Armazenamento e Recuperação de Dados Matriciais em Bancos de Dados Objeto-Relacionais

Lúbia Vinhas Gilberto Câmara
DPI – Divisão de Processamento de Imagens
INPE – Instituto Nacional de Pesquisas Espaciais
{lubia, gilberto}@dpi.inpe

Resumo

Esse trabalho examina algumas quetões relatives a construção de bases de dados geográficos, mais especificamente daqueles representados em uma estrutura matricial. Uma arquitetura geral é apresentada. Essa arquitetura baseia-se na combinação de dois mecanismos: particionamento e multi-resolução. Apresenta-se alguns algoritmos desenvolvidos para implementar tais mecanismos. Apresenta-se ainda exemplos de implementação desse modelo em dois SGBDs objeto-relacionais: MySQL e PostgreSQL. Para esse exemplo são discutidas algumas questões de eficiência na recuperação desses dados.

***Palavras-chave**: bancos de dados geográficos; dados matriciais espaciais.*

1. Introdução

Os sistemas de informação geográfica (SIG) devem incluir em seu nível mais interno, um sistema de gerência de bancos de dados geográficos. Esse sistema oferece armazenamento e recuperação dos dados espaciais e seus atributos. As gerações atuais de SIG utilizam os sistemas gerenciadores de bancos de dados comerciais (SGBD) para executar tal tarefa (Câmara, 2005). Alguns SGBD's como Oracle (Ravada e Sharma, 1999) ou PostgreSQL (Santilli et al., 2005) desenvolveram as chamadas extensões espaciais para suportar o gerenciamento da componente espacial dos dados geográficos, especialmente quando representados por estruturas vetoriais (polígonos, linhas ou pontos). Outros gerenciadores, como MySQL (Widenius et al., 2002) não fornecem extensões espaciais, mas permitem o armazenamento de dados complexos em campos binários longos (BLOB's). Nesse caso o SIG necessita de uma camada intermediária para executar a codificação e a decodificação do dado espacial nos campos binários.

O suporte, por parte dos SGDB's, aos dados matriciais espaciais pode ainda ser considerado um tema emergente (Shekkar e Chawla, 2003). A abordagem mais comum tem sido a criação de servidores especializados, como é o caso do PARADISE (Patel et al., 1997) e do RASDAMAN (Baumann et al., 1998) A principal vantagem dessa abordagem é a capacidade de ganho em desempenho, especialmente para o caso de grandes bases de dados. A desvantagem é a necessidade de um servidor especializado o que sobrecarregaria as necessidades, em termos de gerencia de dados, da maioria das aplicações de sistemas de informação geográficos. Uma segunda abordagem é o tratamento de dados matriciais espaciais em SGBD's objeto-relacionais convencionais. Esse trabalho descreve uma proposta de solução nesse contexto. Tratando especialmente das questões de particionamento, indexação e consulta espacial. Mostra-se um exemplo de implementação em dois SGDB's *freeware*: MySQL sem extensão espacial e PostgreSQL com extensão espacial

2. Bases de Dados Matriciais

A organização dos dados diversos dados geográficos em uma base de trabalho, baseia-se na metáfora do plano de informação (Burrough e McDonnell, 1988) Esse modelo de organização em planos de informação também se a aplica na construção de bases de dados geográficos. Um plano de informação que

contém dados matriciais pode conter um grande volume de dados, como por exemplo, planos derivados do mosaico de várias cenas de imagens de satélite. Soluções eficientes para gerenciar esses dados baseiam-se na combinação de duas técnicas: particionamento (*tiling*) e multi-resolução.

Pode-se compreender essa estratégia através de um exemplo: considere um mosaico de imagens do satélite Ikonos que recobre a área da cidade de São José dos Campos, SP. Essa área é de aproximadamente 1100 Km2. Como a resolução espacial das imagens Ikonos é de 1 Metro, o mosaico teria aproximadamente 1010 *pixels*. Para mostrar essa imagem em um canvas de 1500×750 *pixels*, uma aplicação teria que recuperar toda a imagem do banco de dados e reamostrá-la para uma resolução de 1000 Metros. Da mesma forma, quando é feito uma ampliação para uma pequena área, toda a imagem teria que ser recuperado do banco, mas apenas uma pequena parte seria efetivamente utilizada.

A técnica de multi-resolução consiste em pre-computar diferentes versões reamostradas da imagem e armazená-las no banco de dados. Cada versão recobre a mesma área geográfica, porém com menos *pixels*. As aplicações podem decidir qual versão deve ser recuperada e mostrada com base no tamanho de sua área de desenho. Para compensar o aumento no volume de dados realmente armazenados, são usados algoritmos de compressão antes da efetiva inserção no banco de dados.

A estratégia de particionamento consiste na divisão da imagem total em blocos de mesmo tamanho, dado em número de linhas e colunas, que são armazenados separadamente. Dessa forma, quando a aplicação necessita mostrar somente certa área da imagem, apenas os blocos que interceptam essa área devem ser recuperados, processados e mostrados. Isso demanda que cada bloco possua uma identificação única e possam ser recuperados com base na sua extensão espacial, também chamado de retângulo envolvente.

A estratégia de particionamento consiste na divisão do dado matricial em blocos de mesmo tamanho, dado em número de linhas e colunas, que são armazenados separadamente. Dessa forma, quando a aplicação necessita mostrar somente certa área da do dado matricial, apenas os blocos que interceptam essa área devem ser recuperados, processados e mostrados. Isso demanda que cada bloco possua uma identificação única e possam ser recuperados com base na sua extensão espacial, também chamado de retângulo envolvente.

A multi-resolução consiste em pré-computar diferentes versões do mesmo dado matricial. Cada versão degrada a resolução do dado, de forma que recobrem a mesma área, porém com uma quantidade de elementos menor. As aplicações podem decidir qual versão deve ser recuperada e mostrada com base, por exemplo, no tamanho de uma área de desenho.

Para atender as necessidades decorrentes da combinação de multi-resolução e particionamento, uma base dados matriciais consiste de relações como:

```
raster(tileId:string, band:int,
resLevel: int, box:MBR,tile:BLOB)
```

O termo MBR representa um tipo que permite o armazenamento de um retângulo envolvente. O termo BLOB representa um campo que permite o armazenamento de uma um campo binário longo.

2.1 Particionamento

O algoritmo 1 define uma grade de recorte que define os limites dos blocos gerados. A grade de recorte é criada no domínio geográfico a fim de que suporte a inclusão de outros dados em uma representação já criada.

Algoritmo 1: Particionamento
Entrada: A altura *H* e largura *W* dos blocos que serão gerados em número de elementos

1. Ler os valores de resolução horizontal (*resX*) e vertical (*resY*) e o retângulo envolvente do dado matricial (*MBR*)
2. Calcular a altura (*HG*) e largura (*WG*) dos blocos em coordenadas geográficas
3. Calcular um novo mínimo retângulo envolvente (*nMBR*) para o dado matricial no banco de dados que satisfaça as seguintes condições:
    a. contenha o retângulo envolvente original
    b. sua largura seja múltipla de *WG* e sua altura seja múltipla de *HG*
4. Definir uma grade de recorte sobre *nMBR* com *n* eixos verticais separador por *WG* e *m* eixos horizontais separados por *HG*
5. Para cada bloco definido pela grade de recorte, calcule sua posição vertical e horizontal dada por:
    a. *HP* = posição do eixo esquerdo do bloco *WG*
    b. *VP* = posição do eixo superior do bloco *HG*
6. Criar um identificador único, dado pela combinação de *HP* e *VP*

7. Associar o novo retângulo envolvente (*nMBR*) ao dado matricial armazenado e recalcular o número de linhas e colunas

O Algoritmo 2 define como, dada a posição de um elemento (c,j) (em linha e coluna) encontrar o identificador do bloco que o contém.

Algoritmo 2: IdentBloco

Entrada: A posição de um elemento do dado matricial (*c*,*j*)

1. Calcular a coordenada geográfica do elemento (*x*,*y*)
2. Calcular a posição do bloco que contém o elemento na grade de recorte:
   a. *HP = INT(x/WG)*
   b. *VP = INT(y/HG)*
3. O identificador único do bloco que contém o elemento é dado pela combinação de *HP* e *VP*

O algoritmo de particionamento é aplicado a cada banda de cada versão do dado matricial. Por isso, a banda e a versão (ou seja, o nível de resolução) são incluídas na identificação única dos blocos. A identificação dos blocos são literais, como por exemplo: 'X10Y100B0R1', 'X10Y100B1R1', 'X11Y100B0R1', 'X11Y100B1R1', 'X11Y100B1R2', 'X11Y100B0R2', 'X11Y100B1R2'. Um algoritmo de particionamento similar foi usado na arquitetura do projeto TerraServer da Microsoft (Barclay et al., 2000). Esse projeto é um servidor de imagens e outros dados matriciais onde foi adotado o sistema de projeção UTM. A zona UTM do dado é incluída na identificação dos blocos. O Algoritmo 1 não é restrito a uma projeção em particular.

2.2 Consulta

Os algoritmos de processamento digital de imagens (Gonzalez e Woods, 2002), necessitam percorrer o dado matricial todo, normalmente em uma ordem pré-definida e na resolução original. A melhor maneira de conseguir eficiência é criar um sistema de cache, já que normalmente não é possível manter a imagem inteira em memória. O sistema de cache é capaz de manter alguns blocos em memória. Quando um algoritmo necessita acessar certo elemento, o cache verifica se o bloco já está em memória, se estiver retorna seu valor; se não estiver um algoritmo de atualização é executado e o bloco necessário é trazido para a memória. No caso da visualização é sempre necessário trazer o conjunto de blocos que interceptam um dado retângulo envolvente em certo nível de resolução.

Em ambas as aplicações, pode-se fazer uso da adjacência espacial dos blocos. Blocos adjacentes no espaço devem ser recuperados conjuntamente. Existem na literatura diversas técnicas de agrupamento espacial baseadas em curvas de preenchimento espacial (Böhm et al., 1999; Lawder e King, 2001). O sistema de identificação de blocos proposto nesse trabalho baseia-se na ordem derivada da definição da grade de recorte universal. Essa grade resulta em uma ordenação na direção horizontal e na direção vertical, mas não nas duas (como mostra a Figura 3).

O modelo relacional mostrado na seção 2 permite que a consulta dos blocos seja feita de duas formas: 1) usando o campo `tileId` que é uma chave primária e 2) usando uma combinação dos campos: `band`, `resLevel`, e `box`. A escolha entre as duas alternativas deve ser baseada no tipo da aplicação. Na segunda alternativa o uso correto dos recursos disponíveis no SGDB onde está sendo implementado o banco influencia no desempenho.

3. Estudo de Caso

Usando a biblioteca TerraLib (Vinhas e Ferreira, 2005), a proposta descrita acima foi implementada em dois SGBDs: MySQL e PostgreSQL com a extensão espacial PostGIS. Foi criado um plano de informação com um mosaico de imagens Ikonos da área urbana de São José dos Campos. O mosaico foi construído em blocos de 512×512 elementos, com uma pirâmide de multi-resolução de 6 níveis chegando a uma resolução de 64 Metros no nível mais alto. No total o plano contém 3048 blocos.

```
raster(tileId:string, band:int,
resLevel: int, lower_x: double, lower_y:
double, upper_x: double, upper_y:
double, tile:BLOB)
```

O retângulo envolvente de cada bloco foi armazenado em quarto campos do tipo double representando os cantos inferior-esquerdo e superior-direito. A implementação em PostGIS ficou da seguinte forma:

```
raster(tileId:string, band:int,
resLevel: int, block_box: box,
tile:BLOB)
```

A maior diferença nas duas implementações está na forma de indexação do retângulo envolvente do bloco. No MySQL usa-se uma indexação por B-Tree (Bentley, 1975) e no PostGIS por R-Tree (Guttman, 1984).

3.1 Seleção de blocos pelo seu retângulo envolvente

Para testar a recuperação dos blocos que interceptam uma área de interesse, foram definidas 10 áreas: partindo da área total do plano de informação e diminuindo sucessivamente sua altura e largura em 5%. A Figura 1 mostra a SQL submetida a cada um dos SGDB's para selecionar esses blocos.

MySQL

```
SELECT *
FROM MosaicSJC
WHERE NOT(lower_x >= 419840 OR
          upper_x <= 397824 OR
          lower_y >= 7439872 OR
          upper_y <= 7431168) AND
      resolution_factor = 1;
```

PostgreSQL

```
SELECT *
FROM MosaicSJC
WHERE (block_box && '(410399, 7427399,
      409999,7426999)'::box) AND
      resolution_factor = 1;
```

Figura 1. SQL's de seleção de blocos

A Figura 2 mostra o tempo de resposta na seleção dos blocos por área de interesse nos dois SGDBs. A diferença de desempenho é esperada uma vez que a indexação espacial por R-Tree é mais eficiente.

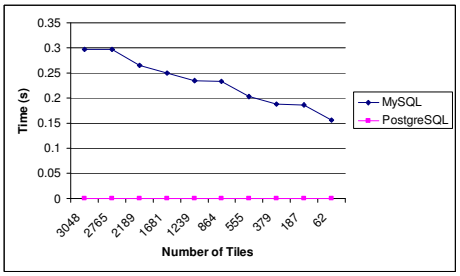

Figura 2. Seleção de blocos por área de interesse.

3.2 Seleção de blocos pelo identificador

A fim de conseguir um desempenho comparável nos dois SGDBs uma estratégia de recuperação diferente foi implementada para o caso do MySQL. Essa estratégia usa a ordem obtida pelo algoritmo de particionamento mostrado na seção 2.1. O algoritmo 3 mostra como encontrar a priori, os identificadores dos blocos que devem ser recuperados.

Algoritimo 3: SeleçãoBlocos
Entrada: uma area de interesse *Box*

1. Encontrar a identificação da coordenada inferior-esquerda de Box: LLTileId
2. Encontrar a identificação da coordenada superior-direita de Box: URTileId
3. Extraia as posições verticais e horizontais de LLTileId e URTileId: VP_LL, HP_LL, VP_UR, VP_LL
4. Gerar o conjunto de identificadores dos blocos que interceptam a area usando as posições geradas no passo 3:
   For *i=VP_LL* to *VP_UR*
   For *j=HP_LL* to *HP_UR*
   Gerar um novo identificador combinando *I* e *J*
5. Retornar o conjunto de identificadores

De posse da lista de identificadores que interceptam a área de interesse pode-se submeter uma consulta que selecione esses identificadores explicitamente como uma cláusula IN, como mostra a Figura 3.

```
SELECT *
FROM MosaicSJC
WHERE block_id IN (
'X820Y14530B0R1','X821Y14530B0R1',...);
```

Figura 3. Seleção de blocos por cláusula IN.

A Figura 4 mostra o tempo de resposta para seleção dos blocos que interceptam uma dada area de interesse usando a estratégia da cláusula IN. O gráfico representa o tempo de resposta para as mesmas 10 áreas de interesse mostradas na seção 3.1 nos dois SGDBs. É possível perceber que para essa estratégia o tempo de resposta no MySQL torna-se muito menor que no PostgreSQL.

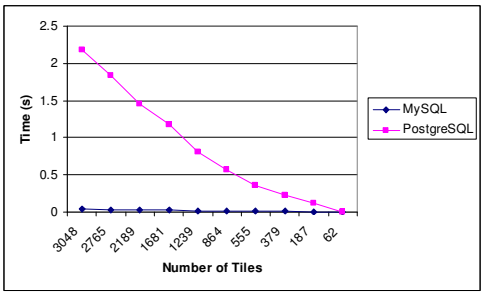

Figura 4. Tempo de resposta por cláusual IN.

A Figura 5 compara o tempo de resposta obtido nas duas melhores estratégias em cada

um dos SGDBs. Apesar do tempo de resposta no MySQL ainda ser bem maior que no PostgreSQL uma melhora de eficiência significante foi obtida.

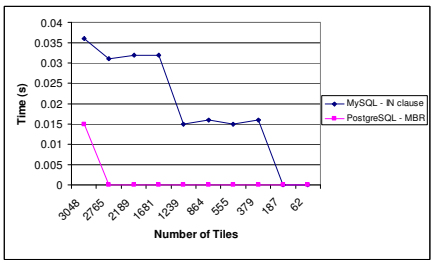

Figura 5. Tempo de resposta nas duas melhores estratégias

A estratégia da consulta por cláusula IN só pode ser efetivamente implementada quando combinada com o mecanismo de multi-resolução. Não é factível a submissão de SQL's de seleção com um número muito grande de identificadores na cláusula IN.

4. Conclusões

Nesse trabalho foi apresentado um modelo geral de um banco de dados matriciais espaciais, usando sistemas gerenciadores de bancos de dados relacionais. A solução proposta baseia-se em uma combinação de mecanismos de multi-resolução e particionamento,

Foram mostrados os algoritmos de pariticionamento e indexação de blocos, de forma que cada bloco possua uma identificação única. Foi mostrado como mapear uma posição de um elemento de um dados matricial special em coordenadas de linha e coluna para a identificação do bloco que o contém.

A modelo foi implementado em dois SGDBs: MySQL e PostgreSQL, usando a biblioteca TerraLib. O uso da interface genérica de banco de dados da TerraLib permite esconder as diferenças no modelo físico final do banco implementado nos dois SGDBs.

Quanto à recuperação, foi mostrado que para se conseguir um grau de eficiência na recuperação dos dados do banco, soluções específicas devem ser implementadas. Essas soluções levam em conta a disponibilidade de recursos de indexação espacial disponível no SGDB. Novamente foi usada a TerraLib e sua interface genérica de banco de dados para implementar essas diferentes estratégias, de modo que um tempo de resposta similar possa ser conseguido.

Esse trabalho está inserido no contexto do suporte a dados matriciais espaciais da biblioteca TerraLib (Vinhas e de Souza, 2005).

5. Referências

BARCLAY, T.; GRAY, J.; SLUTZ, D. Microsoft TerraServer: a spatial data warehouse. In: ACM SIGMOD Record , **Proceedings of the 2000 ACM SIGMOD International conference on Management of data**, 2000.

BAUMANN, P.; DEHMEL, A.; FURTADO, P.; RITSCH, R.; WIDMANN, N. The multidimensional database system RasDaMan. In: **ACM SIGMOD International Conference on Management of Data**, 1998, Seattle, Washington. p. 575–577.

BENTLEY, J. Multidimensional Search Trees Used for Associative Searching. **Communications of the ACM**, v. 18, p. 509-517, 1975.

BÖHM, C.; KLUMO, G.; KRIEGEL, H.-P. XZ-Ordering: A Space-Filling Curve for Objects with Spatial Extension. In: **Advances in Spatial Databases: 6th International Symposium**, 1999, Hong Kong, China. Springer-Verlag.

BURROUGH, P. A.; MCDONNELL, R. A. **Principles of Geographical Information Systems.** Oxford: Oxford University Press,1988.

CÂMARA, G., 2005, Representação Computacional de Dados Geográficos. In: CASANOVA, M. A.; CÂMARA, G.; DAVIS JR., C.; VINHAS, L.; QUEIROZ, G. R., eds., **Bancos de Dados Geográficos**: Curitiba, PR, Editora Mundo-Geo, p. 11-52.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing.** Prentice Hall,2002.

GUTTMAN, A. R-Trees: A Dynamic Index Structure for Spatial Searching. In: **Annual Meeting ACM SIGMOD**, 1984, Boston, MA. p. 47-57.

LAWDER, J. K.; KING, P. J. H., 2001, Querying multi-dimensional data indexed using the Hilbert space-filling curve, **ACM SIGMOD Record**, New York, USA, ACM Press, p. 19-24.

PATEL, J.; YU, J.; KABRA, N.; TUFTE, K.; NAG, B.; BURGER, J.; HALL, N.; RAMASAMY, K.; LUEDER, R.; ELLMANN, C.; KUPSCH, J.; GUO, S.; LARSON, J.; DEWITT, D.; NAUGHTON, J. Building a Scalable Geo-Spatial DBMS: Technology, Implementation, and Evaluation. In: **SIGMOD Conference**, 1997, Tucson, Arizona.

RAVADA, S.; SHARMA, J., 1999, Oracle8i Spatial: Experiences with Extensible Databases. In: GUTING, R. H.; PAPADIAS, D.; LOCHOVSKY, F., eds., **SSD'99**: Lecture

Notes on Computer Science 1651: Berlin, Springer-Verlag, p. 355-359.

SANTILLI, S.; HODGSON, C.; RAMSEY, P.; LOUNSBOURY, J.; BLASBY, D., 2005, PostGIS Manual, Refractions Research.

SHEKKAR, S.; CHAWLA, S. **Spatial Databases - A Tour.** Upper Saddle River, NJ: Prentice-Hall,2003.

VINHAS, L.; DE SOUZA, R. C. M., 2005, Tratamento de Dados Matriciais na TerraLib. In: CASANOVA, M. A.; CÂMARA, G.; DAVIS JR., C.; VINHAS, L.; QUEIROZ, G. R., eds., **Bancos de Dados Geográficos**: Curitiba, PR, Editora Mundo-Geo, p. 441-476.

VINHAS, L.; FERREIRA, K. R., 2005, Descrição da TerraLib. In: CASANOVA, M. A.; CÂMARA, G.; DAVIS JR., C.; VINHAS, L.; QUEIROZ, G. R., eds., **Bancos de Dados Geográficos**: Curitiba, PR, Editora Mundo-GEO.

WIDENIUS, M.; AB MYSQL; AXMARK, D. **MySQL Reference Manual.** O'Reilly,2002. 814 p.