (RAVADA E SHARMA, 1999) e o PostGIS, uma extensão espacial para o PostgreSQL (ver manual do PostgreSQL disponível em *http://www.postgresql.org/*)

Apesar das vantagens da utilização dos índices espaciais, existem sistemas gerenciadores de bancos de dados (SGBDs) que não possuem suporte a dados espaciais. Como exemplo pode-se citar as versões mais antigas do Microsoft SQL Server. Usuários deste e outros SGBDs similares podem precisar de suporte a dados espaciais. Mas usar SGBDs sem indexação espacial para manipular dados espaciais pode não ser uma boa estratégia, principalmente pelo baixo desempenho. Uma solução óbvia seria incluir o suporte a dados espaciais, implementando os índices no núcleo do SGBD. Esta solução, no entanto, pode ser inviabilizada por dois fatores:

- O código fonte do SGBD pode não ser de domínio público (sistemas proprietários).
- Pode-se ter restrições de tempo e custo.

Este trabalho investiga uma forma alternativa de incluir índices espaciais no SGBD. Dois tipos de índices para operações espaciais foram considerados: as curvas de

preenchimento espacial de Hilbert e as grades fixas. A vantagem da utilização destes índices é que eles podem ser implantados em SGBDs relacionais sem a necessidade de criar extensões internas ao SGBD. Porém, é necessário verificar a viabilidade de utilização destes mecanismos no que se refere ao desempenho. Ao utilizar esses mecanismos (curva de preenchimento espacial de Hilbert e as Grades Fixas) existe alguma perda em termos de desempenho em relação aos métodos usados pelas extensões espaciais? Existe algum ganho? Em que situações esses mecanismos são competitivos?

Para verificar a viabilidade da utilização das curvas de Hilbert e das Grades Fixas como mecanismos de indexação espacial, foi criado um *framework* de indexação no ambiente da biblioteca TerraLib. Esse *framework* consiste da implementação das estruturas de indexação, sua persistência e uso em SGBDs relacionais. Com este *framework* foram realizados experimentos com um conjunto de dados representativos no contexto de aplicações espaciais.

Este documento está organizado da seguinte forma:

- O Capítulo 2 apresenta uma fundamentação teórica sobre indexação espacial e as extensões espaciais dos SGBDs comerciais e gratuitos.
- O Capítulo 3 apresenta o *framework* de indexação espacial criado para realização dos experimentos.
- O Capítulo 4 descreve os dados e experimentos realizados com o *framework*.
- O Capítulo 5 apresenta os resultados e conclusões do trabalho, dando direções para trabalhos futuros.

# CAPÍTULO 2

# REVISÃO BIBLIOGRÁFICA

## 2.1 Bancos de Dados Geográficos

O armazenamento e a recuperação de grandes massas de dados de forma a garantir o desempenho, a integridade e a consistência dos dados são tarefas que já há bastante tempo são delegadas aos SGBDs. O problema, para alguns gerenciadores, existe quando estes dados não são convencionais. Dados geográficos, ou dados que possuem atributos espaciais ou espaço-temporais além de atributos convencionais, necessitam de mecanismos de armazenamento e recuperação especiais. Em se tratando de dados geográficos ainda há a preocupação em armazenar não só os dados alfanuméricos, mas também a componente espacial, que é representada em estruturas geométricas vetoriais como linhas, pontos e polígonos, ou ainda em estruturas matriciais como grades regulares ou triangulares. Devido a singularidade da componente espacial dos dados geográficos e das consultas efetuadas sobre eles, diferentes estratégias têm sido adotadas com o objetivo de obter eficiência e robustez, de forma que os SIGs possam cada vez mais utilizar os recursos dos SGBDs.

De maneira geral, podem ser apontadas duas arquiteturas de SIG's quanto ao armazenamento de dados geográficos (CASANOVA et al., 2005):

- **Arquitetura dual**, em que a componente alfanumérica é armazenada no banco de dados e a componente espacial é armazenada em arquivos externos.
- **Arquitetura integrada**, em que ambas as componentes são armazenadas no banco de dados.

As arquiteturas mencionadas acima têm suas peculiaridades, sendo que cada uma tenta suprir as carências ou falhas da outra. A primeira arquitetura citada, a dual, sugere que a componente espacial e a componente alfanumérica sejam armazenadas separadamente.

Ou seja, os dados alfanuméricos são armazenados em um banco de dados relacional e os dados de geometria, ou componentes espaciais, são armazenados em arquivos com formato proprietário. Existem alguns problemas com esta arquitetura como, por exemplo, a dificuldade em garantir a integridade dos dados e de manipular a componente espacial. Neste caso, a responsabilidade de manipular a componente espacial e garantir a integridade dos dados é transferida para o SIG.

Na arquitetura integrada, a componente espacial e a alfanumérica, são ambas armazenadas no banco de dados. A principal vantagem desta arquitetura é a utilização dos recursos do SGBD para controle e manipulação de objetos espaciais, retirando assim, parte da responsabilidade do SIG de gerenciar os dados. Para armazenar os dados alfanuméricos e a componente espacial em um banco de dados, existem diferentes estratégias na literatura. Baseado na estratégia adotada, a arquitetura integrada pode, ainda, ser subdividida em três tipos (CASANOVA et al., 2005):

- **Baseada em campos longos**, que utiliza Campos Binários Longos (BLOBs) para armazenar a componente espacial dos dados.

- **Com extensões espaciais**, ou seja, utilizando-se as extensões espaciais fornecidas pelos fabricantes.

- **Arquitetura integrada combinada**, formada pela combinação das duas anteriores. Por exemplo, pode-se usar as extensões espaciais para armazenar dados vetoriais e os BLOBs para armazenar dados matriciais (fotos, imagens de satélite, ...).

O problema com a arquitetura integrada baseada em campos longos é que os BLOB's são apenas cadeias de bytes e, portanto, o dado espacial perde o seu significado. Fica a cargo da aplicação, tratar a informação guardada em um BLOB. Outra desvantagem é que não existem métodos de acesso para um BLOB. Por exemplo, não se pode criar um índice sobre uma coluna desse tipo. A linguagem de consulta padrão a bancos de dados relacionais, a *Structured Query Language* (SQL) oferece operadores elementares que não são aplicáveis a campos do tipo BLOB. Ou seja, nesta arquitetura a aplicação é a

única responsável pela captura da semântica dos dados e também pela implementação de operadores espaciais e métodos de acesso.

Alguns fabricantes, atentos à demanda por gerência de dados geográficos, desenvolveram as chamadas extensões espaciais para seus SGBDs. Essas extensões fornecem tipos espaciais, métodos de indexação espacial e operadores espaciais que podem ser acessados e usados pelos SIGs, que, desta forma, repassam ao SGBD a tarefa da gerência dos dados espaciais. A principal vantagem esperada com a gerência dos dados no nível mais interno do SGBD é uma manipulação mais eficiente, baseada em métodos de armazenamento, acesso e de otimização de consultas. Até recentemente as extensões espaciais endereçavam apenas dados vetoriais. Para manipular dados matriciais a maioria dos SIGs continua usando BLOBs, estando sujeitos às desvantagens mencionadas anteriormente. Essa forma de utilização é a arquitetura integrada combinada.

As extensões estão limitadas a poucos SGBDs, dentre as mais conhecidas pode-se citar o Oracle Spatial (RAVADA et al., 1999) e o PostgreSQL/PostGIS (POSTGRESQL, 2008). Destas apenas o PostGIS é de domínio público, ou livre de licença. Além disso, em algumas instituições a implantação do gerenciamento da informação geográfica pode partir do princípio de que se deseja trabalhar com um SGBD legado que não possui extensão espacial, ou seja, ainda existe a necessidade da utilização de SIGs fracamente acoplados, ou independentes do SGBD. Por exemplo, para uma empresa que utiliza um SGBD de um fabricante específico, como o MySQL (YARGER et al., 1999), e que já mantém um banco de dados pode não ser interessante utilizar um SIG que demande um SGBD com extensão espacial como PostgreSQL/PostGIS. Isso se deve, entre outros motivos, ao custo envolvido na migração das informações entre os dois SGBDs. Neste contexto, pode-se verificar a necessidade de que o SIG seja independente do SGBD.

Um SIG deve ser capaz de utilizar uma gama de SGBDs, utilizando os dados alfanuméricos já armazenados sem alterar o esquema de tabelas já implantado, apenas acrescentando novas tabelas e informações. Naturalmente, o SIG deve cuidar da

eficiência na recuperação dos dados, uma vez que SGBDs sem extensão espacial não fornecem métodos de acesso para dados espaciais. Isso implica que o SIG deve fornecer os mecanismos de acesso aos dados espaciais. Por isso, esse trabalho trata da criação de estruturas de indexação para dados espaciais, matriciais e vetoriais, em SGBD's sem extensão espacial.

Para armazenar e recuperar dados espaciais eficientemente existem diversos mecanismos de organização do espaço, conhecidos como métodos de acesso espaciais, que serão brevemente discutidos na próxima sessão.

## 2.2 Estruturas de indexação

Uma forma comum de organizar um conjunto de dados, de forma hierárquica e ordenada é através do uso de árvores de pesquisa. As mais simples e conhecidas são as árvores binárias, como: AVL, *Red-Black e Splay Tree* (COMER, 1979). Estas árvores têm uma propriedade em comum: elas são balanceadas, ou seja, todos os caminhos desde a raiz (o nodo do topo da árvore) até as folhas (nodos finais na árvore) possuem o mesmo comprimento. Essas estruturas permitem que operações como a localização de um elemento sejam executadas em tempo logarítmico – *O(log n)* (CORMEN et al., 2000).A Figura 2.1 ilustra a representação de uma árvore binária balanceada, onde as cidades estão organizadas segundo a ordem alfabética de seus nomes. Esse tipo de estrutura é empregado para uso em memória principal.

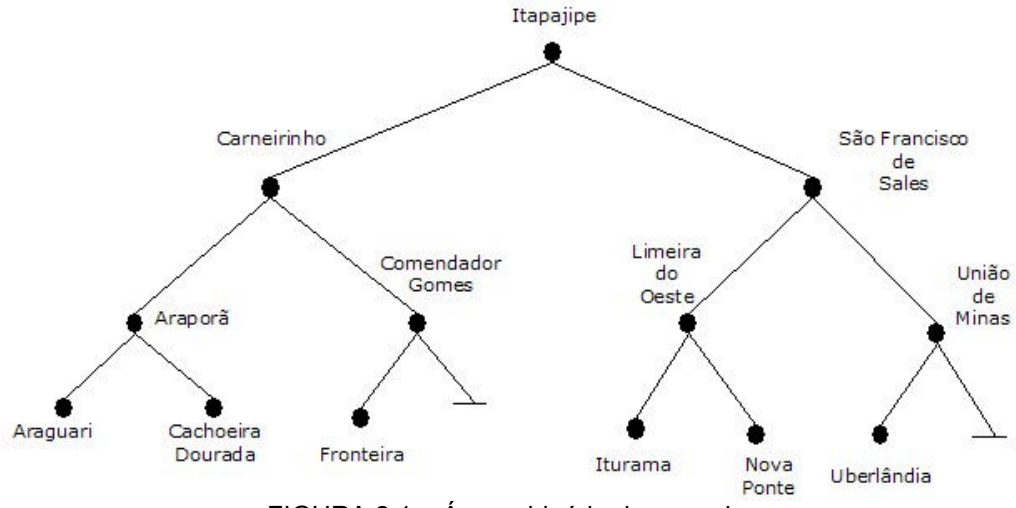



FIGURA 2.1 – Árvore binária de pesquisa.
Fonte: Davis Jr. e Queiroz (2005).

No caso de grandes bancos de dados onde a memória principal pode não ser suficiente para armazenar todos os nós da árvore, é comum armazená-la em disco. Neste caso é desejável uma arquitetura que tente minimizar o acesso a disco, uma vez que acessos a disco são computacionalmente custosos. A forma mais comum, e mais largamente empregada pelos sistemas comerciais atuais, é a representação do índice através de uma árvore B+-Tree (COMER, 1979). A Figura 2.2 apresenta um exemplo de B+-Tree. Essa estrutura tenta minimizar o número de acessos a disco durante o processo de pesquisa agrupando várias chaves em um único nó, denominado de página.

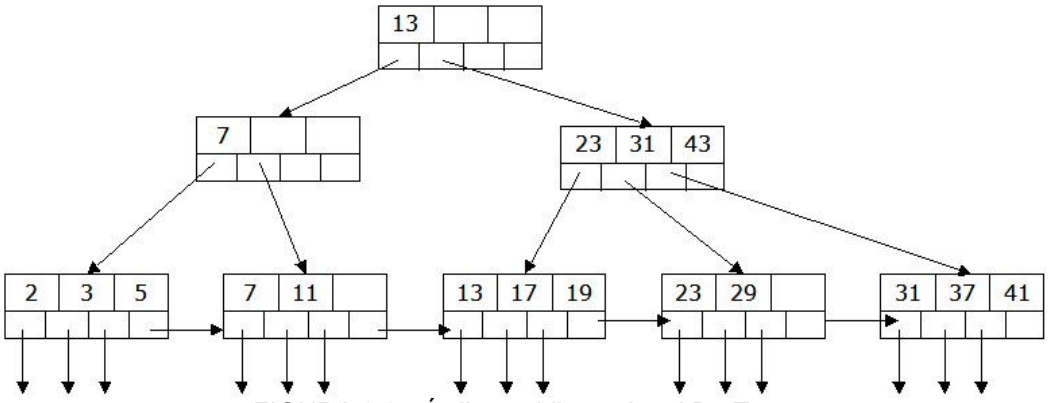



FIGURA 2.2 – Índice unidimensional B+-Tree.
Fonte: adaptada de Garcia-Molina et al. (2001).

Ambas as estruturas, a árvore binária e a B+-Tree, são estruturas unidimensionais. Necessariamente a chave de pesquisa deve ser formada por apenas um atributo ou pela concatenação de vários atributos. Em sistemas que trabalham com informações multidimensionais (cada objeto possui mais que um atributo), como os SIGs, estas estruturas não podem ser diretamente aplicadas. Esses sistemas devem responder a consultas que freqüentemente são espaciais. Entre elas as mais comuns são:

- **Consulta por apontamento:** dado um ponto no espaço, encontre todos os objetos que contém esse ponto (Figura 2.3).

- **Consulta por região:** dada uma região de interesse, encontre todos os objetos que interceptam esta região (Figura 2.4). Geralmente, a região de consulta é retangular, sendo chamada de *window query* (consulta por janela).

- **Junção espacial:** envolve relacionamentos espaciais entre objetos de classes distintas (Figura 2.5).

FIGURA 2.3 – Seleção por apontamento.

FIGURA 2.4 – Seleção por região.

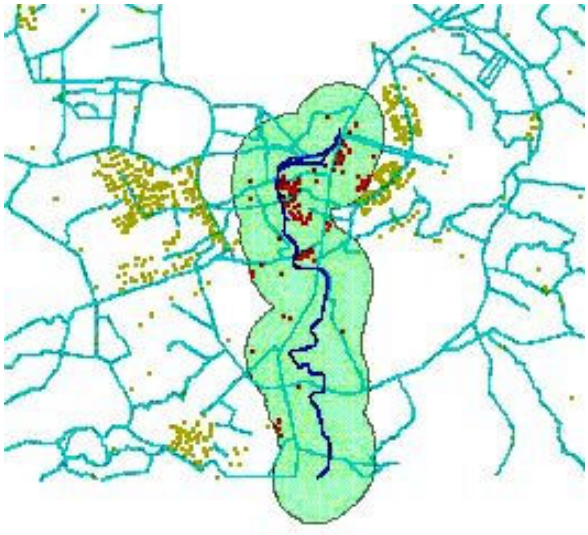


FIGURA 2.5 – Seleção por predicados topológicos.

Para responder a consultas espaciais, essas estruturas unidimensionais não podem ser diretamente aplicadas. A Figura 2.6, apresenta uma área de Minas Gerais onde as sedes municipais estão representadas por pontos no espaço bidimensional. Ao utilizar os métodos de acesso tradicionais como a B+-Tree para indexar esses pontos a chave de pesquisa poderia ser definida pela concatenação das coordenadas em X e em Y. O que se pode verificar, neste caso, é que a ordem definida por este tipo de árvore não preserva a proximidade espacial entre os objetos. A Figura2.7 mostra a ordem definida por este

método. Como pode ser observado, Araporã é, espacialmente, mais próximo de Cachoeira Dourada do que de Fronteira. No entanto esta forma de indexação não é capaz de representar isso, tornando Araporã mais próximo à Fronteira no índice.

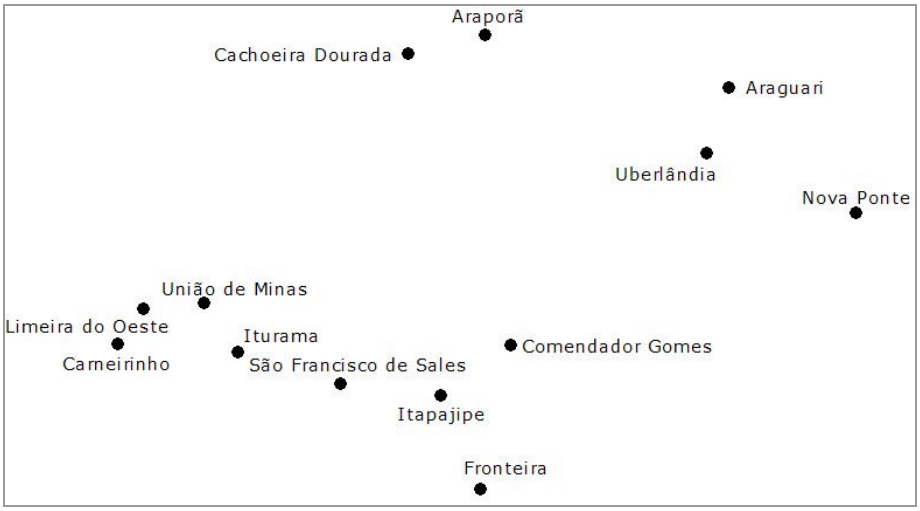



FIGURA 2.6 – Sedes municipais.

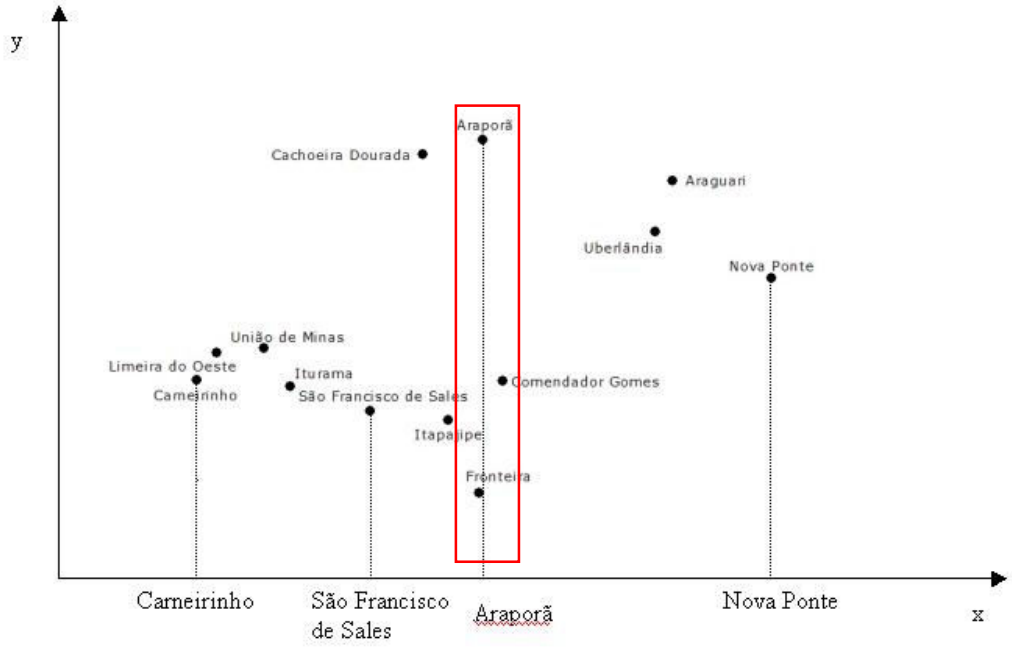



FIGURA 2.7 – Ordem definida pela B+-Tree.

Evidentemente, se a estrutura não preserva a proximidade espacial, então as consultas por janela, por exemplo, não são atendidas da melhor maneira. O caso onde os dados possuem extensão, como os dados poligonais, é ainda pior. A Figura 2.8 mostra isso. Se fosse considerado o retângulo envolvente dos polígonos e fosse construído um índice sobre a concatenação dos valores das coordenadas do canto inferior esquerdo e superior

direito, o resultado teria o efeito mostrado na figura: os objetos 1 e 3, embora mais próximos no espaço, não estarão próximos no índice.

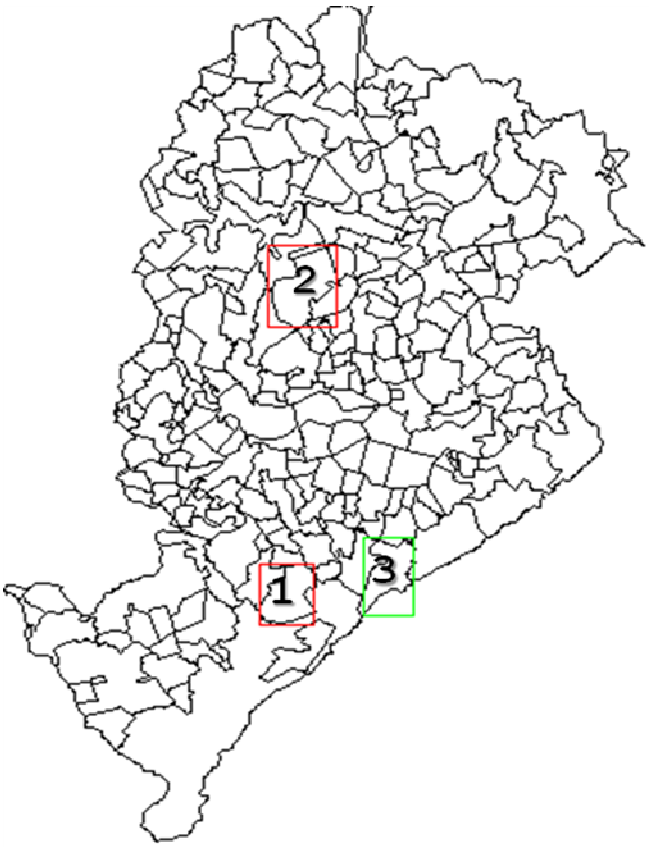



FIGURA 2.8 – Exemplo de polígonos: o índice considera as regiões 1 e 2 mais próximas do que as regiões 1 e 3.

## 2.3 Estruturas de Indexação Espacial

O acesso a dados multidimensionais pode ser muito mais eficiente através do uso de uma classe de métodos conhecidos como *métodos de acesso multidimensionais*. Um exemplo de operação que pode se beneficiar deste tipo de indexação é a operação de vizinhança, que responde à pergunta: "quem são os *K* vizinhos mais próximos do ponto *P1* de coordenadas (*x1*,*y1*)?". Uma solução ingênua seria percorrer todos os pontos medindo e ordenando os pontos por sua distância ao ponto pesquisado e selecionando os *K* mais próximos. Se o número de pontos for muito grande esta solução pode se tornar impraticável pela lentidão do processo. As estruturas de indexação espacial particionam o conjunto de objetos de acordo com sua proximidade espacial, dessa forma reduzindo o

conjunto de busca nas operações espaciais. No exemplo mencionado, o método diminuiria o número de pontos testados para uma pequena parcela dos objetos.

Diversas estruturas ou métodos de indexação são propostos na literatura, como as Quad-Trees (SAMET, 1984) e as R-Trees (GUTTMAN, 1984), cada uma com suas características e comportamentos específicos. Cada estrutura destas organiza o espaço de forma diferente, o que faz com que cada uma tenha um comportamento diferente em relação à distribuição e às geometrias dos dados.

Gaede e Günther (1998) apresentam uma revisão sobre as diversas estruturas de indexação espacial, bem como alguns estudos comparativos entre as mesmas. Uma das conclusões deste estudo é que a escolha do melhor índice é uma tarefa complicada, pois cada um se comporta de uma maneira. Pode-se ter, por exemplo, uma estrutura eficiente em consultas sobre pontos, mas ineficiente em relação a polígonos. Conforme apontado naquele trabalho, os métodos de indexação que se aplicam a geometrias com extensão geralmente consideram uma aproximação da geometria do objeto sendo indexado. Como linhas e polígonos podem assumir formas complexas, o mais comum é que os índices considerem o retângulo envolvente mínimo (*Minimum Bounding Rectangle* - MBR) dessas geometrias. A Figura 2.9 mostra o MBR de um objeto representado por um conjunto de polígonos.

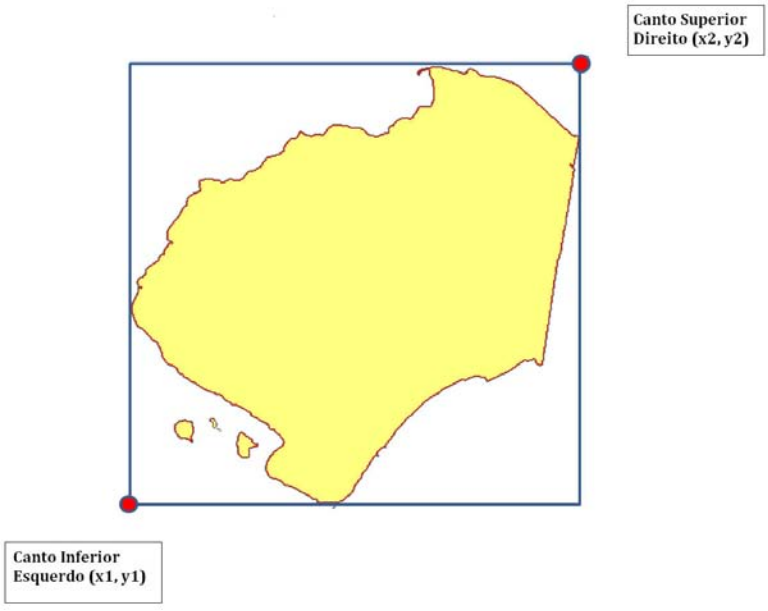



FIGURA 2.9 – MBR de um objeto representado por polígonos.

Entre os principais métodos de acesso espaciais pode-se citar:

- **R-Tree:** este método é uma extensão da B-Tree para o espaço multidimensional, onde os objetos são aproximados através de seus retângulos envolventes, como o exemplo da Figura 2.10. Conforme pode ser visto os nós internos contém entradas da forma <MBR, ptr-nó>, onde o MBR é o retângulo que envolve todos os retângulos do nó filho (apontado por ptr-nó). As entradas dos nós folha contêm o MBR do objeto e um ponteiro para ele. A busca nessa árvore difere da B-Tree pela possibilidade de ser necessário pesquisar pelo dado desejado em mais de uma sub-árvore, caso o retângulo de pesquisa sobreponha mais de um retângulo em cada nível.

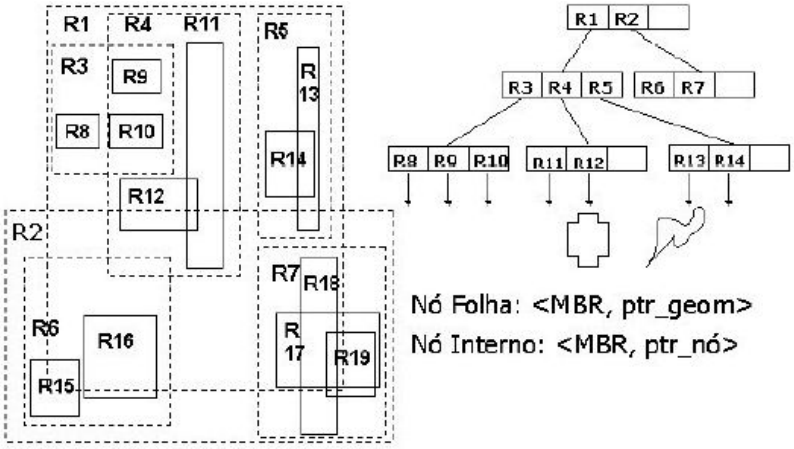



FIGURA 2.10 – Esquema da R-Tree.
Fonte: adaptado de Ciferri e Salgado (2001).

- ***Grid-File:*** este método de indexação particiona o espaço segundo uma grade retangular, onde cada célula é associada a uma página, uma área no disco. Esta associação é feita através de uma matriz bidimensional, conhecida como diretório. A Figura 2.11 ilustra a representação deste tipo de índice utilizada para o caso de pontos. Para objetos mais complexos, tais como polígonos, é utilizada a aproximação pelo MBR, sendo o objeto associado às páginas das células interceptadas por seu MBR. A resolução da grade é um ponto de grande importância para este tipo de indexação.

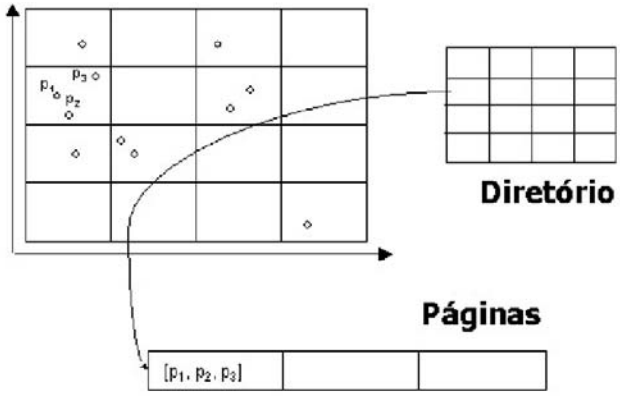


FIGURA 2.11 – Esquema do *Grid-File*.

- ***QuadTree:*** a idéia básica deste método é subdividir o espaço em quadrantes. A árvore é formada por nós que possuem quatro descendentes. Cada descendente é um sub-quadrante do original. Quadrantes que não são mais subdivididos são armazenados em nós folha. Esse método pode ser aplicado a dados no formato matricial, pontos e objetos mais complexos como polígonos (SAMET, 1984). A Figura 2.12 ilustra esse método de indexação.

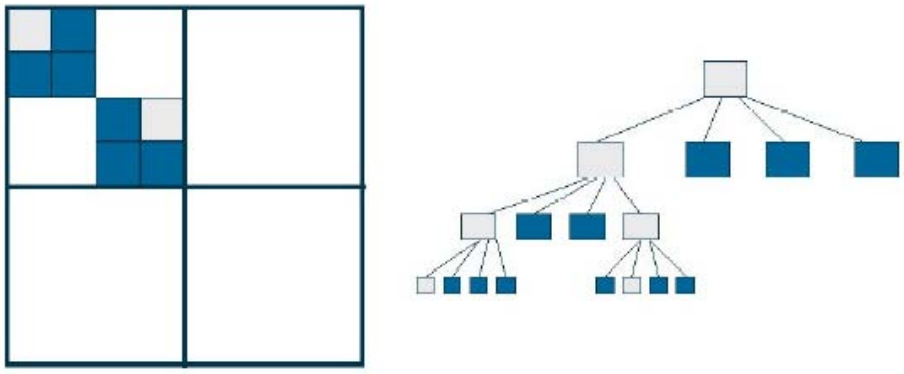
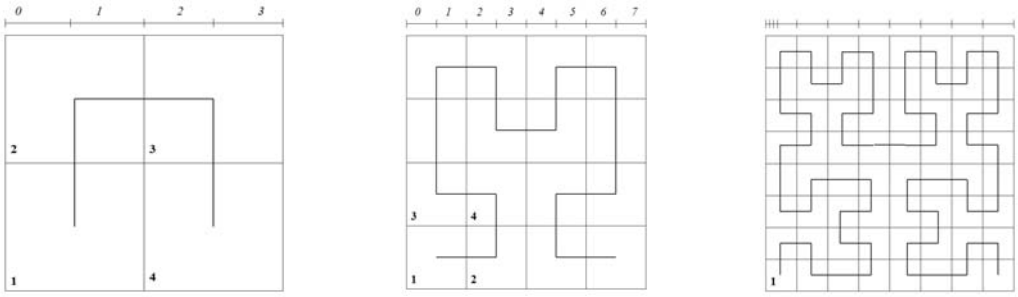


FIGURA 2.12 – Esquema da QuadTree.

## *2.4 Space Filling Curves*

Outra alternativa de indexação espacial é o uso das *Space Filling Curves* (SFC's). Uma SFC é uma curva que passa por todos os pontos do espaço de indexação uma única vez segundo uma determinada ordem (Figura 2.13), podendo ser usada para mapear o espaço multidimensional (espaço das geometrias) para o espaço unidimensional (pontos da curva) (LAWDER E KING, 2000) e (ASANO et al., 1997). Esta é uma característica importante para este estudo, uma vez que possibilita o uso dos mecanismos de indexação tradicionais dos SGBDs, como será visto mais adiante.

Existem várias propostas de SFC's. Para esse trabalho foi considerado o uso da Hilbert SFC, proposta por David Hilbert em 1891 (WIKIPEDIA, 2008). A construção desta curva é mostrada na Figura 2.13. Ela é obtida pelo particionamento recursivo do espaço em quatro quadrantes, onde a cada recursão os quadrantes obtidos no passo anterior são reparticionados segundo um certo algoritmo. A Figura 2.13a mostra a primeira iteração onde o espaço é dividido em quatro sub-quadrantes. A curva correspondente conecta o centro dos sub-quadrantes de forma que intervalos de linhas adjacentes correspondam a sub-quadrantes adjacentes. As Figuras 2.13b e 2.13c mostram mais duas iterações com suas respectivas curvas, essas iterações caracterizam a resolução da curva. Mais adiante será mostrado como os pontos da curva podem ser usados como rótulos dos sub-quadrantes e como são usados para operações de indexação espacial.




(a) (b) (c)

FIGURA 2.13 – Curva de Hilbert.

## 2.5 Bibliotecas de índices espaciais

No contexto da indexação espacial existem pelo menos duas bibliotecas de software livre extensíveis: SAIL (HADJIELEFTHERIOU et al., 2005) e GiST (HELLERSTEIN et al., 1995). A SAIL é uma biblioteca C++ de índices espaciais que combina diversas técnicas de indexação sob uma interface comum, possibilitando a integração desses índices a aplicações. É um *framework* para programadores. Sua proposta é facilitar a incorporação de técnicas de indexação espacial a aplicativos. Ela é baseada em padrões de projeto bem documentados, que promovem reusabilidade e código de melhor qualidade.

A GiST (*Generalized Search Tree*) é usada no SGBD PostgreSQL. Ela define um tipo de índice a partir do qual é possível criar extensões para métodos de indexação específicos. Uma implementação específica de um método requer somente a

implementação de seis métodos – *Consistent*(), *Union*(), *Compress*(), *Decompress*(), *Penalty*(), *PickSplit*() – que encapsulam o comportamento do índice e dos objetos usados como chave.

Essas duas bibliotecas podem ser utilizadas de duas formas:

- A primeira, integrando-as ao núcleo do SGBD. Isso requer um esforço considerável, podendo até mesmo ser um impedimento para o caso de SGBDs proprietários.

- A segunda, a aplicação poderia utilizá-las, gerenciando o índice fora do SGBD. Neste caso, o SGBD não saberia nada a respeito do índice. Este tipo de integração tem a desvantagem de inviabilizar o uso de consultas em SQL que envolvam tanto atributos espaciais quanto alfanuméricos em conjunto. A aplicação fica responsável por interpretar a parte espacial das consultas.

Este trabalho apresenta um framework alternativo, que pode ser implantado em qualquer SGBD. A idéia é que se possa criar índices espaciais em SGBDs sem suporte a dados espaciais, usando os mecanismos disponibilizados por eles, sem a necessidade de alterações em seu núcleo. Os detalhes deste framework serão abordados no próximo capítulo.

# CAPÍTULO 3

# FRAMEWORK DE INDEXAÇÃO ESPACIAL

O *framework* de indexação espacial foi desenvolvido no ambiente da biblioteca TerraLib (CASANOVA et al., 2005). A seguir, será apresentado o contexto da biblioteca no qual o trabalho foi desenvolvido.

## 3.1 Terralib

A TerraLib é uma biblioteca de código livre com classes e funções de SIG. Seu propósito é fornecer aos programadores um ambiente composto de um conjunto vasto de funcionalidades, para armazenamento e manipulação de dados geográficos.

Uma das características de projeto da TerraLib é o uso extensivo dos SGBDs relacionais, com arquitetura integrada, para a manipulação dos dados espaciais. SGBDs relacionais armazenam informações em tuplas. Tipicamente, na arquitetura integrada, uma tupla representa o conjunto de informações sobre um determinado objeto geográfico. Uma tabela armazena as informações de um conjunto de objetos geográficos semanticamente similares, incluindo sua componente espacial, essas tabelas formam o modelo de armazenamento da TerraLib. Além das tabelas que armazenam os dados propriamente ditos, a biblioteca também cria um conjunto de tabelas de metadados. O conjunto dessas tabelas forma o esquema conceitual da TerraLib.

Um outro requisito no projeto da TerraLib é que fosse possível construir o modelo de dados e o modelo conceitual em diferentes SGBDs, para isso, a biblioteca utiliza o conceito de drivers. Um driver é uma implementação concreta da interface abstrata com bancos de dados fornecida pela TerraLib, para um SGBD em particular, resolvendo as diferenças de compatibilidade entre os SGBDs. Além disso, quando se trata do modelo de armazenamento de dados, os drivers se beneficiam das vantagens das extensões espaciais disponibilizadas por alguns fabricantes como as R-Tree no Oracle Spatial ou a R-Tree sobre GiST no PostgreSQL.

Atualmente, nos bancos sem extensão espacial, como é o caso das versões mais antigas do SQL Server ou Oracle sem extensão espacial, os drivers utilizam os métodos de indexação convencionais desses SGBDs sobre as aproximações das geometrias, usando a estratégia da concatenação de coordenadas. Conforme apontado na Seção 2.2, essa estratégia pode apresentar problemas no que tange às aplicações geográficas. Neste trabalho serão utilizados as SFCs e as grades fixas, na busca de uma alternativa geograficamente correta e mais eficiente em termos de desempenho, para a recuperação de dados espaciais em bancos de dados relacionais sem extensão espacial. Para isso, foi desenvolvido um Framework de Indexação Espacial que será descrito na seção seguinte.

## 3.2 O *framework* de indexação espacial

O *Framework* de Indexação Espacial é um arcabouço composto de uma arquitetura de classes que funciona como uma camada de indexação e de uma extensão ao modelo conceitual e de dados da TerraLib para armazenar os dados de indexação. Nesse trabalho, foram implementados dois métodos de indexação: grades fixas (*fixed grid*) e curvas de preenchimento de espaço, em particular a curva de Hilbert. Estes métodos foram escolhidos porque, diferentemente da grande maioria das estruturas de indexação apresentadas na literatura, eles podem ser implementados como uma camada intermediária, sem a necessidade de alterações no núcleo do SGBD. Além disso, os métodos podem ser facilmente extendidos para o funcionamento em *n*-dimensões. Isso é interessante no caso de dados espaço-temporais.

O índice é armazenado no próprio SGBD em um esquema de tabelas e utiliza o próprio mecanismo de indexação convencional do SGBD. As consultas espaciais passam pela camada de indexação onde são transformadas em consultas SQL tradicionais otimizadas para responder ao critério espacial e a seguir são enviadas ao SGBD. A seguir serão apresentados os detalhes dos dois métodos testados.

### 3.2.1 Grade Fixa (*Fixed Grid*)

Esse método parte da divisão de um retângulo envolvente contendo todos os objetos sendo armazenados em células de tamanho fixo. Cada célula intercepta um conjunto de

objetos e um objeto pode interceptar mais de uma célula, como é ilustrado na Figura 3.1. Cada célula deve ter uma identificação única, por exemplo, pela combinação de sua posição em linha e coluna na matriz de células que divide o retângulo envolvente sendo indexado.

Para serem usadas como indexadores espaciais, cada uma das células serve como um diretório que contém todos os objetos que a interceptam, e sabe-se, por construção, qual a sua extensão geográfica. Assim, o primeiro passo em uma consulta espacial é a localização das células nas quais o objeto está indexado e, em um segundo passo, busca-se sequencialmente o objeto específico sendo consultado.

O tamanho, ou resolução, das células é um parâmetro importante do método, pois como pode ser visto na Figura 3.1, da resolução depende o número de objetos que interceptam a célula e também o número de células que um objeto intercepta. Diminuir a resolução significa criar células maiores, diminuir o número total de células, aumentar o número médio de objetos por célula e diminuir o número de células diferentes que um objeto intercepta. A desvantagem de se adotar uma resolução muito baixa é que, ao executar uma consulta, o primeiro passo resultará em um grande número de objetos que deverão ser testados exaustivamente para ver se atendem o critério especificado.

Aumentar a resolução significa criar células menores, aumentar o número total de células, diminuir o número médio de objetos por célula e aumentar o número de células diferentes as quais um objeto intercepta. A vantagem de se adotar uma resolução grande é que no primeiro passo da consulta serão obtidos poucos objetos que deverão ser testados exaustivamente para ver se atendem ao critério especificado. A desvantagem é que o aumento no número de células, que finalmente representa o índice, implica na ocupação de mais espaço de para armazenar o índice.

Para contornar as desvantagens relacionadas a resolução da grade, optou-se por uma implementação de uma grade fixa com pelo menos três resoluções diferentes como mostrado na Figura 3.1 . Cada resolução é chamada de nível. A escolha das melhores resoluções a serem usadas em cada nível, fica a cargo do usuário. Esta escolha é feita

com base na distribuição e geometrias dos dados. Cada nível é definido por uma função aplicada aos parâmetros do nível anterior, num processo que será descrito no Capítulo 4.

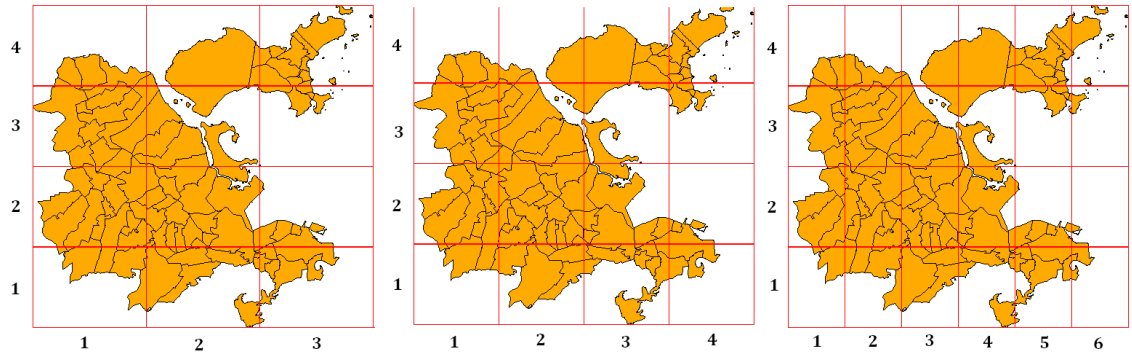



FIGURA 3.1 – Grade Fixa Multinível.

Como mencionado anteriormente o índice é armazenado em tabelas, uma para cada nível, dentro do banco de dados. Cada tabela relaciona o identificador do objeto e a identificação da célula, através de sua posição em termos de linhas e colunas da matriz de células daquele nível. Esses dois últimos campos são indexados por B-Tree formando uma chave concatenada. A Figura 3.2 mostra a indexação de um objeto por uma grade fixa de 4 linhas por 4 colunas.

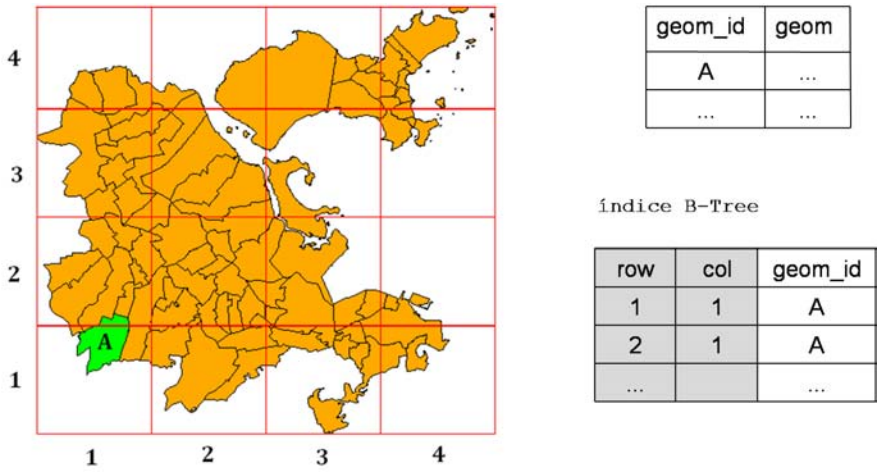



FIGURA 3.2 – Exemplo de indexação usando Grades Fixas

Na tabela do índice de grade, é armazenada a relação entre o objeto e a célula da grade. No exemplo da Figura 3.2, o objeto *A* é associado a duas células da grade de rótulos: (1,1) e (2,1). Essa associação é feita através do MBR do objeto. A partir do MBR do objeto, calculam-se as células da grade que o interceptam. Cada par (linha, coluna) do resultado é uma célula interceptada pelo MBR do objeto e é armazenada na tabela. No

final do processamento de todos os objetos do conjunto de dados, cria-se um índice B-tree sobre as colunas *row* e *col* desta tabela. Pode se observar que objetos espacialmente próximos interceptam células cujas identificações são próximas na linha e na coluna. Essa proximidade é capturada pela B-Tree.

Para responder uma consulta, a camada de indexação determina, o conjunto de pares (linha, coluna), interceptadas pelo retângulo de seleção, ou que contém o ponto (no caso de consultas por apontamento). Essa etapa do processamento define as células que devem ser buscadas no banco de dados, funcionando como um filtro espacial. Uma vez encontradas as células, pesquisa-se pelos objetos que estão relacionados a elas. A última etapa da seleção de objetos é determinar, em memória, os objetos que se enquadram no critério definido pela consulta.

A Figura 3.3 apresenta o modelo de classes utilizado na implementação deste tipo de índice. As classes destacadas em vermelho são responsáveis pelas respostas dos índices às consultas espaciais. Dada uma consulta espacial – por área ou apontamento – os objetos dessas classes calculam os identificadores das células que foram interceptadas por aquela consulta. As classes marcadas em verde fazem a manutenção dos índices no SGBD: criam, excluem e alteram as tabelas de índices e metadados de índices. Outra função desses objetos é criar a consulta em linguagem SQL para a busca dos objetos no SGBD: os objetos das classes em vermelho calculam os identificadores das células e o objeto do tipo TeMultiLevelFixedGridIdxDb, cria a restrição em linguagem SQL, para recuperar os objetos.

Na próxima seção será apresentado os detalhes do método Hilbert SFC.

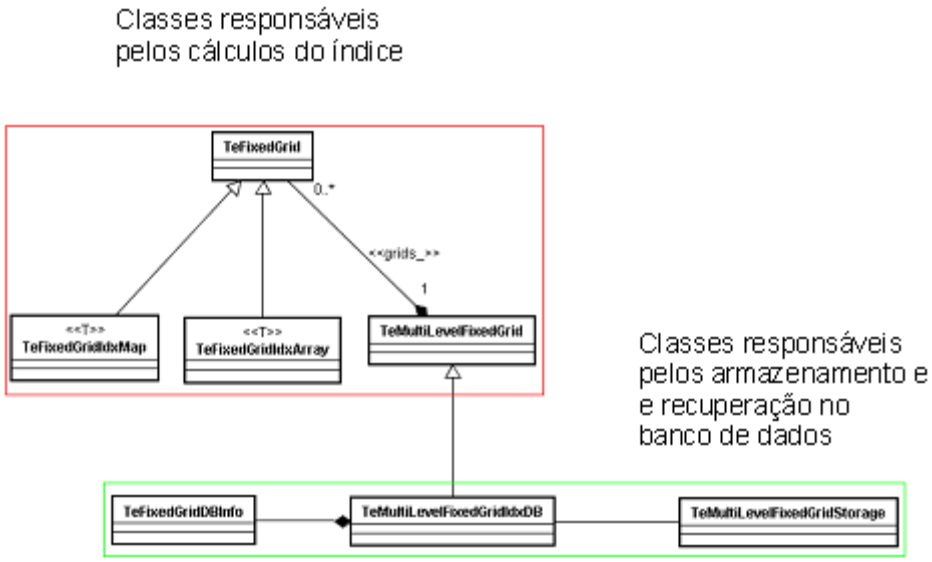



FIGURA 3.3 – Diagrama de Classe para a Grade Fixa Multinível.

### *3.2.2 Hilbert Space Filling Curve*

Esse método é semelhante ao método das grades fixas, a diferença está na forma como o espaço é particionado, como mostra a Figura 3.4. A partir do MBR do conjunto de dados, calcula-se a curva de Hilbert recursivamente. A Figura 3.4 mostra, da esquerda para a direita, as curvas obtidas após uma, duas e três recursões.

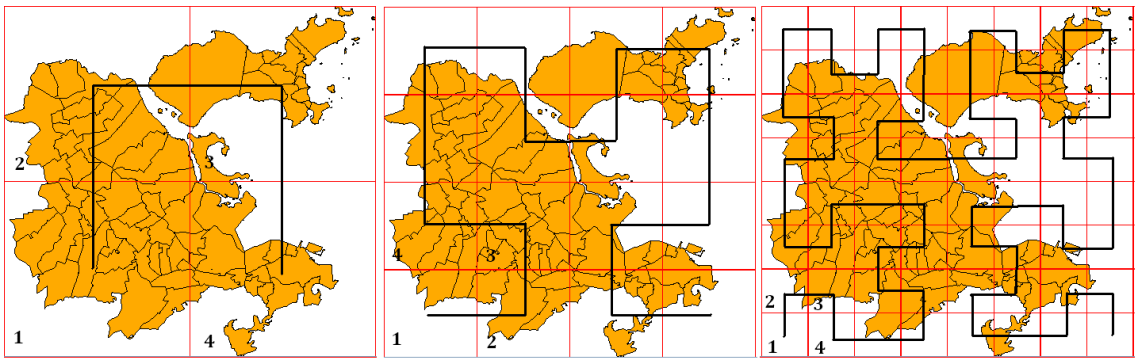



FIGURA 3.4 – Curvas de Hilbert com diferentes níveis de recursão construídas sobre um conjunto de dados geográficos

O número de recursões tem influência direta na resolução da grade e é um parâmetro importante. Aumentar a resolução, ou usar mais iterações, resulta na criação de células menores, aumento no número total de células, diminuição do número médio de objetos por célula e aumento no número de células diferentes as quais um objeto intercepta. O número de linhas é sempre igual ao número de colunas. O número total de células é igual a $4^i$ para i igual ao número de recursões utilizado na construção da curva.

Cada ponto da curva identifica a célula por onde passa, ou seja, o índice nesse caso é armazenado em tabelas que guardam o relacionamento entre a célula e os identificadores dos objetos que a interceptam. A Figura 3.5 mostra um objeto indexado por uma curva de Hilbert.

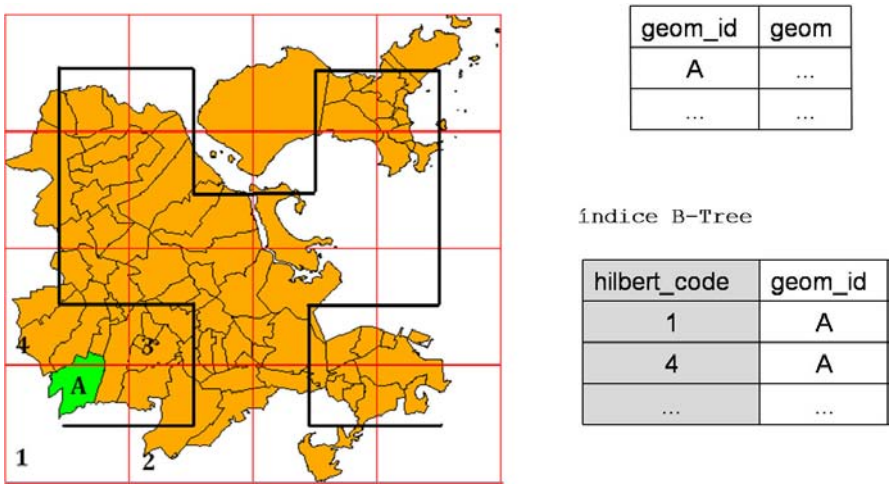



FIGURA 3.5 – Exemplo de indexação usando a curva de Hilbert.

No exemplo da Figura 3.5, o objeto *A* é associado às células de rótulo *1* e *4*. Assim como as grades fixas, o índice da curva de Hilbert é gerado a partir do cálculo dos rótulos das células interceptadas pelo MBR do objeto a ser indexado. Esse processamento é feito para todos os objetos do conjunto de dados. No final tem-se uma tabela que relaciona os objetos, através de seus identificadores, às células da grade, através do rótulo da célula na curva de Hilbert. A coluna com os códigos de Hilbert são indexadas usando a B-Tree.

Para responder uma consulta, a camada de indexação determina os rótulos das células interceptadas pelo retângulo de seleção, ou que contém o ponto (no caso de consultas por apontamento). Assim como as grades fixas, essa etapa do processamento define as células que devem ser buscadas no banco de dados, funcionando como um filtro espacial. Uma vez encontradas as células, pesquisa-se pelos objetos que estão relacionados a elas.

Para computar o código de Hilbert, foi utilizado o algoritmo recursivo mostrado na Figura 3.6, que pode ser entendido como uma interpretação da máquina de estados apresentada em (LAWDER et al., 2000).

**Algorithm 1:** Encontra os rótulos dos sub-quadrantes que interceptam um dado MBR.

**Name:** HilbertCoding

**Input:**
- *W-MBR:* retângulo do espaço que será particionado
- *MBR:* O MBR de um objeto sendo mapeado ou um retângulo de seleção
- *State:* estado atual da curva
- *Order:* ordem da curva sendo considerada na recursão

**Output:**
- *Hilbert-code-list:* Lista com os códigos de Hilbert para o dado MBR

**Constants:**
- CurveOrder: Ordem da curva usada para indexar os objetos
- TransitionStates: tabela que mapeia: estado x quadrante →: próximo-estado, representando a máquina de estado.

**Globals:**
- HilbertCode: Variável auxiliar que conterá o valor de código Hilbert para cada sub-quadrante visitado pelo algoritmo

```
1: if Order <= CurveOrder
2: | Q ← split W-MBR in for quadrants: SO, NO, NE, SE
3: | for each Qi in Q
4: | | if intersects(MBR, Qi)
5: | | | HilbertCoding(Qi, MBR,
6: | | |               TransitionStates[State][Q.name()],
7: | | |               Order + 1)
8: | | else
9: | |    update(HilbertCode)
10:else
11:   append(Hilbert-code-list, HilbertCode)
```

FIGURA 3.6 – Algoritmo de cálculo dos rótulos da curva de Hilbert.

O algoritmo mostrado na Figura 3.6 pode ser usado tanto na etapa de indexação dos objetos quanto na criação das consultas espaciais. Na indexação de um objeto invoca-se o algoritmo passando como parâmetros iniciais o retângulo envolvente do conjunto de dados (união dos MBRs de todos os objetos contidos no conjunto), o MBR do objeto a ser indexado, o estado inicial da curva de Hilbert e o valor de ordem 1 (iteração inicial). No algoritmo, a função que divide (split) o retângulo de uma partição (W-MBR) deve retornar uma lista de quadrantes (Q) ordenados de acordo com o estado em que a curva se encontra. Isto é, para que a tabela com a transição de estados (TransitionStates) possa ser corretamente utilizada. Outro ponto importante do algoritmo é quando o retângulo

de pesquisa (MBR) não intercepta um dado quadrante (Qi), neste caso é possível realizar um corte na recursão, atualizando o valor do código de Hilbert para o próximo quadrante a ser testado (função update).

É interessante notar que as células criadas com a SFC e com a grade fixa preservam a proximidade espacial entre os objetos com os quais interceptam, mas de forma diferente. Na grade fixa, as células são ordenadas pelas colunas (da esquerda para direita) e em seguida pelas linhas (de baixo para cima). Na SFC, a adjacência entre as células é dada pela ordem imposta pela SFC. Disso resulta que no caso da SFC é possível transformar uma consulta espacial em uma SQL onde a cláusula WHERE contém busca por intervalos de valores nos rótulos das células. Dependendo da consulta, esta pode ser resolvida por um único intervalo. No caso das grades fixas, cada célula candidata deve ser explicitamente inserida na cláusula WHERE da SQL.

A Figura 3.7 apresenta o modelo de classes utilizado na implementação das classes de índices SFC. Note que cada SFC implementada é uma especialização da classe TeSpaceFillingCurve , ou seja, outras SFCs podem ser implementadas seguindo este modelo.

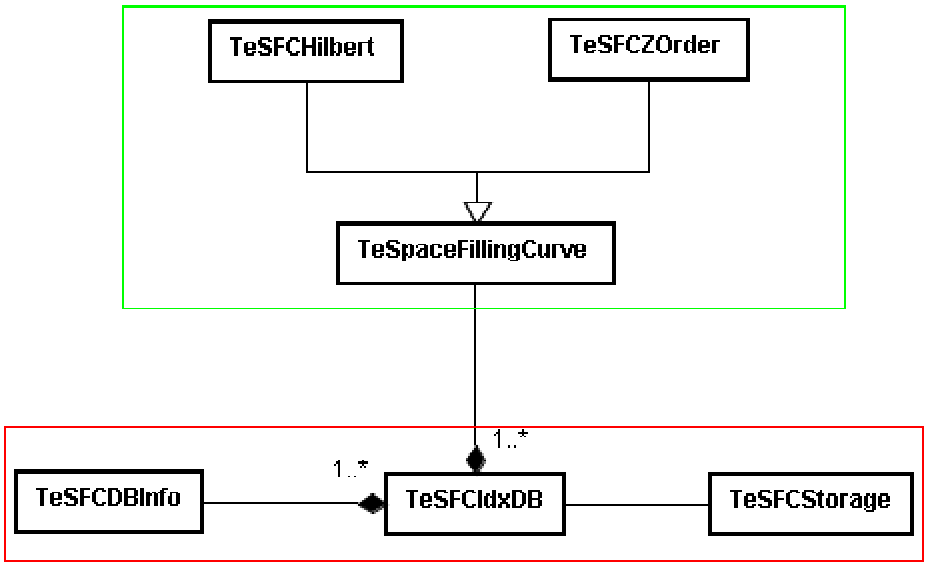



FIGURA 3.7 – Diagrama de Classes das SFCs.

As classes destacadas em verde são responsáveis pela resposta do índice às consultas espaciais. As classes em vermelho fazem a manutenção dos índices nos SGBDs: criam,

excluem e alteram as tabelas de índices e metadados. Outra função desses objetos e criar a consulta em linguagem SQL para a busca dos objetos no SGBD: os objetos das classes em verde calculam os identificadores das células e o objeto do tipo TeSFCIdxDB, cria a restrição em linguagem SQL, para recuperar os objetos.

O próximo capítulo apresenta os experimentos realizados com estes métodos de indexação, visando responder às perguntas apresentadas no Capítulo 1.

# CAPÍTULO 4

# EXPERIMENTOS

Diversos *benchmarks* de desempenho de sistemas de bancos de dados podem ser encontrados na literatura, como em Bitton, DeWitt e Turbyfill (1983), Ciferri e Salgado (2000), Quak, Oosterom e Tijssen (2002), Stonebraker et al. (1993), Theodoridis (2003) e Werstein (1998). Cada um deles tenta avaliar o desempenho de determinados tipos de consultas, sobre os SGBDs testados.

O objetivo deste trabalho não é fazer um comparativo entre os diferentes SGBDs. Nosso objetivo é verificar o desempenho dos mecanismos de indexação investigados – SFCs e Grade – em relação ao método utilizado na TerraLib para sistema de bancos de dados convencionais. Outro objetivo, é medir a competitividade destes métodos em relação aos métodos de indexação espaciais presentes nas extensões espaciais.

Como parâmetro de medida da eficiência, foi tomado o tempo de resposta às consultas espaciais. Para verificar esse tempo nas diferentes estratégias de recuperação, foram feitos testes exaustivos de simulação de seleções por janela (*range query*). A bateria de testes foi criada de forma a simular as demandas de uma aplicação SIG real, normalmente utilizada por qualquer tipo de usuário. Ex: visualizar uma determinada área do espaço, selecionar uma coleção de objetos, entre outros.

Para os testes foram utilizados dois SGBDs: PostgreSQL com extensão espacial PostGIS e MySQL (sem extensão espacial). Para o primeiro SGBD, avaliou-se a R-tree sobre GiST, a B-tree usando a estratégia convencional da TerraLib (valores de chave concatenados), a Grade Fixa e a Hilbert SFC. Para o segundo, avaliou-se a B-tree usando a estratégia convencional da TerraLib, a Grade Fixa e a Hilbert SFC. Os testes ainda foram realizados numa máquina servidora com um processador Pentium 4 Hyper Threading - 3 Gigahertz, 1 GigaByte de memória principal e um disco rígido de 100 GigaBytes. O sistema operacional usado foi o Linux Mandriva 2008, executando os

serviços básicos. Ainda foram feitos testes complementares num ambiente com sistema operacional Microsoft Windows XP. A criação dos índices é feita após a inserção de todos os objetos no SGBD. Esses índices ainda são reorganizados e balanceados por aplicativos fornecidos pelos SGBDs, o VACUUM no caso do PostGIS e a função ANALYZE no caso do MySQL. A idéia é tentar diminuir o impacto das inserções que, em geral, tendem a reduzir a organização da estrutura de indexação e interferir no desempenho.

Como carga de trabalho, foram utilizados quatro conjuntos de dados:

1. Mapa de Municípios Brasileiros, com aproximadamente 5700 polígonos (Figura4.1). As geometrias deste mapa são simples, com poucos vértices e buracos por polígono. No entanto, a forma e as dimensões das geometrias são heterogêneas.

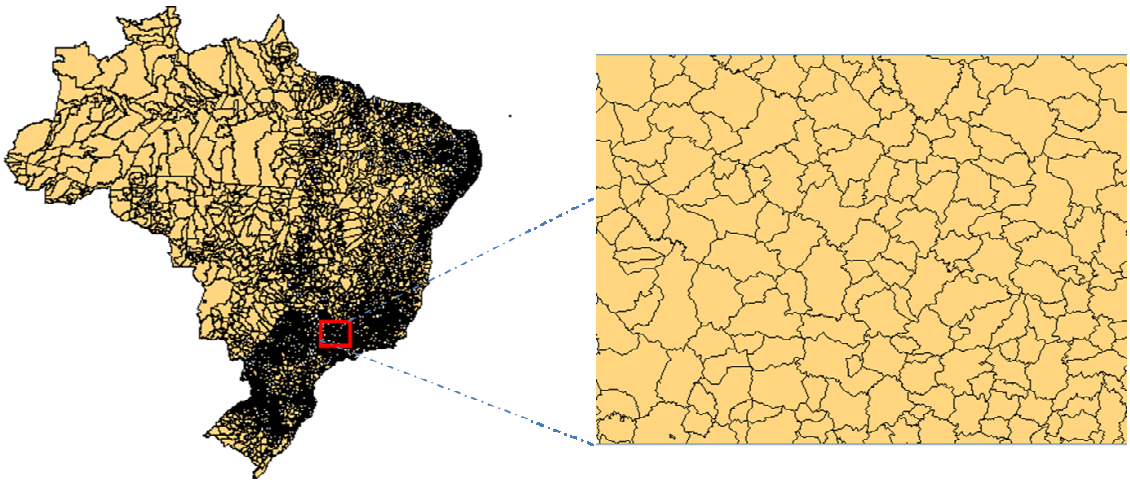


FIGURA 4.1 – Municípios Brasileiros

2. Mapa de Setores Censitários de São Paulo, com aproximadamente 12.000 polígonos (Figura 4.2). As geometrias deste mapa são simples, com poucos vértices e buracos por polígono. A forma e as dimensões das geometrias são heterogêneas.

3. Mapa de Lotes, com aproximadamente 120.000 polígonos (Figura 4.3). As geometrias deste mapa são simples, com poucos vértices e nenhum buraco

por polígono. A forma e as dimensões das geometrias são bastante homogêneas.

4. Mapa do Sistema de Monitoramento do Desflorestamento (PRODES), com aproximadamente 1.100.000 polígonos (Figura 4.4). As geometrias deste mapa são complexas, com muitos vértices e buracos por polígono. A forma e as dimensões das geometrias são bastante heterogêneas.

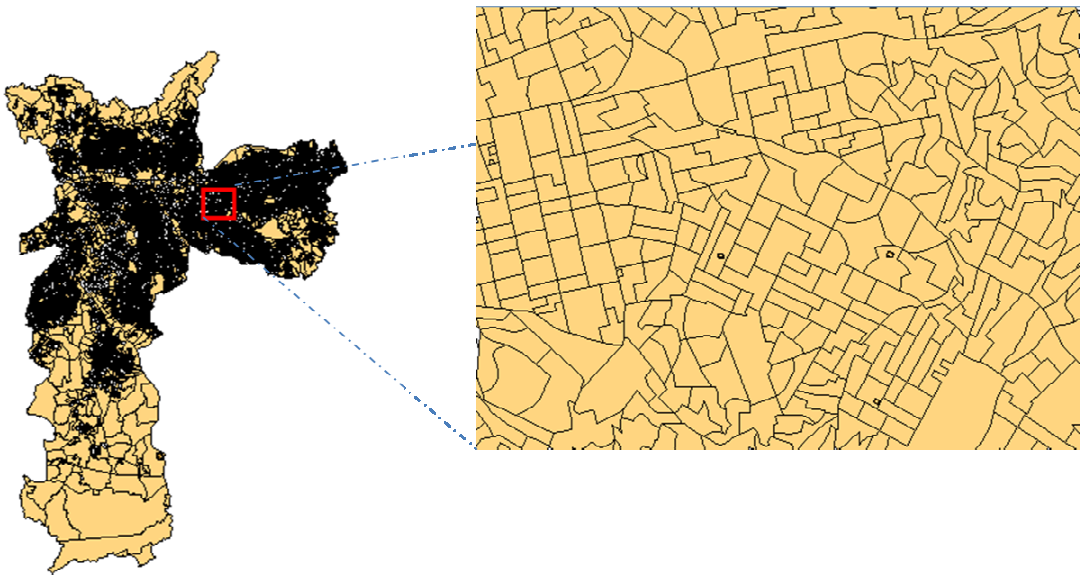


FIGURA 4.2 – Setores Censitários São Paulo

FIGURA 4.3 – Lotes

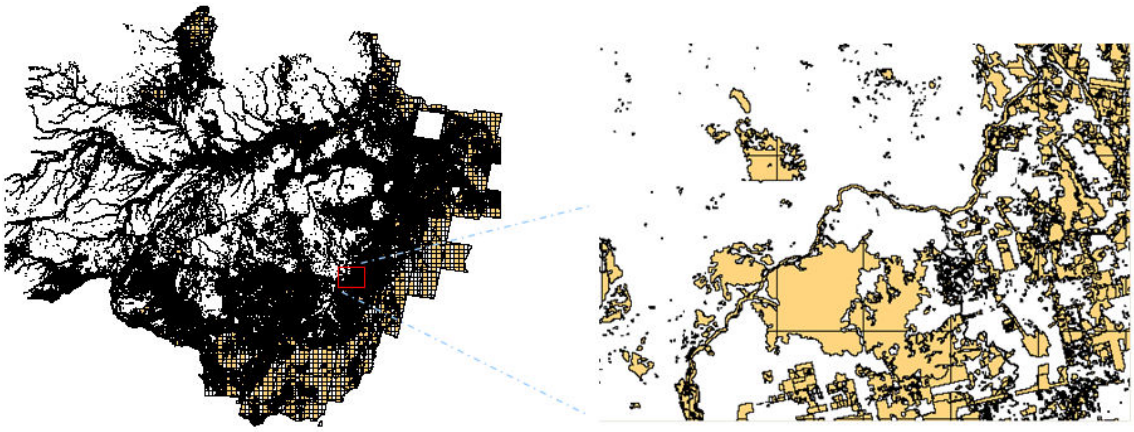


FIGURA 4.4 – Dados Sistema de Monitoramento do Desmatamento

Esses conjuntos de dados foram usados porque possuem distribuições e geometrias variadas. Os dados são reais, de fenômenos comuns encontrados no dia a dia, e que são manipulados por SIG's em diversos segmentos, como sistemas de análise do desmatamento, sistemas de cadastros de prefeituras, sistemas para armazenamento de dados censitários, etc. As tabelas 4.1 e 4.2 mostram algumas medidas coletadas para tentar descrever os dados.

TABELA 4.1 – Estatísticas do conjunto de municípios do Brasil

| | |
|---|---|
| Número de polígonos | 5794 |
| Número de polígonos com buracos | 4 |
| Número de buracos do polígono com maior Número de buracos | 1 |
| Número médio de vértices por polígono | 98 |
| Número de vértices do polígono com maior Número de vértices | 1921 |
| Área media por polígono | 0,1226580399 |
| Polígono de maior área | 13,0595492797 |

TABELA 4.2 – Estatísticas do conjunto de setores censitários de São Paulo

| | |
|---|---|
| Número de polígonos | 13459 |
| Número de polígonos com buracos | 292 |
| Número de buracos do polígono com maior Número de buracos | 9 |
| Número médio de vértices por polígono | 15 |
| Número de vértices do polígono com maior Número de vértices | 249 |
| Área media por polígono | 113190,08382 |
| Polígono de maior área | 77791061,34003 |

TABELA 4.3 – Estatísticas do conjunto dos dados do PRODES

| | |
|---|---|
| Número de polígonos | 1052684 |
| Número de polígonos com buracos | 15898 |
| Número de buracos do polígono com maior número de buracos | 747 |
| Número médio de vértices por polígono | 37 |
| Número de vértices do polígono com maior número de vértices | 21945 |
| Área media por polígono | 0.0001267896 |
| Polígono de maior área | 0.0625000000 |

Todos os dados foram importados para bancos de dados no formato da TerraLib. Nestes bancos criou-se uma série de tabelas para materializar os índices de cada método avaliado. Para a Hilbert SFC foram usadas curvas com três ordens (iterações) diferentes: curva de ordem 6 (H6), curva de ordem 7 (H7) e curva de ordem 8 (H8). No caso das Grades Fixas, foram criadas 6 tabelas de índices com resoluções diferentes:

- **G1:** corresponde à resolução de grade mais fina. Seja *w(G1)* a resolução de **G1** sobre o *eixo x, h(G1)* a resolução de **G1** sobre o *eixo y, w(p[i])* a largura do MBR do i-ésimo polígono *p* e *h(p[i])* a altura do MBR do i-ésimo polígono *p*. Para cada conjunto de dados composto por *n* objetos, as resoluções *w(G1)* e *h(G1)* são definidas pelas fórmulas:
  - $w(G1) = \sum_{i=0}^{n} w(p[i])$
  - $h(G1) = \sum_{i=0}^{n} h(p[i])$
- **G2:** grade mais grosseira, cuja resolução é igual a 2 x G1.
- **G3:** 4 x G1
- **G4:** 6 x G1
- **G5:** 8 x G1
- **G6:** 10 x G1

Sobre cada conjunto de dados da carga de trabalho foram aplicados 1000 testes. Cada teste consiste de um retângulo envolvente, escolhido aleatoriamente sobre o espaço de interesse (o conjunto dos dados). A partir desse retângulo envolvente, gera-se uma consulta que será submetida ao SGBD. Essas consultas são geradas para cada índice espacial analisado no experimento (Hilbert, Grades, B-tree, R-tree). O tempo de resposta do SGBD a cada consulta é, então, medido.

A seguir serão apresentados os resultados e as conclusões obtidas a partir dos experimentos descritos neste capítulo.

# CAPÍTULO 5

# ANALISE DOS RESULTADOS

A Tabela 5.1 mostra os resultados obtidos para os experimentos realizados. Nela, são apresentadas as médias dos tempos dos testes sobre cada conjunto de dados, tomados em milisegundos, em cada SGBD. A linha rotulada com o nome B-tree, significa o índice B-tree construído sobre a concatenação dos valores de coordenadas das extremidades opostas dos retângulos envolventes dos objetos, como mostrado nas Seções 2.2 e 3.1. As linhas G0 a G5 representam o método das Grades Fixas, com as resoluções mostradas no capítulo anterior. As linhas H6, H7 e H8 representam os resultados do método Hilbert SFC de ordem 6, 7 e 8 respectivamente. As células da tabela marcadas com 'x' significam que o método não foi avaliado para o SGBD em questão.

TABELA 5.1 – Resultado dos Experimentos

| | **Municípios do Brasil** | | **Setores de São Paulo** | | **Lotes** | | **Prodes** | |
|---|---|---|---|---|---|---|---|---|
| | **Media Tempos(ms)** | | **Media Tempos(ms)** | | **Media Tempos(ms)** | | **Media Tempos(ms)** | |
| **Método** | **PostGIS** | **MySQL** | **PostGIS** | **MYSQL** | **PostGIS** | **MySQL** | **PostGIS** | **MySQL** |
| **B-tree** | 7,788 | 73,145 | 17,393 | 198,774 | 288,644 | 1748,276 | 650,464 | 34927,626 |
| **R-tree** | 28,820 | x | 10,829 | x | 20,547 | x | 91,257 | x |
| **G0** | 19,389 | 16,682 | 18,564 | 25,556 | 82,684 | 139,653 | 120,219 | 426,736 |
| **G1** | 10,342 | 14,393 | 9,847 | 18,638 | 53,538 | 82,663 | 46,729 | 68,055 |
| **G2** | 8,230 | 15,495 | 9,330 | 17,869 | 52,512 | 53,822 | 36,544 | 40,923 |
| **G3** | 9,921 | 15,336 | 11,733 | 20,005 | 49,302 | 67,425 | 33,413 | 36,141 |
| **G4** | 13,651 | 17,232 | 15,034 | 19,473 | 50,171 | 59,001 | 34,497 | 33,466 |
| **G5** | 19,906 | 16,382 | 21,427 | 20,187 | 58,421 | 60,072 | 34,803 | 30,329 |
| **H6** | 4,776 | 15,615 | 8,025 | 16,358 | 221,051 | 52,504 | 110,515 | 59,066 |
| **H7** | 6,565 | 16,774 | 9,910 | 18,083 | 39,685 | 42,847 | 32,659 | 36,745 |
| **H8** | 22,136 | 27,889 | 16,768 | 26,820 | 44,942 | 51,384 | 24,692 | 27,542 |

A Tabela 5.1 mostra os resultados dos experimentos realizados. A comparação deve ser feita entre os índices no mesmo conjunto de dados. Não é possível comparar, por exemplo, o índice H6 no conjunto de dados 'Municípios do Brasil' com o índice H6 do conjunto 'Lotes'. Isso se deve ao fato de os retângulos de teste gerados para o primeiro e o segundo conjuntos, serem totalmente diferentes.

Outra comparação possível é entre SGBDs. Por exemplo, pode-se comparar os tempos dos índices H6 gerados sobre o conjunto de dados 'Municípios do Brasil' no PostGIS e no MySQL, pois os retângulos de teste usados nesses dois SGBDs são exatamente os mesmos.

Para o conjunto de dados dos 'Municípios do Brasil', pode-se observar que até mesmo o índice B-tree possui um tempo de resposta competitivo em relação a todos os demais métodos de indexação espacial. Todos os tempos estão dentro da mesma ordem de grandeza. Como o número de objetos é muito pequeno, a utilização de métodos de indexação não apresenta melhora expressiva no tempo de resposta das consultas. A R-tree do PostGIS foi a menos eficiente. Verificou-se que este comportamento não está relacionado com a ordem de inserção dos objetos. Por dois motivos:

- Primeiro porque o índice foi gerado após a inserção de todos os objetos no banco. Ainda foi executado o aplicativo Vacuum do PostGIS que tenta balancear o índice criado.

- Os dados foram inseridos em ordens diferentes e os tempos de resposta computados foram semelhantes.

No conjunto de dados dos 'Setores Censitários de São Paulo', com cerca de 12.000 objetos, os índices de Grade-Fixa (G2) e Hilbert (H6) se mostraram competitivos em relação à R-tree para o PostGIS. Para o MySQL, o resultado foi ainda mais expressivo, uma vez que os índices G2 e H6 foram mais de 10 vezes mais rápidos do que o método B-tree.

Para os testes com os dados de 'Lotes', com cerca de 120.000 objetos, os índices G3 e H7 se mostraram competitivos com a R-tree no PostgreGIS e bem superiores ao índice B-tree. No MySQL os tempos dos índices G2 e H7 foram melhores do que o B-tree. O índice H6 se mostrou ineficiente, devido, provavelmente, ao grande número de objetos por célula que esta resolução gerou.

No conjunto de dados do 'Prodes', com cerca de 1.100.000 objetos, os métodos G3 e H8 no PostGIS foram competitivos com a R-tree. Os três métodos foram bem superiores ao da B-tree neste banco. No MySQL, os índices G5 e H8 foram bem melhores que o B-tree.

As Tabelas 5.25.3 5.4 e 5.5 mostram o tamanho dos índices, em Kbytes e número de linhas, aplicados a cada conjunto de dados.

TABELA 5.2 – Tamanhos dos Índices sobre o conjunto 'Municípios do Brasil'.

| Municípios do Brasil | | | | |
|---|---|---|---|---|
| | PostGIS | | MySQL | |
| índice | KBytes | linhas | KBytes | linhas |
| B-tree | 304 | | 350 | |
| R-tree | 296 | | | |
| Grade nível 0 | 1896 | 28749 | 3104 | 28749 |
| Grade nível 1 | 968 | 14469 | 2032 | 14469 |
| Grade nível 2 | 632 | 9414 | 736 | 9414 |
| Grade nível 3 | 552 | 8130 | 592 | 8130 |
| Grade nível 4 | 504 | 7382 | 576 | 7382 |
| Grade nível 5 | 488 | 7089 | 544 | 7089 |
| Hilbert 6 | 920 | 15786 | 1968 | 15786 |
| Hilbert 7 | 1888 | 32952 | 3104 | 32952 |
| Hilbert 8 | 5040 | 88423 | 6176 | 88423 |

TABELA 5.3 – Tamanhos dos Índices sobre o conjunto 'Setores de São Paulo'.

| Setores Censitarios de São Paulo | | | | |
|---|---|---|---|---|
| | PostGIS | | MySQL | |
| índice | KBytes | linhas | KBytes | linhas |
| B-tree | 680 | | 1100 | |
| R-tree | 656 | | | |
| Grade nível 0 | 4224 | 64430 | 6176 | 64430 |
| Grade nível 1 | 2160 | 32923 | 3104 | 32923 |
| Grade nível 2 | 1424 | 21550 | 3104 | 21550 |
| Grade nível 3 | 1248 | 18715 | 3104 | 18715 |
| Grade nível 4 | 1136 | 17102 | 3104 | 17102 |
| Grade nível 5 | 1088 | 16457 | 3104 | 16457 |
| Hilbert 6 | 1648 | 28590 | 3104 | 28590 |
| Hilbert 7 | 2944 | 51516 | 5152 | 51516 |
| Hilbert 8 | 6936 | 121874 | 9248 | 121874 |

TABELA 5.4 – Tamanhos dos Índices sobre o conjunto ‘Lotes’.

| Lotes | | | | |
|---|---|---|---|---|
| | PostGIS | | MySQL | |
| índice | KBytes | linhas | KBytes | linhas |
| B-tree | 8240 | | 10130 | |
| R-tree | 8336 | | | |
| Grade nível 0 | 46080 | 702089 | 55424 | 702089 |
| Grade nível 1 | 24864 | 385574 | 31808 | 385574 |
| Grade nível 2 | 17088 | 264597 | 22560 | 264597 |
| Grade nível 3 | 15000 | 229714 | 19488 | 229714 |
| Grade nível 4 | 13936 | 213369 | 19488 | 213369 |
| Grade nível 5 | 13320 | 203894 | 18464 | 203894 |
| Hilbert 6 | 9760 | 148692 | 15392 | 148692 |
| Hilbert 7 | 13960 | 245539 | 17440 | 245539 |
| Hilbert 8 | 19288 | 339576 | 24608 | 339576 |

TABELA 5.5 – Tamanhos dos Índices sobre o conjunto ‘Prodes’.

| Prodes | | | | |
|---|---|---|---|---|
| | PostGIS | | MySQL | |
| índice | KBytes | linhas | KBytes | linhas |
| B-tree | 51200 | | 53531 | |
| R-tree | 54272 | | | |
| Grade nível 0 | 627712 | 9627710 | 811936 | 9627710 |
| Grade nível 1 | 242688 | 3723480 | 316016 | 3723480 |
| Grade nível 2 | 130048 | 1983420 | 165200 | 1983420 |
| Grade nível 3 | 103424 | 1582480 | 131344 | 1582480 |
| Grade nível 4 | 92160 | 1416690 | 119056 | 1416690 |
| Grade nível 5 | 86016 | 1328130 | 111824 | 1328130 |
| Hilbert 6 | 62464 | 1086280 | 72848 | 1086280 |
| Hilbert 7 | 63488 | 1124240 | 82096 | 1124240 |
| Hilbert 8 | 68608 | 1209500 | 85184 | 1209500 |

Pode-se perceber que o espaço em disco necessário para armazenar os índices criados (Grades Fixas e Hilbert SFCs) é maior do que o espaço necessário para a criação dos índices do SGBD (B-tree para PostGIS e MySQL e R-tree para o PostGIS).

A partir dos resultados apresentados na Tabela 5.1 e nos tamanhos dos índices apresentados nas Tabelas 5.25.3 5.4 e 5.5 fica claro que a resolução – no caso das Grades Fixas – e ordem da curva – no caso das Hilbert SFCs –, definidas na criação dos índices, são fatores determinantes do desempenho do índice. Em outras palavras, uma grade muita fina ou uma Hilbert SFC com grau de ordem muito alto, pode tornar o uso do índice impraticável por dois motivos:

1. Uma resolução muito fina ou uma ordem muito alta tendem a gerar muitas entradas na tabela de índice para o mesmo objeto aumentando assim o espaço em disco necessário para armazenar o índice.

2. Quando a resolução não é adequada, a consulta SQL gerada pelo *framework* se torna um problema, pois esta consulta pode conter uma cláusula WHERE muito extensa, levando muito tempo para ser enviada e processada pelo SGBD.

Por outro lado, grades muito grosseiras e Hilbert SFCs com ordens muito baixas podem gerar um número muito alto de objetos por célula. As consultas enviadas ao SGBD, então, podem retornar muitos objetos. Isso pode diminuir o desempenho dos índices. Por exemplo, em operações de seleção de objetos. Nesse tipo de operação é necessário realizar uma etapa de filtragem. Neste caso, quanto menor o número de objetos resultantes da filtragem, melhor o desempenho.

# CAPÍTULO 6

# CONCLUSÕES E TRABALHOS FUTUROS

A hipótese do trabalho é verdadeira: as estruturas propostas se mostraram mais eficientes do que a utilização do índice B-tree nos dois SGBDs usados. Foi verificado que, nos conjuntos de objetos onde há muitas sobreposições entre os MBRs dos objetos, os índices Hilbert SFC e Grades Fixas apresentam tempos de resposta competitivos, inclusive sobre métodos já consolidados como a R-tree do PostGIS. Isso pode ser verificado na Tabela 5.1 os índices gerados sobre os conjuntos de dados de 'Municípios do Brasil', 'Setores de São Paulo' e do 'Prodes', melhoraram o desempenho médio dos tempos de resposta das consultas geradas sobre estes conjuntos. Já no conjunto 'Lotes', onde há poucas sobreposições a R-tree foi mais eficiente, porém os métodos ainda foram competitivos. Os índices propostos são pouco sensíveis a inserções e remoções de objetos. A menos que o retângulo envolvente do conjunto de dados seja alterado, a inserção de um objeto significa apenas indexar este objeto. A remoção da mesma maneira. No caso de uma inserção que altere o MBR do conjunto de dados, o índice deve ser recriado.

A má configuração dos índices compromete o desempenho dos mesmos. Exemplos podem ser vistos na Tabela 5.1. Uma resolução muito fina, como a H8 e a G0 geradas sobre o conjunto 'Municípios do Brasil' ou a G0 gerada sobre o conjunto 'Prodes', tornam o índice ineficiente. Neste caso, um outro problema que se percebe, é o aumento do espaço necessário para armazenar o índice. Da mesma maneira, resoluções muito grosseiras como a G5 e H6 geradas sobre o conjunto 'Prodes' tornam os índices ineficientes. Nos experimentos realizados, as SFCs com resoluções menores que 6 iterações são ineficientes, assim como as maiores que 8 iterações. Mais do que 8 iterações, não houve resposta porque a string de consulta ficou muito grande.

Como resultado desse trabalho, construiu-se um arcabouço para a criação de outros tipos de indexadores espaciais baseados em B-Trees.

Como trabalho futuro, pode-se pesquisar uma forma de calcular a melhor resolução para criar o índice, baseado nos dados que se pretende indexar. Uma pergunta que não teve resposta neste trabalho é “Como escolher a resolução ideal?”. Hoje a escolha seria feita baseada em testes e observações. Também é preciso analisar a indexação de maiores massas de dados de pontos e linhas, e em outros SGBDs. Ainda pretende-se criar um ambiente onde os índices possam ser criados, configurados e utilizados sobre o ambiente da TerraLib.

## REFERÊNCIAS BIBLIOGRÁFICAS


ASANO, T.; RANJAN, D.; ROOS, T.; WELZL, E.; WIDMAYER, P. Space-filling curves and their use in the design of geometric data structures. **Theoretical Computer Science**, v. 181, n. 1, p. 3--15, 1997.

BITTON, D.; DEWITT, D. J.; TURBYFILL, C. Benchmarking database systems: a systematic approach. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 9., 1983, San Francisco (CA, USA). **Electronic Proceedings**... San Francisco: [s.n]. Disponível em: <http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=673616#>. Acesso em: 20 jan. 2008.

CASANOVA, M.; CÂMARA, G.; DAVIS, C.; VINHAS, L.; QUEIROZ, G. R. D. **Bancos de dados geográficos**. Curitiba, 2005.

CIFERRI, R. R.; SALGADO, A. C. Performance evaluation of multidimensional access methods. In: ACM INTERNATIONAL SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 8., 2000, Washington (DC, USA). **Electronic Proceedings..**. Disponível em: <http://portal.acm.org/citation.cfm?id=355302&dl=ACM&coll=GUIDE>. Acesso em: 15 fev. 2008.

COMER, D. Ubiquitous B-Tree. **ACM Comput. Surv.**, v. 11, n. 2, p. 121-137, 1979.

COMMUNITY, P. PostGIS manual. 2001.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to algorithms** McGraw-Hill Book Company, 2000.

DAVIS JR., C.; QUEIROZ, G. R. Métodos de acesso para dados espaciais. In: Casanova, M. A.; Câmara, G.; Davis Jr., C. A.; Vinhas, L.; Queiroz, G. R. (Ed.). **Bancos de dados geográficos**. Curitiba, PR: MundoGEO, 2005. cap. 6, p. 213-231.

GAEDE, V.; GÜNTHER, O. Multidimensional access methods. **ACM Comput. Surv.**, v. 30, n. 2, p. 170-231, 1998.

GUTTMAN, A. R-trees: a dynamic index structure for spatial searching. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1984, Boston (Massachusetts, USA). **Electronic Proceedings …** Disponível em: < http://portal.acm.org/citation.cfm?id=602266 >. Acesso em 12 jan. 2008.

HADJIELEFTHERIOU, M.; HOEL, E.; TSOTRAS, V. J. SaIL: a spatial index library for efficient application integration. **Geoinformatica**, v. 9, n. 4, p. 367-389, 2005.

HELLERSTEIN, J. M.; NAUGHTON, J. F.; PFEFFER, A. Generalized search trees for database systems. **Morgan Kaufmann Publishers Inc.** In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 21., San Francisco (CA, USA). **Electronic Proceedings…**Disponível em: < http://portal.acm.org/citation.cfm?id=673145 >. Acesso em: 15 jan. 2008.

LAWDER, J. K.; KING, P. J. H. Using space-filling curves for multi-dimensional indexing. **Springer-Verlag** In: BRITISH NATIONAL CONFERENCE ON DATABASES: ADVANCES IN DATABASES, 17., 2000, London (UK). **Electronic Proceedings** … Disponível em: < http://portal.acm.org/citation.cfm?id=646102.681186 >. Acesso em: 12 jan. 2008.

POSTGRESQL. **PostgreSQL main site**. 2008. Disponível em: < http://www.postgresql.org>. Acesso em: 18 fev. 2008.

QUAK, W.; OOSTEROM, P. V.; TIJSSEN, T. Testing current DBMS products with realistic spatial data. In: AGILE CONFERENCE ON GEOGRAPHIC INFORMATION SCIENCE, 5., 2002, Palma (Mallorca, Spain). **Proceedings…** Palma: [s.n], 2002.

RAVADA, S.; SHARMA, J. Oracle8i spatial: experiences with extensible databases. **Springer-Verlag** In: INTERNATIONAL SYMPOSIUM ON ADVANCES IN SPATIAL DATABASES, 6., 1999, London (UK). **Electronic Proceedings …** Disponível em: <http://portal.acm.org/citation.cfm?id=647226.719081#>. Acesso em: 19 jan. 2008.

SAMET, H. The Quadtree and related hierarchical data structures. **ACM Comput. Surv.**, v. 16, n. 2, p. 187-260, 1984.

STONEBRAKER, M.; FREW, J.; GARDELS, K.; MEREDITH, J. The SEQUOIA 2000 storage benchmark. **SIGMOD Rec.**, v. 22, n. 2, p. 2-11, 1993.

THEODORIDIS, Y. Ten benchmark database queries for location-based services. **Computer Journal**, v. 46, p. 713-725, 2003.

TIMOS, K. S.; NICK, R.; CHRISTOS, F. Multidimensional access methods: trees have grown everywhere. **Morgan Kaufmann Publishers Inc.** In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 23., 1997, San Francisco (CA, USA). **Electronic Proceedings…** Disponível em: <http://portal.acm.org/citation.cfm?id=645923.673658&coll=GUIDE&dl=GUIDE>. Acesso em: 13 dez. 2007.

WERSTEIN, P. A performance benchmark for spatialtemporal databases. In: COLLOQUIUM OF SPATIAL INFORMATION RESEARCH CENTRE, 10., 1998, New Zealand. **Proceedings…** New Zealand: University of Otago, 1998.

WIKIPEDIA. **Hilbert curve**. 2008. Disponível em: <http://en.wikipedia.org/wiki/Hilbert_curve>. Acesso em: 18/02/2008.

YARGER, R. J.; REESE, G.; KING, T. **MySQL and mSQL.** Cambridge: O'Reilly, 1999. 487 p. ISBSN 1565924347. 553408.

**APÊNDICE A**

# Performance analysis of Hilbert space-filling curves for spatial database queries

Frederico Bedê[1], Gilberto Ribeiro de Queiroz[1], Lúbia Vinhas[1], Gilberto Câmara[1*]

[1] National Institute for Space Research, Avenida dos Astronautas 1758, 12227-001 São José dos Campos, Brazil

**Abstract.** This paper compares the performance of Hilbert space-filling curves with R-trees as index structures for spatial databases. R-trees are the most common index in spatially enabled DBMS. We show that Hilbert curves are a competitive alternative to R-trees for spatial queries.

**Keywords:** spatial databases, data structures, spatial indexes, Hilbert space-filling curves.

## Introduction

Spatial databases need indexes for efficient performance of spatial queries. There has been much research on multidimensional indexes such as R-trees [6] and its variants [15]. For comprehensive reviews of spatial indexes, see [4] and [13]. Spatial indexes improve significantly the performance of spatial queries. This fact motivated database developers to include in-core support for R-trees on their products [14]. This paper explores an alternative way of including spatial indexes in DBMS, using Hilbert space-filling curves (SFC). One benefit of Hilbert SFC is that they can be implemented in standard relational tables, and thus do not need internal extensions to the DBMS. We consider that it is sometimes convenient to use an out-of-core spatial index, without having to adapt the DBMS kernel.

The motivation for our research is the interest on developing indexes for spatio-temporal data, since applications such as location-based services and dynamical modelling need efficient indexing [9] [7]. One approach for building spatio-temporal indexes is to extend R-trees [16], a task which demands changes in the DBMS kernel and thus be outside the reach of many researchers or application developers. Additionally, the DBMS may not have the facilities for query optimization for spatio-temporal data. In this case, having an efficient index outside the DBMS core helps development of query optimizers for spatio-temporal queries.

However, performance considerations are important in spatial databases. This paper compares the performance of Hilbert space-filling curves and R-trees for a representative set of workloads for spatial database applications. The use of Hilbert space-filling curves (SFC) for indexing and querying spatial data was proposed by [11]. The authors argue that "*comparison with R-trees showed a 75% saving to populate a database and 90% for executing range queries*" [11]. However, they do not provide details of their workloads on their paper. The use of space-filling curves for spatial indexing is also investigated in [1], where the authors indicate there is a significant performance gain with space-filling curves built on top of a B-tree. However, But they do not compare SFCs with R-trees. More recently, [12] analyse the conditions where SFCs could be used for multidimensional indexing, but do not provide performance metrics. Thus, we consider the issue of comparative performance of Hilbert space-filling curves and R-trees has not been sufficiently covered in the literature.

**Spatial Indexing and Queries Using Hilbert SFC**

Spatial indexes are auxiliary data structures built considering the geometries of the geographical data. They are essential for efficient processing of spatial queries. We consider spatial databases as composed by *layers,* where each layer has a set of objects and each object has a unique identification and a geometry. A spatial query applied to the database returns a subset of the objects of a layer that satisfies the query´s predicates.

A space-filling curve is a mapping from a bidimensional space into a one-dimensional space. Given a set of spatial data, it is divided in cells and the curve represents a thread that passes through every cell once. The sequence of points in the curve dictates the order in which each cell is visited [12]. David Hilbert proposed the Hilbert SFC in 1891 [8], as a mapping which considers that a point on a line is the limit of a sequence of intervals whose length tends to zero. Thus, a point will also be the limit of a sequence of squared cells whose area tends to zero[11]. Figure 1 shows three Hilbert curves of first, second and third orders, which cover the extent of the municipalities in the Brazilian Amazonian region.

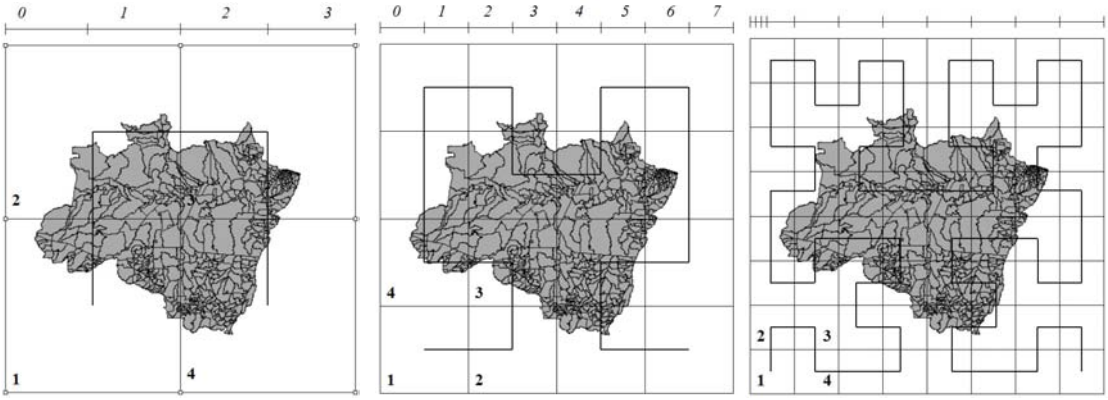


**Fig. 1**. Examples of Hilbert space-filling curves (first, second and third orders).

We measure how spatial index performs by assessing the database response to spatial queries. The most common types of spatial queries are those that retrieve the objects that intercept a given rectangle (*range query*) and those that return the objects that contain a given point (*point query*) (Figure 2).

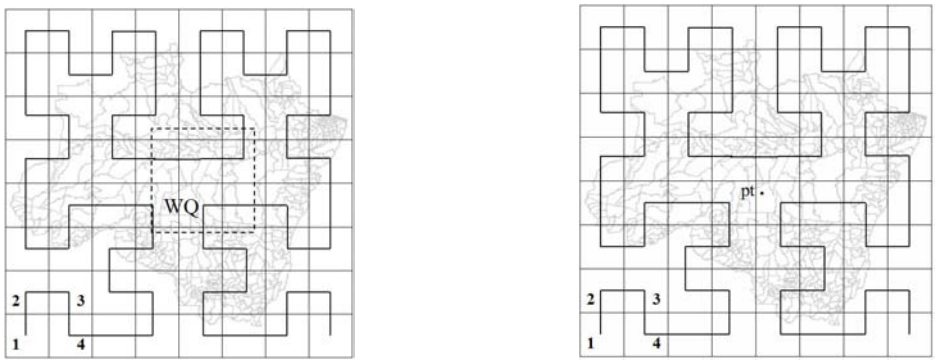



**Fig. 2.** Spatial queries: range query (left) and point query (right).

A spatial query has two parts: a *filter* part and a *refine* part. In the filter part, the spatial indexes are used to retrieve efficiently the list of candidates to answer the query, reducing the search space. Then, in the *refine* part, we test whether the actual geometry of the candidates satisfies the query predicates, and thus we find the subset of spatial objects that match the spatial query. The Hilbert SFCs are used in the filter part of the query. To filter a spatial query using Hilbert SFCs, we first recover the list of Hilbert

codes that intercepts the query window `WQ` or the point `pt`. The result will be a list of Hilbert codes, for example the list {1, 2, 3, 11, 16, 17, 18}. We break the list in intervals, for example {[1:3], [11], [16:18]}. Then an SQL join query maps the intervals to the object identifiers. The result gives us the list of possible candidates for the *refine* part. In the *refine* part, we compare the details of the candidates to see whether they match the query. Using Hilbert SFCs, we transform a bidimensional query (spatial) into a one-dimensional query that can be efficiently answered using the B-Tree indexes provided by relational DBMS.

**Experimental Results**

In our experiments, we compared an in-kernel spatial index (R-trees in PostGIS [14]) with an out-of-kernel method (the Hilbert SFC). We used three datasets: (a) The census tracts of São Paulo, Brazil, with roughly 12,000 polygons, which are fairly homogeneous in area, but where each polygon has a different number of vertices (Figure 3). (b) The land parcels of São José dos Campos in Brazil, with around 120,000 polygons, that are regular in area and number of vertices (Figure 4); (c) The deforestation areas detected in the Amazon rain forest in Brazil for 2007, with roughly 1,200,000 polygons, which are diverse in area and whose number of vertices can vary from 10 to 30,000 (Figure 5).

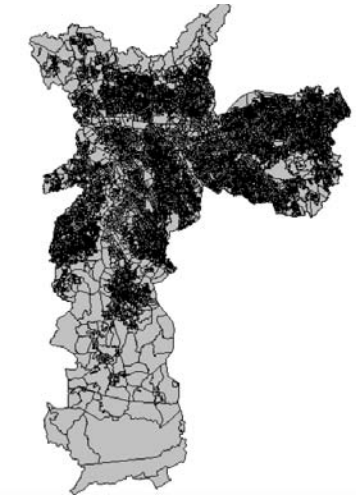


**Fig.3**. Census tracts of São Paulo.

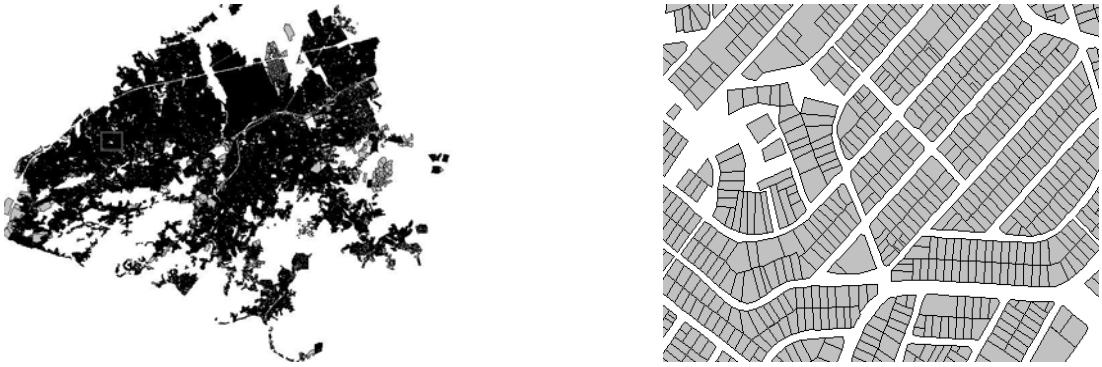


**Fig. 4.** Parcels of São José dos Campos (left) and detail of some parcels (right).

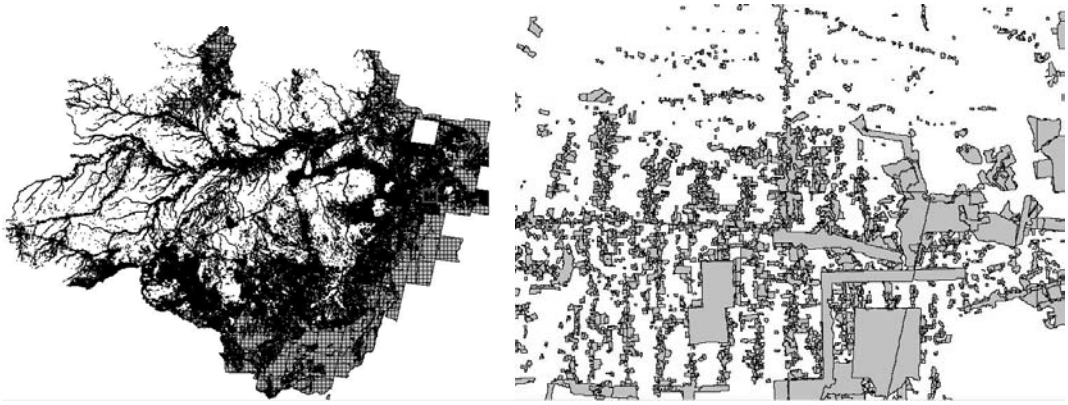


**Fig. 5.** Deforestation polygons for Amazonia (2007) with detail of some polygons (right).

Our benchmark was developed using the open source TerraLib GIS library [2]. TerraLib has an abstract interface that represents a spatial database and provides *drivers*

for different DBMS. Drivers for DBMSs with spatial capabilities use the indexes provided by the extension. For example, the PostGIS add-on to PostgreSQL [14] implements in-core R-Trees for spatial data handling. Drivers for DBMSs without spatial extensions use TerraLib to manage the spatial data. In these cases TerraLib stores explicitly the minimum bounding rectangle (MBR) of the objects in four fields (*lower_x*, *lower_y*, *upper_x* and *upper_y*) and creates a B-Tree index using the combination of these fields. We have added indexing by Hilbert SFC to TerraLib as an alternative for handling spatial data in DBMS.

We compared the in-core R-tree index in PostGIS with out-of-core Hilbert SFCs of orders 6, 7 and 8, referred as H6, H7 and H8. SFCs with orders lower than 6 produced subsquares that intercepted too many objects. This produced too many candidates for the refining step and decreasing the overall performance. SFCs with orders higher than 8 produce small subsquares. The same polygon intercepts many subsquares. This creates many intervals, and the joint queries are too large to be processed by the DBMS. For each dataset we simulated 1000 random window queries with variable sizes and we measured the average time to get the results. Table 1 shows the results of the experiments. The experiment compares query processing time using TerraLib B-tree index, the PostGIS R-tree index, and Hilbert SFCs. The experiments have been performed on a machine with a Pentium 4HT processor of 3GHz. This computer has 1GB of main memory and 100 GB of disk space, running on a Linux Mandriva 2008 operational system. The DBMS used is PostgresSQL 8.2 with spatial extension PostGIS 1.3.

**Table 1 – Average retrieval times (in ms) for 1000 random range queries**

| Spatial index | Census tracts | Urban Parcels | Deforestation |
|---|---|---|---|
| B-tree | 17.39 | 288.64 | 650.46 |
| R-tree | 10.83 | 20.54 | 91.25 |
| Hilbert H6 | 8.02 | 221.05 | 110.51 |
| Hilbert H7 | 9.91 | 39.68 | 32.66 |
| Hilbert H8 | 16.76 | 44.94 | 24.69 |

Analysis of the results shows that Hilbert SFCs have a credible performance when compared with in-core R-trees, provided the SFC have a suitable order. In both the census and deforestation datasets, Hilbert SFC of order H7 achieved a better performance than R-trees. For heterogeneous polygons, such as the deforestation ones, Hilbert SFC are better than R-trees for at least a factor of 3. R-trees have a better performance in the urban parcels data, which are regular in area and number of vertices. When polygons increase in size and complexity, the Hilbert SFC provides a sound way to extend DBMs to handle spatial data. Thus, the extra time and effort needed to develop in-core indexes in DBMs to support spatial data may not be strictly necessary. The good performance of Hilbert SFCs makes them good candidates to support spatio-temporal indexes for applications such as location-based services.

## Conclusions

Spatial indexes are important to ensure performance in queries in spatial databases. R-tree and its variants have become the index of choice for most spatial databases implementations. There are many benchmarks for different types of spatial indexes using R-trees in the literature [5] [10] [3] [17]. Each of these benchmarks evaluates the

performance of different types of spatial queries using R-trees and its variants. The main difference of our work from the previous studies in the literature is that we compare an in-kernel spatial index (R-trees in PostgreSQL) with an out-of-kernel method (the Hilbert SFC), using data sets that are diverse in nature. Our main contribution is to assess the costs and the benefits of in-core R-trees, compared with out-of-core methods such as Hilbert SFCs. This issue has not sufficiently being addressed in the literature.

Our performance analysis shows that Hilbert SFCs provide a competitive alternative to R-trees, with the added benefit of extensibility. Thus, Hilbert SFCs are a possible index for more complex applications, such as those involving spatio-temporal objects and queries.

## References


[1] C. Bohm, G. Klump, H.-P. Kriegel, XZ-Ordering: A Space-Filling Curve for Objects with Spatial Extension, in: Proceedings of the 6th International Symposium on Advances in Spatial Databases. Lecture Notes In Computer Science, Vol. 1651. Hong Kong, China, Springer-Verlag, 1999, pp. 75-90.

[2] G. Câmara, L. Vinhas, G. Queiroz, K. Ferreira et al., TerraLib: An open-source GIS library for large-scale environmental and sócio-economic applications, in: B. Hall, (ed.), Open Source Approaches to Spatial Data Handling. Springer, Berlin, 2008 (in press).

[3] R. R. Ciferri, A. C. Salgado, *Performance evaluation of multidimensional access methods*, in *Proceedings of the 8th ACM international symposium on Advances in geographic information systems*. 2000, ACM: Washington, D.C., United States.

[4] V. Gaede, O. Günther, Multidimensional Access Methods. ACM Computing Surveys 30 (1998) 170-231.

[5] D. Greene, An Implementation and Performance Analysis of Spatial Data Access Methods, in: IEEE Fifth International Conference on Data Engineering. Los Angeles, CA1989.

[6] A. Guttman, R-Trees: A Dynamic Index Structure for Spatial Searching, in: Annual Meeting ACM SIGMOD. Boston, MA1984, pp. 47-57.

[7] M. Hadjieleftheriou, G. Kollios, P. Bakalov, V. J. Tsotras, Complex spatio-temporal pattern queries, in: 31st International Conference on Very Large Data Bases Trondheim, Norway, VLDB Endowment, 2005.

[8] D. Hilbert, Ueber die stetige Abbildung einer Line auf ein Flächenstück. Mathematische Annalen 38 (3) (1891) 459-460.

[9] G. Kollios, V. J. Tsotras, D. Gunopulos, A. Delis et al., Indexing animated objects using spatiotemporal access methods. IEEE Transactions on Knowledge and Data Engineering 13 (5) (2001) 758-777.

[10] H.-P. Kriegel, M. Schiwietz, R. Schneider, B. Seeger, Performance Comparison of Point and Spatial Access Methods, in: A. Buchmann, et al., (ed.), Symposium on the Design and Implementation of Large Spatial Databases. Springer-Verlag, New York, NY, 1989, pp. 89-114.

[11] J. K. Lawder, P. J. H. King, *Using Space-Filling Curves for Multi-dimensional Indexing*, in *Proceedings of the 17th British National Conferenc on Databases: Advances in Databases*. 2000, Springer-Verlag.

[12] M. F. Mokbel, W. G. Aref, I. Kamel, Analysis of multi-dimensional space-filling curves. GeoInformatica 7 (3) (2003) 179-209.

[13] H. Samet, Foundations of multidimensional and metric data structures, Elsevier/Morgan Kaufmann, Amsterdam ; Boston, 2006.

[14] S. Santilli, M. Leslie, C. Hodgson, P. Ramsey et al., *PostGIS Manual - Version 1.3.2*. 2007, Refractions Research: Victoria, CA.

[15] T. Sellis, N. Roussopoulos, C. Faloutsos, The R+-Tree: A Dynamic Index for Multi-Dimensional Objects, in: P. Stocker and W. Kent, (ed.), 13th International Conference on Very Large Data Bases. Brighton, England, 1987, pp. 507-518.

[16] Y. Tao, D. Papadias, BMV3R-Tree: A spatio-temporal access method for timestamp and interval queries, in: Proceedings of Very Large Data Bases (VLDB) Conference, 2001, pp. 431-440.

[17] Y. Theodoridis, Ten Benchmark Database Queries for Location-based Services. Computer Journal 46 (2003) 713-725.

## PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

### Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

### Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

### Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

### Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

### Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.

### Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

### Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

### Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

### Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.