Armazenamento e Processamento de Grandes Grafos em Bancos de Dados Geográficos

Eric S. Abreu[1], **Sergio Rosim**[1], **João Ricardo de F. Oliveira**[1], **Gilberto Ribeiro**[1], **Luciano V. Dutra**[1]

[1]Instituto Nacional de Pesquisas Espaciais (INPE)
Caixa Postal 1758 – 12227-010 – São José dos Campos – SP– Brasil

{eric,sergio,joao,gribeiro,luciano}@dpi.inpe.br

Abstract. *This paper presents a methodology that allows the storage and manipulation of large graphs in geographic database by defining a set of relational tables. These tables represent the graph by using the connection information and attributes, as well as its metadata. The graphs discussed in this work are those that can be spatially defined. A library of geo-processing and a cache policy are also used to optimize the access to this data.*

Resumo. *Este trabalho apresenta uma metodologia que permite o armazenamento e manipulação de grandes grafos em banco de dados geográficos através da definição de um conjunto de tabelas relacionais. Essas tabelas representam o grafo através das informações de conexão e atributos, bem como seus metadados. Os grafos abordados nesse trabalho são aqueles que podem ser espacialmente definidos. Uma biblioteca de geo-processamento e uma política de cache também são utilizadas para otimizar o acesso a esses dados.*

Palavras-chave: *Grafos, SGBD, Cache, Terralib.*

1. Introdução

Utilizamos a estrutura de grafos quando queremos representar a conectividade e a relação entre objetos de um determinado conjunto. Matematicamente os grafos são definidos por um par ordenado *(V, A)*, onde *V* é um conjunto de vértices e *A* uma relação binária sobre *V*, cujos elementos são denominados de arestas [Bollobás 1998].

No caso específico deste trabalho estamos interessados em grafos que sejam espacialmente definidos, ou seja, que seus vértices possuam localizações espaciais bem definidas, tais como redes hidrológicas e fluxos. Ao armazenarmos esse tipo de informação em banco de dados geográficos, nos é permitido uma série de operações que facilitam o processamento desses dados. Atualmente existem bancos de dados relacionais dedicados ao armazenamento de grafos (Oracle Spatial Network Data Model [Murray 2009], PgRouting [Kastl and Junod 2010], etc) ou mesmo bancos de dados não relacionais que são específicos para o armazenamento de grafos (Neo4J [Team 2012], DEX [Martínez-Bazan et al. 2007], etc).

A proposta deste trabalho é criar um arcabouço para manipulação de grafos em bibliotecas geográficas que sejam capazes de tratar os casos citados acima, tendo como principal abordagem a definição de um modelo de grafos flexível para armazenamento em banco de dados relacionais, bem como a criação de funções de recuperação e persistência que

sejam capazes de manipular grandes quantidades de dados. Para a implementação desse projeto escolhemos a biblioteca geográfica *Terralib5*.

O projeto TerraLib [Câmara et al.] visa atender grandes demandas institucionais na área de Geoinformática, criando um ambiente para pesquisa e desenvolvimento de inovações em geo-processamento. Este trabalho se encontra vinculado ao projeto Terralib5 [Team] fornecendo subsídios técnicos e científcos para sua elaboração.

A Terralib5, Figura 1, é uma plataforma de *software* tendo seu núcleo escrito em C++ e tem como principais características [Queiroz et al. 2011]:

- Acesso aos dados: acesso a diferentes tipos de fontes de dados (SGBD's, dados vetoriais, imagens, serviços web entre outros).
- Persistência/mapeamento: mapea dados de diversas fontes de dados para diferentes finalidades.
- Estruturas de agregação: fornece um mecanismo extensível capaz de introduzir novas representações de alto nível.

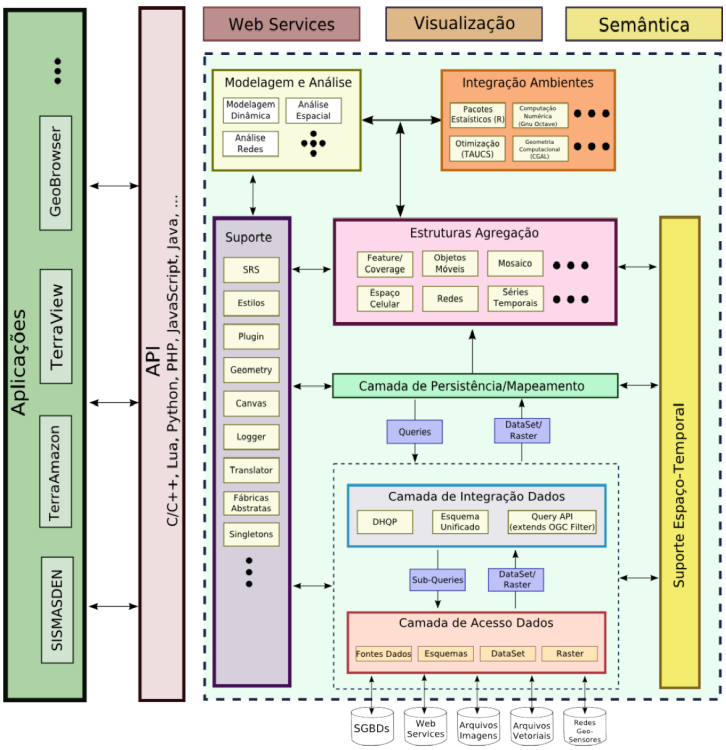


Figura 1. Esquema de representação da Terralib5.

Baseado nessa ultima característica, que nos permite criar novos tipos de representação, e utilizando das facilidades de acesso aos dados é que iremos implementar nosso projeto fazendo uso dessa tecnologia.

A seção 2 descreve a metodologia desenvolvida, a seção 3 exemplifica uma estratégia de extração de grafos; e por fim a seção 4 apresenta as conclusões.

2. Metodologia

Esta seção apresenta de que maneira uma estrutura de grafo pode ser armazenada e como recuperar essas informações de forma eficiente através do uso de uma biblioteca

de geo-processamento, mantendo a robustez do processamento independente do tamanho do grafo.

2.1. Modelo de Persistência

Para que um grafo possa ser armazenado e posteriormente recuperado é necessário um conjunto de metadados que o descrevam. Informações que indiquem quais atributos estão sendo associados aos grafos, localização e organização dos dados do grafo, são de fundamental importância para sua utilização. A Figura 2 e a Tabela 1 exemplificam este modelo.

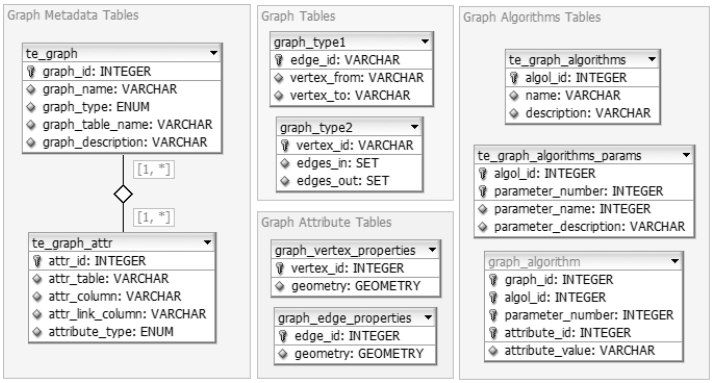


Figura 2. Modelo de Persistência.

Tabelas	
te_graph	Metadados de um grafo.
te_graph_attr	Metadados dos atributos de um grafo.
graph_type1	Tabela de dados ordenado por arestas.
graph_type2	Tabela de dados ordenado por vértices.
graph_vertex_properties	Propriedades dos vértices.
graph_edge_properties	Propriedade das arestas.
te_graph_algorithms	Metadado dos algoritmos.
te_graph_algorithms_params	Metadado dos parâmetros dos algoritmos.
graph_algorithms	Lista dos grafos associados a algoritmos.

Tabela 1. Tabelas do Modelo de Persistência

As tabelas que possuem prefixo "'te_"' fazem parte do modelo conceitual e sempre estão presentes no banco de dados. As outras tabelas irão depender da forma de como o grafo será armazenado e utilizado.

2.2. Modelo de Dados

O modelo de dados utilizado para representar os elementos do grafo usa o conceito de classes. É possível definir um modelo flexível através da definição abstrata das principais operações; funções de inserção, remoção e acesso aos elementos são definidas em uma classe virtual chamada de *AbstractGraph*, facilitando extensões para tipos de grafos específicos (como bidirecionais, unidirecionais e etc.). A Figura 3 ilustra o modelo de dados.

Uma breve descrição das classes é feita abaixo:

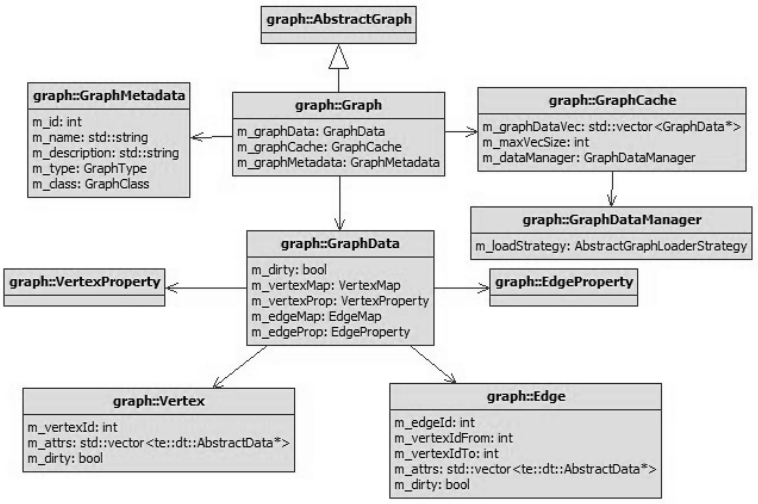


Figura 3. Modelo de Dados.

- *Graph* - Uma implementação concreta da classe AbstractGraph.
- *GraphData* - Representação de um pacote de dados.
- *GraphDataManager* - Acesso aos dados diretamente no repositório.
- *GraphCache* - Representação de uma estrutura de cache.
- *Vertex* - Representação do objeto vértice.
- *Edge* - Representação do objeto aresta.
- *VertexProperty* - Lista dos metadados dos atributos associado aos vértices.
- *EdgeProperty* - Lista dos metadados dos atributos associados às arestas.

2.3. Recuperação e Armazenamento

A classe *GraphData* foi projetada com a finalidade de criar um conceito de pacote, permitindo que os dados possam ser agrupados e gerenciados de forma simples. Esta classe é composta por dois containers, um de vértices e outro de arestas. A sequência de operações necessárias para acessar um elemento é descrito a seguir.

- Requisição de um elemento (vértice ou aresta) utilizando a classe abstrata *AbstractGraph*.
- Classe concreta *Graph* verifica em seu pacote corrente (classe *GraphData*) pelo elemento.
- Caso nao encontre, o *cache* deve ser consultado.
- Feito uma pesquisa em todos os pacotes presentes no *cache* pelo elemento requisitado.
- Se o elemento nao estiver carregado é necessário acessar a fonte de dados.
- Classe *GraphDataManager* faz a ponte com os dados e busca o elemento seguindo uma estratégia de busca.
- Um conjunto de dados é carregado junto com o elemento procurado.

2.4. Acesso aos Dados

A classe *GraphDataManager* é considerada uma ponte para acesso aos dados do grafo e fornece métodos de acesso aos dados dado um identificador de um elemento, seja vértice

ou aresta. Esta classe possui um atributo que é a classe abstrata *AbstractGraphLoaderStrategy*.

Foi definido o conceito de *LoaderStrategy* que determina a maneira que o dado deve ser carregado. Algumas estratégias foram consideradas, como mostra a Figura 4.

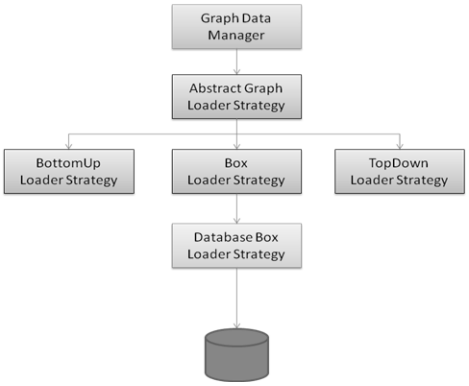


Figura 4. Estratégias de carga dos dados.

- *BottomUp*: carrega um conjunto de objetos a partir do objeto procurado, percorrendo o grafo de forma reversa (grafo direcionado).
- *TopDown*: carrega um conjunto de objetos a partir do objeto procurado, percorrendo o grafo seguindo seu fluxo normal (grafo direcionado).
- *Box*: carrega um conjunto de objetos tendo como objeto central o item procurado.

2.5. Cache

Uma parte importante deste projeto e que irá auxiliar no desempenho de acesso aos elementos do grafo é a estrutura de *cache*. A classe *GraphCache* é constituída por um vetor de *GraphData* e possui um atributo que é a classe *GraphDataManager*, utilizado quando um elemento desejado não está presente no *cache* (Figura 5).

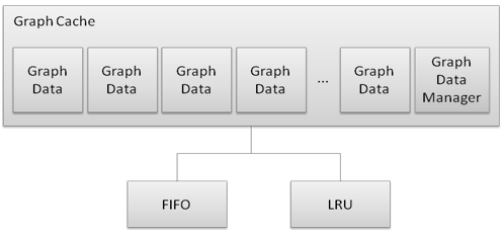


Figura 5. Modelo de Cache.

Duas observações importantes a fazer a respeito dessa estrutura de *cache* são:

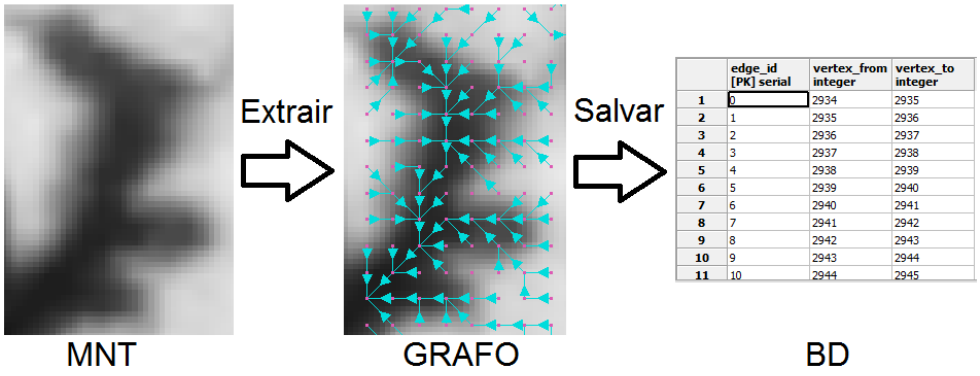
- Tamanho do vetor: quanto mais pacotes o cache possuir, maiores são as chances de o elemento procurado estar carregado, porém em cada busca ele terá que pesquisar em mais pacotes para ver se o elemento está carregado.
- Política de *cache*: o tamanho do vetor de armazenamento dos pacotes é configurável, mas uma vez atingido o limite máximo é necessário começar a descartar pacotes. Foram definidas duas políticas de *cache*: *FIFO (First In First Out)* e *LRU (Least Recently Used)*.

Essa política passa a ser simples devido ao fato de os elementos estarem agrupado em pacotes. O controle é feito por pacotes e não por elementos individuais, evitando uma sobrecarga de checagens e controles. Em testes realizados, a estratégia de bloco único é 9%, em média, mais rápido do que a estratégia por múltiplos blocos. Entretanto, ao utilizar um bloco único, as políticas de *cache* se tornam mais complexas, sendo necessário um controle individual de cada elemento para verificar seus acessos.

Outro fator importante dessa estrutura é a paralelização: se usarmos o acesso ao cache de forma seqüencial, toda a aplicação irá ficar parada esperando que um novo elemento seja carregado, criando um grande gargalo no processamento.

3. Extração dos Grafos

Para comprovar a real capacidade de armazenamento do grafo no SGBD e posteriormente sua recuperação, foi implementada uma metodologia de extração de grafos a partir de um *MNT (Modelo Numérico de Terreno)*, seguindo a especificação definida em [Rosim 2008].



	edge_id [PK] serial	vertex_from integer	vertex_to integer
1	0	2934	2935
2	1	2935	2936
3	2	2936	2937
4	3	2937	2938
5	4	2938	2939
6	5	2939	2940
7	6	2940	2941
8	7	2941	2942
9	8	2942	2943
10	9	2943	2944
11	10	2944	2945

Figura 6. Extração de grafos a partir de um MNT.

Em testes realizados, os grafos foram corretamente extraídos e armazenados no banco dados.

4. Conclusão

Percebemos que o modelo adotado é eficiente, pois permite que aplicações distintas, como modelagem hidrológica [Rosim 2008] e fluxos [de Oliveira 2005] (ambas feitas usando a TerraLib) possam utilizar o mesmo arcabouço, independente do tamanho dos dados a ser processado. O modelo de tabelas para armazenar o grafo que tem demonstrado melhores resultados é o ordenado por arestas. Mesmo considerando que o banco de dados relacional não tem a mesma eficiência que um banco de dados para grafos, vale ressaltar que o modelo se adapta muito bem a bancos de dados espaciais, o que facilita sua utilização junto à *TerraLib*.

Referências

Bollobás, B. (1998). *Modern graph theory*. Springer-Verlag.

Câmara, G., Vinhas, L., Ferreira, K. R., De, G. R., Cartaxo, R., Souza, M., Miguel, A., Monteiro, V., Carvalho, M. T. D., Casanova, M. A., and De, U. M. Terralib: An open source gis library for large-scale environmental and socio-economic applications.

de Oliveira, E. X. G. (2005). *A multiplicidade do único território do SUS*. PhD thesis, Escola Nacional de Saúde Pública.

Kastl, D. and Junod, F. (2010). W-10: Foss4g routing with pgrouting tools, openstreetmap road data and geoext. In *FOSS4G*.

Martínez-Bazan, N., Muntés-Mulero, V., Gómez-Villamor, S., Nin, J., Sánchez-Martínez, M.-A., and Larriba-Pey, J.-L. (2007). Dex: high-performance exploration on large graphs for information retrieval. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*.

Murray, C. (2009). *Oracle Spatial Topology and Network Data Model Developers Guide*. Oracle.

Queiroz, G. R., Ferreira, K. R., Vinhas, L., Câmara, G., Monteiro, A. M. V., Garrido, J. P., Haram, L., Xavier, M., Castejon, E. F., and Souza, R. C. M. (2011). Terralib5.0: Supporting data-intensive giscience. In *X Worcap - Instituto Nacional de Pesquisas Espaciais - INPE*.

Rosim, S. (2008). *Estrutura baseada em grafos para representação unificada de fluxos locais para modelagem hidrológica distribuída*. PhD thesis, Instituto Nacional de Pesquisas Espaciais.

Team, T. N. (2012a). *The Neo4j Manual v1.8.M06*. Neo Technology.

Team, T. T. (2012b). The design and implementations of terralib5. In *TerraLib5 Wiki - http://www.dpi.inpe.br/terralib5/wiki*.