Interface para Operações Espaciais em Banco de Dados Geográficos

KARINE REIS FERREIRA
JOÃO ARGEMIRO CARVALHO PAIVA
GILBERTO CÂMARA

INPE - Instituto Nacional de Pesquisas Espaciais, Caixa Postal 515, 12201 São José dos Campos, SP, Brasil
{ karine@dpi.inpe.br, miro@dpi.inpe.br, gilberto@dpi.inpe.br }

Abstract. This work presents a generic programming interface, or API (Application Programming Interface), for spatial operations in geographical database developed in the TerraLib environment - a base library for construction of geographical applications with integrated architecture. This API provides operations on geographical data stored in relational DBMS (RDBMS) and object-relational DBMS (ORDBMS). In the case of a new generation of ORDBMS which has a spatial extent, like Oracle Spatial, the API explores at most its resources to treat geographical data, for example, spatial indexes, operators and functions to manipulate and query these data through the query language SQL. The supplied operations of this API can be grouped as: (1) operations over vector data, for example, topological and metrical relation query, generation of a new geometry through a distance around an specific geometry (buffer), and set operations (intersection, union and difference); and (2) operations over raster data, as zonal operation and clipping based in a mask.

1 Introdução

Um número cada vez mais crescente de sistemas de informação vem incluindo técnicas para tratamento computacional de dados geográficos. Estes sistemas, chamados de sistemas de informação geográfica (SIG), foram inicialmente desenvolvidos nas décadas de 80 e 90 como sistemas "stand-alone", sem a capacidade de compartilhar ou gerenciar dados de forma eficiente. Adicionalmente, os SIG foram desenvolvidos como ambientes monolíticos, constituídos de pacotes de uso genérico de centenas de funções, o que dificultava sobremaneira seu aprendizado e uso por não-especialistas.

Para resolver estas limitações, nos anos recentes os SIGs estão evoluindo de modo a satisfazer duas grandes classes de requisitos:

(a) Gerenciar os dados espaciais através de Sistemas Gerenciadores de Banco de Dados (SGBDs) [1]. Com isso, os SIGs passam a utilizar os recursos oferecidos pelos SGBDs para controle e manipulação dos dados espaciais, como por exemplo, gerência de transações, controle de integridade e concorrência de acessos, ao invés de ter que implementá-los. A tecnologia de SIG baseada no uso de SGBD é chamada na literatura de "arquitetura integrada" [2].

(b) Compor um ambiente modular e extensível que permita a transição dos atuais sistemas monolíticos, que contêm centenas de funcionalidades, para uma nova geração de aplicativos geográficos (*spatial information appliances*), que são sistemas dedicados para necessidades específicas [3]. Esta nova geração de SIGs deverá ser modular, extensível e capaz de suportar a incorporação de sistemas independentes e autônomos de maneira modular, adicionando ao aplicativo geográfico novas funcionalidades conforme os requisitos da aplicação [4].

Portanto, para seguir essas duas classes de requisitos, a nova geração de SIGs deve suportar a incorporação de diferentes SGBDs, para gerenciar dados geográficos, de maneira modular.

Os aplicativos geográficos baseados na arquitetura integrada armazenam todos os tipos de dados geográficos, suas componentes espacial e alfanumérica, em um SGBD usualmente baseado na tecnologia objeto-relacional (SGBDOR) [5]. Para atender à evolução dos SIGs, os SGBDs objeto-relacionais estão sendo estendidos para tratar tipos de dados geográficos, chamados de Sistemas de Banco de Dados Geográficos ou extensões espaciais. Essas extensões fornecem funções e procedimentos que permitem armazenar, acessar e analisar dados geográficos de formato vetorial [6]. Existem hoje três extensões comerciais disponíveis: Oracle Spatial [7], IBM DB2 Spatial Extender [8] e Informix Spatial Datablade [9]. Há ainda a extensão PostGIS [10] para o SGBD PostgreSQL, que é objeto-relacional, gratuito e de código fonte aberto.

Para atender ao duplo requisito de construir aplicativos geográficos modulares baseados no suporte de SGBD relacionais e objeto-relacionais, está sendo desenvolvida a TerraLib [11], uma biblioteca de software livre base para a construção de uma nova geração de aplicativos geográficos baseados na arquitetura integrada. No entanto, o desenvolvimento de uma biblioteca eficiente e útil apresenta muitos

desafios, alguns dos quais são discutidos e resolvidos neste trabalho, a saber:

- Como compatibilizar, numa única interface de programação (API), implementações de tipos de dados espaciais realizadas por diferentes fabricantes de SGBDOR?
- Como estender as facilidades disponíveis nos SGBDOR, para tipos de dados necessários para os aplicativos geográficos, mas não disponíveis nestes sistemas?
- Como oferecer uma interface de programação genérica que permita a realização de operações espaciais sobre diferentes tipos de dados geográficos, de forma modular e eficiente para o programador de aplicações?

Este trabalho apresenta uma interface de programação genérica, ou API (*Application Programming Interface*), para operações espaciais em banco de dados geográficos que foi desenvolvida para a biblioteca TerraLib. Essa API é composta por funções que permitem realizar operações espaciais sobre dados geográficos armazenados em diferentes SGBDs, relacionais e objeto-relacionais. No caso da nova geração de SGBDORs que possuem extensões espaciais, como o Oracle Spatial, a API explora ao máximo seus recursos para tratar dados geográficos, como indexação espacial e operadores e funções para manipular esses dados.

2 Interface com diferentes SGBDs

A TerraLib se relaciona com diferentes SGBDs através de uma interface comum chamada `Database`, formada por duas classes bases `TeDatabase` e `TeDatabasePortal`, e um conjunto de *drivers* [12]. Para cada SGBD existe um *driver*, ou seja, uma classe específica que implementa todas as funcionalidades da interface comum conforme as particularidades e características de cada SGBD. Dentre essas funcionalidades estão inserção, recuperação e remoção de dados geográficos. Esses *drivers* exploram ao máximo os recursos oferecidos por cada SGBD, principalmente os que apresentam extensões espaciais. Atualmente, existem quatro *drivers* na TerraLib: TeOracleSpatial (Oracle Spatial), TePostgreSQL (PostgreSQL), TeMySQL (MySQL) e TeADO (acessar SGBDs via tecnologia ADO[13], como por exemplo Access e SQL Server). A Figura 2.1 ilustra a relação entre a interface comum, alguns *drivers* e os SGBDs.

Para a representação dos dados geográficos nos bancos de dados, a TerraLib possui um modelo de dados específico que é criado por cada *driver* no respectivo SGBD. Esse modelo é composto por esquemas de tabelas espaciais para armazenar dados geográficos vetorias e matriciais [14].

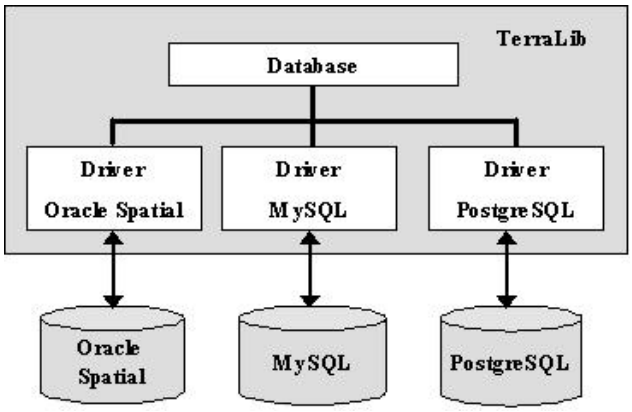

Figura 2.1 Interface com SGBDs

Os SGBDs que não possuem extensão espacial armazenam os dados geográficos vetoriais em campos longos binários, chamados de BLOB. Já os SGBDs que possuem essa extensão, como o Oracle Spatial, armazenam esses dados em tipos de dados específicos. Mas, em todos os SGBDs, os dados matriciais são armazenados em BLOBs, pois nenhuma extensão espacial oferece recursos para tratar esse tipo de dado.

Como exemplo, as figuras a seguir mostram o esquema da tabela de polígonos do modelo de dados da TerraLib representada em um SGBD relacional (Figura 2.2) e no Oracle Spatial (Figura 2.3). No SGBD relacional o polígono é armazenado em um campo do tipo longo binário (spatial_data) e, além disso, outras informações são armazenadas explicitamente como o número de coordenadas (num_coords) e o mínimo retângulo envolvente (lower_x, lower_y, upper_x e upper_y) de cada polígono. No Oracle Spatial, o polígono é armazenado em um campo do tipo SDO_GEOMETRY (spatial_data), que é um objeto espacial fornecido pela extensão. Neste caso, outras informações referentes aos polígonos não precisam ser armazenadas explicitamente, pois podem ser adquiridas através de métodos do objeto SDO_GEOMETRY.

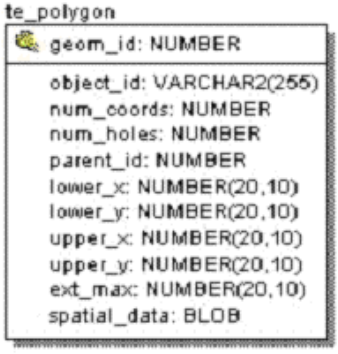

Figura 2.2 Tabela de polígonos em SGBDs Relacionais

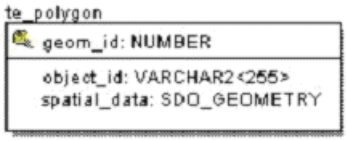

Figura 2.3 Tabela de polígonos no Oracle Spatial

Portanto, o modelo de dados da TerraLib é criado em cada SGBD com algumas diferenças considerando suas características e recursos para tratar dados geográficos. Assim, cada *driver* implementa métodos de inserção, recuperação e manipulação dos dados geográficos baseados no modelo criado em cada SGBD.

A estrutura de interface com SGBDs implementada na TerraLib é extensível e modular. Para incorporar um novo SGBD, basta desenvolver um *driver* específico que implemente todas as funcionalidades previstas na interface comum.

3 API para Operações Espaciais

A API apresentada neste trabalho foi implementada nas classes de interface entre a TerraLib e os SGBDs e é composta por um conjunto de funções que permitem realizar operações espaciais sobre dados geográficos, vetoriais e matriciais, armazenados em diferentes SGBDs, relacionais ou objeto-relacionais. No caso de SGBDORs que possuem extensão espacial, a API explora ao máximo seus recursos para computar as operações sobre os dados geográficos, como indexação espacial, funções e operadores utilizados em consultas SQL (*Structed Query Language*) [15].

As operações espaciais fornecidas pela API podem ser agrupadas em: (1) operações sobre dados vetoriais [1], como consultas sobre relações topológicas e métricas, geração de uma nova geometria a partir de uma distância em torno de uma geometria específica (*buffer*) e operações de conjunto (interseção, união e diferença); e (2) operações sobre dados matriciais [16], como operações zonais e recorte a partir de uma máscara (Mask). A Tabela 3.1 contém todas as operações disponíveis na API. Neste artigo serão mostradas algumas dessas operações, a saber: **`calculateArea`**, **`SpatialRelation`**, **`Buffer`**, **`Mask`** e **`Zonal`**.

Tabela 3.1 – Operações da API

Operações sobre dados vetoriais	Operações sobre dados matriciais
`CalculateLength`	`Zonal`
`CalculateArea`	`Mask`
`Buffer`	
`ConvexHull`	
`Centroid`	
`SpatialRelation`	
`NearestNeighbors`	
`CalculateDistance`	
`Intersection / Union`	
`Difference / Xor`	

A Figura 3.1 citua a API no contexto TerraLib, Aplicativos Geográficos e SGBDs. Os aplicativos geográficos desenvolvidos sobre a TerraLib utilizam as funções da API para realizarem operações espaciais sobre os dados geográficos armazenados nos SGBDs. A API é responsável pela interface com os diferentes SGBDs e pela computação dessas operações.

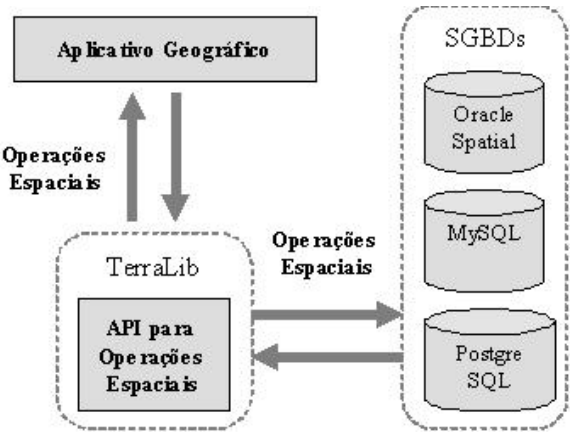

Figura 3.1 API para operações espaciais

Como trabalho relacionado com a API desenvolvida, pode ser citado o ArcSDE, um aplicativo que fornece uma interface entre os ArcGIS (ArcInfo, ArcEditor, ArcView GIS e ArcIMS) e diferentes SGBDs, como Oracle, Informix, IBM DB2 e Microsoft SQL Server [17].

O ArcSDE possui dois tipos de APIs para operações espaciais: (1) API que fornece funções para executar operações espaciais em memória; e (2) SQL API que apenas designa a execução de uma operação em SQL, que utiliza operadores e funções espaciais, para o Informix Spatial Datablade ou para o IBM DB2 Spatial Extender.

A API desenvolvida neste trabalho disponibiliza as operações espaciais em um nível maior de abstração do que as APIs do ArcSDE. A API da TerraLib encapsula as operações espaciais de forma que o usuário não precise ter conhecimento de como elas são computadas, se é pelo SGBD com extensão espacial ou em memória. Já as APIs disponíveis no ArcSDE exigem que o usuário

saiba como as operações devem ser computadas e escolher qual API utilizar.

4 Operações sobre dados vetoriais

As operações espaciais sobre dados vetoriais da API foram implementadas em dois níveis: (1) como métodos da classe base `TeDatabase`, sendo utilizadas por todos os SGBDs que não possuem extensão espacial, como Access, SQL Server, MySQL e PostgreSQL; e (2) como métodos do *driver* do Oracle Spatial, `TeOracleSpatial`, utilizando os recursos de sua extensão espacial.

As operações da classe `TeDatabase` utilizam as funções da TerraLib sobre os dados espaciais em memória, recuperados do SGBD. Essas operações seguem duas etapas:

(1) Recuperação das geometrias do SGBD através de uma consulta SQL: o SGBD retorna as geometrias como binários longos (BLOBs), o `TeDatabase` faz a leitura de cada BLOB e o transforma para uma das estruturas de dados geográficos da TerraLib (`TePolygon`, `TeLine2D`, `TePoint`, etc);
(2) Aplicação de uma função da TerraLib sobre a estrutura de dado geográfico gerada.

A Figura 4.1 mostra um exemplo da operação "Calcule a área do estado de Minas Gerais", seguindo as duas etapas descritas acima. Neste exemplo, a operação da API **`calculateArea`** utiliza uma função chamada `AREA` da TerraLib, que recebe um `TePolygon` como argumento e retorna sua área.

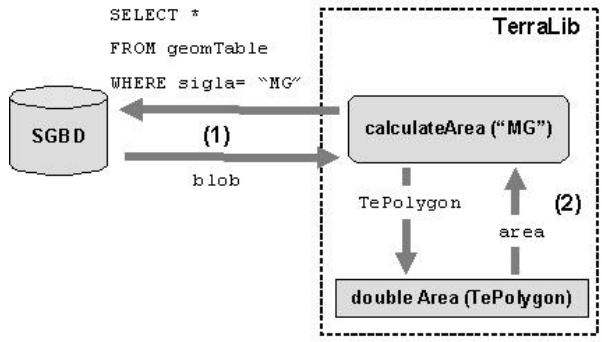

Figura 4.1 SGBDs relacionais

As operações do *driver* `TeOracleSpatial` consistem em montar uma consulta em SQL utilizando os operadores e funções da extensão espacial e designar sua computação para o SGBD. Por exemplo, a operação **`calculateArea`** monta uma consulta em SQL utilizando a função SDO_AREA do Oracle Spatial. Essa consulta é passada para o SGBD que a executa e retorna a área das geometrias especificadas na cláusula WHERE. A Figura 4.2 ilustra a operação "Calcule a área do estado de Minas Gerais", utilizando a extensão Oracle Spatial.

Observa-se que as operações do `TeDatabase` são executadas em memória pelas funções da TerraLib, e as operações do `TeOracleSpatial` são executadas pelo próprio SGBD, através de seus operadores e funções utilizados juntamente com a consulta SQL. Uma das características importantes dessa API é disponibilizar essas operações a um nível alto de abstração e transparência para os desenvolvedores de aplicativos geográficos.

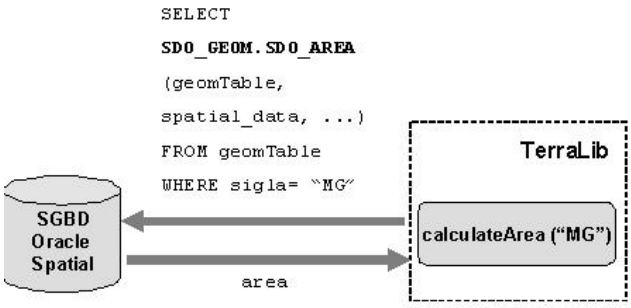

Figura 4.2 Oracle Spatial

4.1 Relações espaciais

A operação **`SpatialRelation`** da API retorna todos os objetos geográficos que possuem uma determinada relação topológica com um outro objeto específico. Essa função recebe como parâmetro a relação topológica a ser computada: DISJOINT, TOUCHES, CROSSES, WITHIN, OVERLAPS, CONTAINS, INTERSECTS, EQUALS, COVERS e COVEREDBY [18].

No `TeDatabase`, a operação **`SpatialRelation`** foi implementada em duas etapas, seguindo a mesma idéia do modelo de consulta do Oracle Spatial [7]. A primeira etapa é executada sobre aproximações das geometrias, ou seja, sobre seu mínimo retângulo envolvente (MBR), e a segunda, sobre as geometrias exatas. As duas etapas podem ser resumidas em:

(1) Recuperar do SGBD, através de uma consulta em SQL, somente as geometrias cujo mínimo retângulo envolvente têm alguma interseção com o mínimo retângulo envolvente da geometria específica;
(2) Para cada geometria recuperada verificar se existe uma determinada relação topológica com a geometria específica, utilizando as funções topológicas da TerraLib.

Como exemplo, considera-se a consulta "Selecione todos os estados do Brasil que fazem fronteira com o estado de Minas Gerais". A primeira etapa recupera do SGBD somente os estados cujo mínimo retângulo envolvente tem alguma interseção com o mínimo retângulo envolvente do estado de Minas Gerais, como ilustrado na Figura 4.3. E a segunda etapa utiliza a função topológica `TeTouches` da TerraLib entre os

estados recuperados na etapa anterior e o estado de Minas Gerais, retornando o resultado final da consulta. A Figura 4.4 mostra os resultados dessa consulta, onde os estados pintados de preto são os estados resultantes.

Figura 4.3 Primeira etapa - MBRs

Figura 4.4 Resultado da consulta

Seguindo essas duas etapas, a função topológica é aplicada somente a um subconjunto de geometrias retornadas na primeira etapa. Isso diminui o custo computacional da operação, já que as funções topológicas são robustas e computacionalmente caras.

No `TeOracleSpatial`, a operação **`SpatialRelation`** monta uma consulta em SQL utilizando o operador SDO_RELATE do Oracle Spatial e designa sua computação ao SGBD. A extensão espacial executa a consulta e retorna o resultado final. Esse operador recebe como parâmetro a relação topológica a ser consultada e utiliza a indexação espacial criada sobre as tabelas espaciais envolvidas na consulta para executar a operação. Como exemplo, a SQL gerada para solucionar a consulta anterior, "Selecione todos os estados do Brasil que fazem fronteira com o estado de Minas Gerais", é mostrada a seguir:

```
SELECT  t1.*
FROM    estados_Brasil t1,
        estados_Brasil t2
WHERE   t2.object_id = 'MG'
AND     SDO_RELATE
         (t1.spatial_data,
          t2.spatial_data,
         'mask= TOUCH
querytype = WINDOW')= 'TRUE'
```

Nesta consulta, foi considerado que as geometrias dos estados do Brasil estão armazenadas em uma tabela espacial chamada "estados_Brasil" em sua coluna chamada "spatial_data". A SQL gerada utiliza o operador SDO_RELATE, que recebe como parâmetro a relação topológica a ser consultada (mask=`TOUCH`) e retorna os estados mostrados na Figura 4..

4.2 Operação de Buffer

A operação **`Buffer`** da API gera uma nova geometria a partir de uma distância em torno de um objeto geográfico específico. No `TeDatabase` essa operação recupera a geometria referente ao objeto geográfico do SGBD e aplica a função `TeBuffer` da TerraLib sobre essa geometria.

No `TeOracleSpatial`, essa operação monta uma consulta em SQL utilizando a função SDO_BUFFER da extensão espacial. Como exemplo, a SQL gerada para a operação "Gere uma nova geometria a partir de uma distância de 50000 ao redor do estado de Minas Gerais", é mostrada a seguir:

```
SELECT SDO_GEOM.SDO_BUFFER
       (t.spatial_data,
       m.diminfo, 50.000)
FROM   estados_Brasil t,
       USER_SDO_GEOM_METADATA m
WHERE  m.table_name =
                'estados_Brasil'
AND    m.column_name =
                'spatial_data'
AND    t.object_id = 'MG';
```

A SQL gerada utiliza a função SDO_BUFFER e, ao ser executada pela extensão, retorna um objeto espacial do tipo SDO_GEOMETRY, que contém a nova geometria gerada:

```
SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY (1, 1005, 8, 1,
     2, 2, 5, 2, 1, 9, 2, 2, 13, 2,
     1, 15, 2, 2, 19, 2, 1, 23, 2, 2,
     27, 2, 1),
SDO_ORDINATE_ARRAY (18, 17, 19, 18,
     18, 19, 16.4142136, 19,
     14.7071068, 20.7071068, 14, 21,
     13.2928932, ...., 15.6173166,
     17.0761205, 16, 17, 18, 17))
```

O objeto espacial SDO_GEOMETRY do Oracle Spatial é composto por dois vetores: SDO_ORDINATES_ARRAY, que contém as coordenadas dos elementos que compõem o objeto; e SDO_ELEM_INFO_ARRAY, que contém informações sobre o tipo de cada elemento e como cada um deve ser interpretado. A função SDO_BUFFER do Oracle Spatial retorna um SDO_GEOMETRY do tipo POLYGON, que é formado por um conjunto de elementos, onde cada elemento pode ser do tipo polígono simples (cujos vértices são conectados por linhas retas), polígono circular (cujos vértices são formados por arcos circulares) ou polígono composto (formado por alguns vértices conectados por linhas retas e outros por arcos circulares).

Assim, a operação **Buffer** do `TeOracleSpatial` deve interpretar o objeto SDO_GEOMETRY retornado pelo Oracle Spatial e gerar um polígono em memória, ou seja, uma estrutura do tipo `TePolygon` da TerraLib.

Um arco circular no Oracle Spatial é representado apenas por três pontos e o tipo `TePolygon` da TerraLib é formado por pontos conectados por segmentos de linha reta. Portanto, para representar um polígono composto do Oracle Spatial no `TePolygon`, para cada arco desse polígono é necessário gerar pontos intermediários que, ao serem ligados por segmentos de linha reta, se aproximem de um arco. Para isso, foi necessário implementar uma função que gera um arco formado por *n* pontos intermediários a partir de três pontos de um arco retornados pelo Oracle Spatial.

Essa função, chamada `TeGenerateArc`, foi implementada na TerraLib e recebe como argumentos três pontos (pt_1, pt_2 e pt_3) e o número *n* de pontos intermediários que devem ser gerados. Como resultado, essa função gera um arco formado por esses *n* pontos que passa pelos três pontos de entrada [19].

5 Operações sobre dados matriciais

As operações sobre dados matriciais da API, **Zonal** e **Mask**, foram implementadas somente como métodos da classe `TeDatabase`, pois nenhuma extensão espacial oferece recursos para tratar este tipo de dado. Portanto, esses métodos são utilizados por todos os SGBDs que possuem uma interface com a TerraLib, como Access, MySQL, PostgreSQL e Oracle Spatial.

A classe da TerraLib específica para representar dados matriciais é chamada `TeRaster`. Essa classe, juntamente com outras estruturas da biblioteca, é capaz de manipular um dado matricial armazenado tanto em arquivos de diferentes formatos, como JPEG e geoTIFF, quanto em SGBDs [14].

Como essas duas operações são executadas sobre regiões específicas de um dado *raster* delimitadas por polígonos, foi necessário implementar um mecanismo que percorresse esse dado somente na região interna ou externa a um polígono. Esse mecanismo foi implementado como um iterador, chamado **`iteratorPoly`**, sobre a classe `TeRaster`. Esse iterador é capaz de percorrer um dado *raster* segundo alguma estratégia e um polígono limitante. O polígono limitante define a área de interesse e a estratégia de percorrimento define se o iterador percorrerá a área externa ou interna desse polígono.

Iteradores são uma generalização de ponteiros; são objetos que apontam para outros objetos [20]. Segundo Stroustrup [21], um iterador é uma abstração da noção de um ponteiro para uma sequência. O conceito de iterador é muito importante para a programação genérica, pois fornece uma interface entre as estruturas de dados e os algoritmos, permitindo assim um desacoplamento entre os dois [22]. Esse desacoplamento é essencial para a implementação de algoritmos genéricos e reusáveis [23].

A operação **Mask** da API consiste em recortar um dado matricial a partir de uma máscara definida por um polígono. O recorte pode ser feito a partir da região interna ou externa ao polígono, dependendo da estratégia escolhida. O resultado dessa operação é um outro objeto do tipo `TeRaster` que contém o *raster* recortado.

A operação **Zonal** consiste em calcular as estatísticas de uma determinada região ou zona de um dado matricial delimitada por um polígono. O resultado dessa operação é um conjunto de estatísticas referentes à região ou zona. As estatísticas calculadas são: contagem, valor mínimo e máximo, soma, média, desvio padrão, variância, assimetria, curtose, amplitude, mediana, coeficiente de variação e moda.

Como exemplo dessas operações sobre dados matriciais, são mostradas as figuras a seguir. A Figura 5.1 mostra uma imagem de Brasília e um polígono, pintado de branco, que define uma região de interesse. Esse polígono será usado como uma máscara na operação **Mask** e como uma zona de interesse na

operação **`Zonal.`** A Figura 5.2 mostra a imagem de Brasília recortada a partir dessa máscara. Essa imagem foi gerada pela operação **`Mask`** da API, considerando a região interna da máscara. A Figura 5.3 mostra a interface que mostra as estatísticas geradas da zona de interesse da imagem de Brasília.

Figura 5.1 Imagem de Brasília

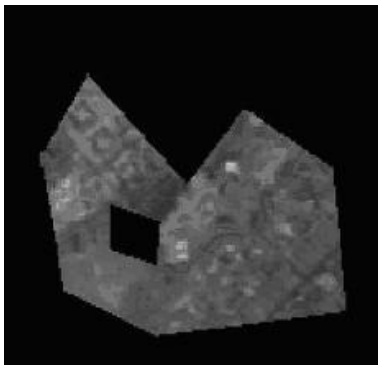

Figura 5.2 Imagem recortada (região interna)

Estatísticas do raster

Estatísticas	Banda 0	Banda 1	Banda 2
soma	851164.000000	862173.000000	1091580.000000
valor máximo	205.000000	165.000000	206.000000
valor mínimo	30.000000	29.000000	28.000000
contagem	11365.000000	11365.000000	11365.000000
desvio padrão	18.811450	12.338327	24.338319
média	74.893445	75.862121	96.047514
variância	353.870652	152.234311	592.353748
assimetria	1.116281	1.030130	0.300146
curtose	5.929326	6.152706	3.588302
amplitude	175.000000	136.000000	178.000000
mediana	72.000000	74.000000	96.000000
coeficiennte de variação	25.117619	16.264147	25.339873
moda	70.000000	74.000000	97.000000

Figura 5.3 Estatísticas geradas

6 Conclusão e trabalhos futuros

Este trabalho apresentou o desenvolvimento de uma API (*Application Programming Interface*) para operações espaciais em banco de dados geográficos, implementada na TerraLib, uma biblioteca base para a construção de aplicativos geográficos de arquitetura integrada. Essa nova arquitetura consiste em utilizar SGBDs para armazenar e gerenciar dados geográficos, explorando seus recursos como controle de integridade e concorrência, gerência de transações, restrições de acessos e mecanismos para recuperação de falhas.

A API desenvolvida permite encapsular diferentes implementações de tipos de dados espaciais realizadas pelos diferentes fabricantes de SGBDs. Além disso, ela estende as facilidades disponíveis nestes sistemas, como no caso de suporte a dados matriciais. Como a TerraLib é uma biblioteca base para a construção de aplicativos geográficos, a API desenvolvida fornece tais funcionalidades a um nível maior de abstração para os desenvolvedores desses aplicativos. Assim, esse desenvolvedor pode utilizar as operações espaciais disponíveis na biblioteca sem precisar ter conhecimento de como são computadas, podendo então, se dedicar à implementação de outras funcionalidades, como por exemplo, visualização dos dados, interface com o usuário e ferramentas para análise espacial.

Como trabalhos futuros, podemos citar alterações nas operações sobre dados matriciais da API quando as extensões espaciais dos SGBDORs começarem a oferecer recursos para tratar esses dados. A proposta de tratar dados matriciais já foi feita por algumas extensões, como por exemplo, Oracle Spatial.

7 Referências

[1] Rigaux, P.S., M.; Voisard, A., *Spatial Databases with application to GIS*. 2002, San Francisco: Morgan Kaufmann. 408.

[2]Câmara, G.C., M. A.; Hemerly, A. S.; Magalhães, G. C.; Medeiros, C. B., *Anatomia de Sistemas de Informação Geográfica*. 1996, Campinas: 10ª Escola de Computação. 193.

[3] Egenhofer, M. *Spatial Information Appliances: A Next Generation of Geographic Information Systems.* in *First Brazilian Workshop on GeoInformatics*. 1999. Campinas, SP.

[4] Voisard, A.S., H., *Abstraction and Decomposition in Open GIS*. 1997, International Computer Science Institute: Berkeley, California.

[5] Lassen, A.R.O., J.; Osterbye, K., *Object Relational Modeling*. 1998, Centre for Object Technology (COT).

[6] Güting, R., *An Introduction to Spatial Database Systems.* VLDB Journal, 1994. **3**.

[7] Ravada, S.S., J. *Oracle8i Spatial: Experiences with Extensible Databases.* in *SSD'99*. 1999.

[8] Corporation, I., *DB2 Spatial Extender User's Guide and Reference*. 2001.

[9] Corporation, I., *Working with the Geodetic and Spatial DataBlade*. 2002.

[10] Ramsey, P., *PostGis Manual*. 2002.

[11] Câmara, G.S., R. C. M.; Pedrosa, B.; Vinhas, L.; Monteiro A. M.; Paiva, J. A. C. P.; Gattas, M. *TerraLib: Technology in Support of GIS Innovation*. in *II Workshop Brasileiro de Geoinformática*. 2000. Curitiba, PR.

[12] Ferreira, K.R.Q., G. R.; Paiva, J. A. C.; Souza, R. C. M.; Câmara, G. *Arquitetura de Software para Construção de Bancos de Dados Geográficos com SGBD Objeto-Relacionais*. in *XVII Simpósio Brasileiro de Banco de Dados*. 2002. Gramado, RS.

[13] Roff, J.T., *ADO: ActiveX Data Objects*. 1 st ed. 2001: O´Reilly & Associates. 672.

[14] INPE, *TerraLib*. 2003.[*www.terralib.org*]

[15] Korth, F.H.S., A., *Sistemas de Bancos de Dados*. 1994, São Paulo: McGraw-Hill. 693.

[16] Tomlin, C.D., *Geographic Information Systems and Cartographical Modeling.* 1990, New York: Prentice-Hall.

[17] ESRI, *ArcSDE*. 2003. [*http://arcsdeonline.esri.com/index.htm*]

[18] Egenhofer, M.F., R., *On the Equivalence of Topological Relations.* International Journal of Geographical Information Systems, 1995. **9**(2): p. 133-152.

[19] University, D., *The Math Forum*. 2003.

[20] Austern, M.H., *Generic Programming and the STL: Using and Extending the C++ Standard Template Library*. 1998, Massachusetts: Addison-Weslwy. 548.

[21] Stroustrup, B., *The C++ Programming Language*. 1999: USA: Addison-Wesley.

[22] Köthe, U., *STL-Style Generic Programming with Images*, in *C++ Report Magazine*. 2000.

[23]Vinhas, L.Q., G. R.; Ferreira, K. R.; Câmara, G.; Paiva, J. A. C. *Programação Genérica Aplicada a Algoritmos Geográficos.* in *IV Simpósio Brasileiro de GeoInformática*. 2002. Caxambu, MG.