UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA
2008.1

Estendendo o Servidor OLAP Mondrian com UDF Envolvendo Operadores Espaciais

TRABALHO DE GRADUAÇÃO

Aluno: Fábio Rocha de Pinho (frp@cin.ufpe.br)
Orientador: Robson do Nascimento Fidalgo (rdnf@cin.ufpe.br)
Co-Orientador: Joel da Silva (js@cin.ufpe.br)

Recife, 17 de junho de 2008

UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA
2008.1

Estendendo o Servidor OLAP Mondrian com UDF Envolvendo Operadores Espaciais

TRABALHO DE GRADUAÇÃO

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Pernambuco, como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aluno: Fábio Rocha de Pinho (frp@cin.ufpe.br)
Orientador: Robson do Nascimento Fidalgo (rdnf@cin.ufpe.br)
Co-Orientador: Joel da Silva (js@cin.ufpe.br)

Recife, 17 de junho de 2008

“Tenha sempre como meta muita força, muita determinação e sempre faça tudo com muito amor e com muita fé em Deus, que um dia você chega lá. De alguma maneira você chega lá.” (Ayrton Senna)

A Deus, que esteve comigo durante
toda esta jornada.

Aos meus pais, meus exemplos.
Meu reconhecimento e gratidão
pela paciência, compreensão, amor e
por tudo mais que me proporcionaram,
em especial este curso.

Agradecimentos

Eu gostaria de expressar os meus mais profundos e sinceros agradecimentos...

A Deus, por estar presente, guiando esta caminhada com saúde e paz.

A minha família, em especial à minha mãe, pelo amor sem fronteiras, e ao meu pai, pelo exemplo, amizade, apoio e incentivo.

Aos Professores do Centro de Informática, em especial ao Professor Robson Fidalgo, por despertar o meu interesse na área, à Professora Valéria Times, pelos puxões de orelha mais que enriquecedores, e ao Professor Fernando da Fonseca, pelos conhecimentos e amizade.

A Joel da Silva, pela paciência, orientação e contribuições ao longo deste trabalho.

Aos amigos, em especial a Bruno Lima e Humberto Moreira, pelas trocas de idéias, companheirismo e incentivo.

A todos que acreditaram, apoiaram e participaram de alguma forma desta jornada.

Resumo

Muitos trabalhos têm sido desenvolvidos na área de Banco de Dados com o objetivo de integrar as funcionalidades de processamento analítico-multidimensional e geográfico presente em ferramentas OLAP e SIG, respectivamente. Entretanto, a integração destas áreas não é trivial devido ao fato de que ambas foram concebidas para propósitos distintos.

Este trabalho tem como objetivo estender o Servidor OLAP Mondrian com funções envolvendo operadores espaciais, tais como topológicos, métricos e posicionais. Para a extensão, utilizaremos um recurso do Mondrian chamado UDF (User Defined Functions – Funções Definidas Pelo Usuário), que permite que o usuário defina novas funções e em seguida utilize-as em construções da linguagem GeoMDQL para consultar os dados de um Data Warehouse Geográfico.

Este trabalho está inserido no contexto do projeto GOLAPA, que já possui outras pesquisas relacionadas com a integração dos processamentos analítico-multidimensional e geográfico, como por exemplo, a definição de esquemas para DWG, metamodelos de integração e uma linguagem de consulta para DWG.

Abstract

Many studies have been developed in the area of Data Base in order to integrate the functions of multi-analytical processing and geographic presence in OLAP tools and GIS, respectively. Meanwhile, the integration of these areas is not trivial due to the fact that both were designed for different purposes.

This paper aims to extend the OLAP Mondrian server with functions involving spatial operators, such as topological, metric and positional. To accomplish this, we use a feature of Mondrian called UDF (User Defined Functions), which allows the user to define new functions and then use them in constructions of language GeoMDQL to retrieve data from a Geographic Data Warehouse.

This work is inserted in the context of GOLAPA project, which already has other research related to the integration of multi-analytical processing and geographic, for example, the definition of schemes for GDW, meta-integration and a query language for DWG.

Índice

Lista de quadros

1. Introdução

1.1. MOTIVAÇÃO

O mundo moderno e dinâmico que vivenciamos produz uma quantidade de informação cada vez maior e as organizações, uma vez inseridas neste cenário, precisam de informações relevantes que reflitam a realidade dos seus ambientes internos e externos de forma que as decisões sejam baseadas na realidade organizacional.

Portanto, é requisitado cada vez mais esforços na organização e manipulação destas informações e, além disso, que informações antes não tão importantes no processo decisório sejam incluídas para que este seja efetuado de forma eficaz, como é o caso das informações geográficas.

Muitos trabalhos têm sido desenvolvidos na área de Banco de Dados com o objetivo de integrar as funcionalidades de processamento de dados analíticos e geográficos [1, 2, 3, 4], para que os objetivos citados acima sejam alcançados. Este tipo de ambiente vem sendo referido como SOLAP (Spatial OLAP) [5], porém, a integração destas áreas não é trivial devido ao fato de que ambas foram concebidas para propósitos distintos.

Em [6], Silva propõe uma linguagem de consulta denominada GeoMDQL, a qual, até a defesa deste trabalho, é única disponível na literatura que é destinada especificamente para SOLAP, ou seja, ela possui uma sintaxe única para incorporar operadores espaciais e multidimensionais nas consultas.

Parte destes operadores é implementada através da utilização de UDF [7], apresentadas neste trabalho, as quais contribuem para aumentar o potencial de consulta da linguagem GeoMDQL, pois elas adicionam ao servidor OLAP Mondrian [7] funcionalidades para manipulação de dados geográficos.

1.2. OBJETIVOS

A partir do exposto na motivação acima, este trabalho tem como objetivo implementar algumas das funções definidas em [6], as quais serão utilizadas na linguagem GeoMDQL e contribuirão para estender o Servidor OLAP Mondrian com funcionalidades envolvendo operadores espaciais, tais como topológicos, métricos e posicionais, para serem utilizados em consultas multidimensionais.
Para a concretização deste objetivo, foi utilizado um recurso do Mondrian chamado UDF (User Defined Functions – Funções Definidas Pelo Usuário), que permite ao usuário definir novas funções e em seguida utilize-as para consultar os dados de um Data Warehouse Geográfico.

Serão apresentadas também estas funções em funcionamento no estudo de caso, o Data Warehouse Sistema Único de Saúde (SUS) brasileiro.

1.3. ESTRUTURA DA MONOGRAFIA

Para atingir os objetivos propostos, o restante deste trabalho está organizado em temas da seguinte maneira:

- **Capítulo 2 – Conceitos Básicos:** serão abordados conceitos introdutórios, mas importantes para o entendimento deste trabalho, através de descrições conceituais, seus princípios e objetivos. Leitores familiarizados com SIG, DW e sistemas OLAP podem prosseguir para o capítulo 3;
- **Capítulo 3 – Trabalhos Relacionados:** contextualiza este trabalho, apresentando pesquisas relacionadas a este e principalmente o projeto GOLAPA, contexto dentro do qual este projeto está inserido;
- **Capítulo 4 – Tecnologias envolvidas:** apresenta um maior detalhamento sobre as tecnologias utilizadas para a realização deste trabalho. O servidor OLAP Mondrian, a linguagem de consulta MDX e um dos temas mais importantes deste trabalho, o suporte fornecido pelo Mondrian a funções definidas pelo usuário (Mondrian User-defined Functions);
- **Capítulo 5 – Extensão do Servidor Mondrian com UDF Espacial:** apresenta detalhes do desenvolvimento e implementação dos operadores espaciais através da utilização das funções definidas pelo usuário (Mondrian User-defined Functions);
- **Capítulo 6 – Estudo de Caso:** apresenta uma aplicação prática deste trabalho; e
- **Capítulo 7 – Conclusão:** por fim, são apresentadas algumas conclusões, contribuições alcançadas e recomendações para trabalhos futuros.

2. Conceitos Básicos

2.1. *Sistemas de Informações Geográficas*

Sistemas de informações geográficas (SIG) são os sistemas computacionais que permitem a captura, o armazenamento, a manipulação, a recuperação, a análise e a apresentação de dados referenciados geograficamente, ou seja, dados que representam objetos ou fenômenos cuja localização geográfica está associada a uma posição na superfície terrestre [10].

Os SIG são ferramentas importantes para áreas que necessitam manipular e analisar informações baseadas em sua localização no espaço [6].

Para a realização de análises e consultas em SIG, geralmente são apresentados mapas que mostram visualmente a disposição, na superfície terrestre, de dados armazenados no banco de dados geográficos.

Vários autores sugerem uma arquitetura em camadas para SIG, de forma que cada camada seja especializada em determinada tarefa ou determinado tipo de dados. Casanova et al. apresentam, em [11], uma arquitetura básica para um SIG. A estrutura apresentada na Figura 2.1 representa o relacionamento existente entre estes componentes.

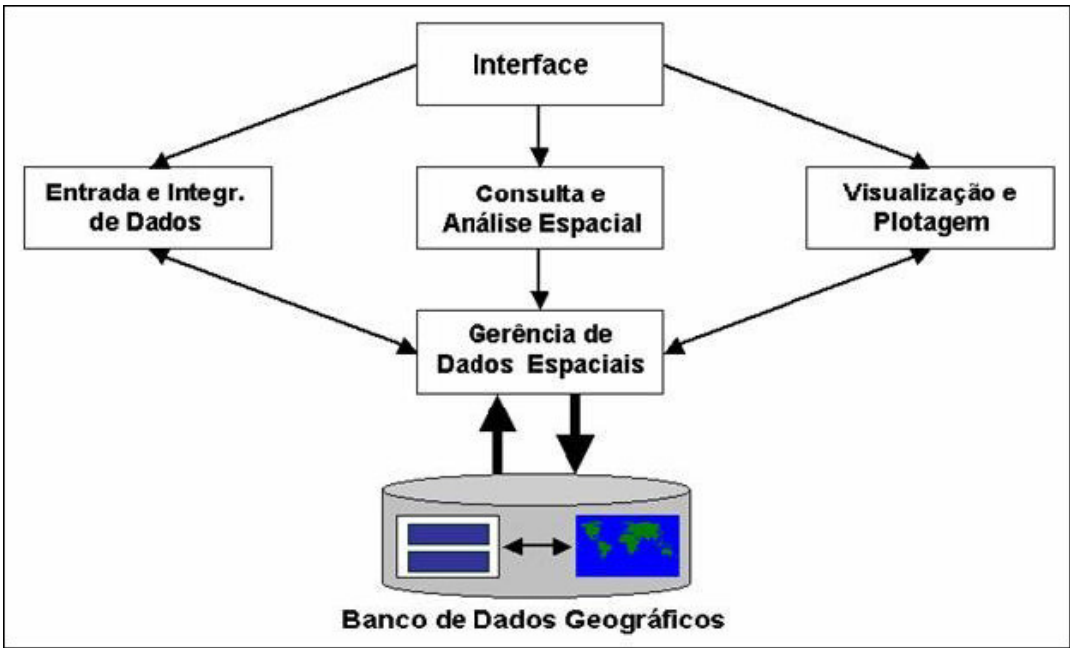


Figura 2.1 Componentes de um Sistema de Informações Geográficas [11]

Em um nível mais externo, se encontra a interface com o usuário, a qual disponibiliza a forma como o sistema é manipulado. No nível intermediário, se encontram os módulos responsáveis pelo processamento dos dados espaciais,

análise e visualização. Em um nível mais interno, se encontra o sistema de gerenciamento de banco de dados geográficos, o qual deve prover o suporte para armazenamento e a recuperação dos dados espaciais.

No nível intermediário, para o processamento dos dados espaciais, existem operadores que, entre outras, possuem as seguintes funções [12]:

- Converter geometrias em formatos de dados externos como, por exemplo, GML (Geography markup language) [30];
- Recuperar propriedades das geometrias, tais como as coordenadas de pontos, comprimentos de linhas, áreas de polígonos, entre outros;
- Comparar os relacionamentos espaciais existentes entre duas geometrias. Isto inclui as operações para verificar se duas geometrias são iguais, se intersectam ou se tocam, entre inúmeras outras; e
- Gerar novas geometrias, como por exemplo, a união de duas geometrias existentes.

As duas últimas categorias citadas são as mais relevantes para este trabalho, visto que são as mais utilizadas em consultas de suporte à decisão. As duas seções a seguir, apresentam uma lista de operadores para estas categorias.

2.1.1. Operadores para Comparar Relacionamentos Espaciais

Esta lista classifica os operadores espaciais em topológicos, métricos e posicionais, descritas a seguir.

Operações Topológicas

Este tipo abrange as operações que testam os relacionamentos topológicos entre as geometrias, são estas [31]:

- Toca – Verifica se uma geometria toca outra em algum ponto de sua superfície;
- Cruza – Verifica se uma geometria cruza uma outra geometria;
- Sobrepõe – Verifica se uma geometria sobrepõe outra geometria; e
- Dentro de – Verifica se uma geometria está dentro de outra geometria.

Operações Métricas

Este tipo abrange as operações que fazem medições e retornam um resultado escalar como, por exemplo:

- Distância – Calcula a menor distância entre duas geometrias;
- Área – Calcula a área de uma geometria;
- Comprimento – Calcula o comprimento de uma geometria; e
- Perímetro – Calcula o perímetro de uma geometria.

Operações Posicionais

Essa categoria engloba as operações de posicionamento. Este grupo inclui os operadores de direção, são estes [32]: Ao Norte De, Ao Sul De, Ao Leste De, Ao Oeste De, Ao Nordeste De, Ao Noroeste De, Ao Sudeste De e Ao Sudoeste De.

2.1.2. Operações que Geram Novas Geometrias

Esta categoria de operações diz respeito às operações, que quando executadas, geram como resultados novas geometrias. Por exemplo:

- Zona de Influência – Retorna uma geometria que representa todos os pontos que estão localizados a determinada distância de um ponto ou uma geometria;
- Intersecção – Retorna uma geometria que representa a área de interseção de duas geometrias;
- União – Retorna uma geometria que representa a união entre duas geometrias; e
- Diferença – Retorna uma geometria que representa a diferença entre duas geometrias.

2.2. *DATA WAREHOUSE*

Proposto por Inmon, [13], o termo Data Warehouse (DW) representa uma coleção de dados utilizada para suporte a decisões gerenciais e possui as seguintes características:

- É orientado a assunto - armazena dados importantes sobre temas específicos do negócio da organização;
- Perfeitamente integrado - consolida dados de diferentes fontes, organizando-os para resolver as divergências freqüentemente

existentes no formato e na codificação dos dados, de forma que a estrutura final tenha uma representação única;

- Variante no tempo - permite o acompanhamento histórico dos dados ao possibilitar a análise da evolução do dado ao longo do tempo; e
- Não volátil - informações raramente são modificadas, dados devem apenas ser carregados e analisados. Esta característica, juntamente com o fato de armazenar dados históricos, faz com que o DW tenda a ser mais volumoso do que os bancos de dados operacionais.

Portanto, DW se projeta como uma tecnologia útil para o processo de tomada de decisão, uma vez que sua estrutura facilita a manipulação de informações e a extração de novas informações, na maioria das vezes relevantes e estratégicas. Essa tecnologia vem sendo utilizada positivamente em diversos setores: produção, varejo, serviços financeiros, entre outros [14, 15].

O povoamento dos dados de um DW geralmente compreende as seguintes etapas:

1. Extração – Coleta de dados nos sistemas existentes;

2. Transformação e limpeza – Uniformização e integração dos dados; e

3. Carga – Alimentação do DW com dados resultantes das etapas anteriores.

Este processo é conhecido pelo nome de ETL [16], do inglês, "Extract, Transform, and Load". Além disso, deve-se ainda incluir metadados para manter informações a respeito do que está armazenado no DW.

Após esse processo de criação do DW, os dados ficam disponíveis para consulta através de várias ferramentas, como servidores OLAP [16], data mining [39] e geradores de relatório.

O modelo de dados utilizado em DW é o modelo dimensional, onde os dados geralmente são representados utilizando o esquema estrela [16]. Tal esquema é estruturado ao redor de uma tabela principal, denominada fato, que se liga às tabelas secundárias, denominadas dimensões, como mostra a Figura 2.2.

A tabela fato armazena as medidas numéricas do negócio, como por exemplo, número de unidades produzidas e número de unidades vendidas. Cada fato representa um item, uma transação ou um evento do negócio.

As dimensões armazenam descrições secundárias do negócio, como por exemplo, uma dimensão produto (que possui como atributos marca, categoria, embalagem), que auxiliam nas análises.

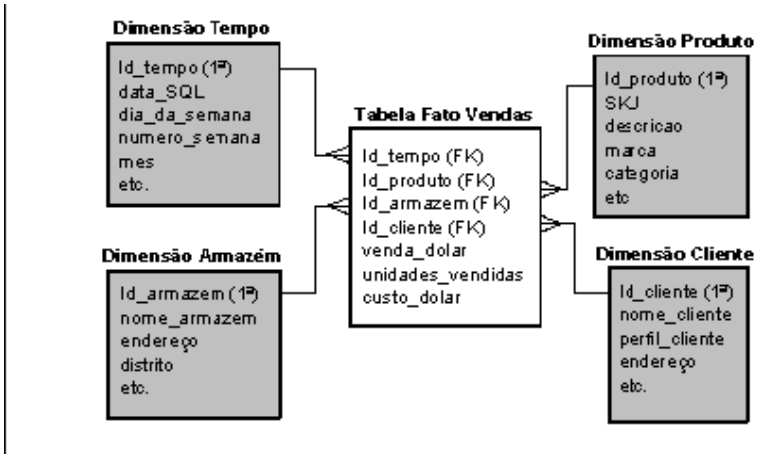


Figura 2.2 Modelo do esquema estrela [16]

O esquema estrela tradicional normalmente é utilizado na maioria dos projetos de bancos de dados analíticos. No entanto, em certas situações seu uso deve ser abdicado, como é o caso de quando uma tabela dimensional possui uma quantidade muito grande de atributos. Neste caso podem ser considerados outros tipos de estruturas como o esquema floco de neve [16].

2.3. *DATA WAREHOUSE GEOGRÁFICO*

Um Data Warehouse Geográfico (DWG) é um complemento da abordagem tradicional de Data Warehouse, onde este é estendido para que possa manipular dados espaciais. Basicamente, isto significa estender o modelo estrela através da inserção de propriedades geográficas.

É importante ressaltar que um DWG deve manter as características tradicionais de um DW [13], ou seja, ser orientado ao assunto, integrado, não-volátil e variante no tempo.

Um DWG pode ser especificado por três diretivas [17], são elas:

- A primeira diz respeito à necessidade de uma ferramenta que execute as agregações espaciais, conversões geográficas e geo-referenciamento;
- A segunda referindo-se à modelagem do banco de dados de apoio ao DWG que deve ser feita seguindo uma combinação entre as características de um banco de dados espacial e de um banco de dados voltado para DW; e
- Por último, o enfoque sobre a análise estatístico-espacial dos dados e a apresentação alternativa dos dados em mapas ou tabelas.

O objetivo mais importante desses enfoques é sempre permitir que sejam feitas: (1) consultas ao DWG a partir dos atributos geográficos, (2) análise espacial

destes resultados, (3) refinamentos sucessivos e (4) agregações dos resultados que podem ser visualizadas instantaneamente por áreas geográficas.

A partir de um DWG, as ferramentas OLAP devem ser capazes de analisar os dados geográficos produzidos por um SIG, enquanto este, por sua vez, deve ser capaz de realizar análises espaciais sobre os dados multidimensionais manipulados pelas ferramentas OLAP [18].

2.4. *ON-LINE ANALYTICAL PROCESSING*

O termo OLAP (On-Line Analytical Processing) [19] diz respeito a um conjunto de tecnologias que tem como objetivo dar suporte ao processo decisório através de consultas, análises e cálculos, estejam estes dados armazenados em DW ou não.

Sistemas OLAP permitem que seus usuários (analistas, gerentes, executivos, diretores, entre outros) obtenham conhecimento e perspicácia nas consultas e análises dos dados. É importante ressaltar que as análises de dados em sistemas OLAP devem ser rápidas, interativas e consistentes para que apóiem eficientemente a tomada de decisões em níveis gerenciais [16, 20]. Algumas ferramentas OLAP também possuem recursos para fazer previsões, análises de tendências, simulações, entre outras [21].

O componente principal de sistemas OLAP é o servidor OLAP, que fica situado entre um cliente e o sistema de gerenciamento de banco de dados (SGBD) do DW. O Servidor OLAP entende como os dados estão organizados no banco de dados e possui funções especiais para analise dos dados. Os servidores OLAP geralmente são categorizados de acordo com a maneira de como guardam seus dados e podem ser:

- MOLAP (Multidimensional OLAP) – Quando este armazena seus dados no disco em estruturas otimizadas para acesso multidimensional. Geralmente essas estruturas são formadas por arrays multidimensionais;
- ROLAP (Relational OLAP) – Quando o servidor armazena seus dados em bancos de dados relacionais; e
- HOLAP (Hybrid OLAP) – Servidores MOLAP armazenam dados dos fatos em formato multidimensional, mas se existirem mais do que algumas dimensões, esses dados ficaram esparsos, e o servidor não terá um bom desempenho. Um Servidor HOLAP resolve este problema deixando os dados mais granulares armazenados em um banco de dados relacional, enquanto os agregados no formato multidimensional.

2.5. *SPATIAL ON-LINE ANALYTICAL PROCESSING*

Spatial On-Line Analytical Processing (Spatial OLAP ou SOLAP) pode ser definido como uma plataforma visual, construída especialmente para dar suporte rápido e fácil à análise espaço-temporal e exploração de dados seguindo uma abordagem multidimensional, composta de visualizadores cartográficos, bem como de visualizadores em forma de tabelas e diagramas [29].

Uma tecnologia SOLAP ideal deve tratar objetos espaciais como parte integrada da navegação e oferecer alternativas para manipular eficientemente estes objetos.

No próximo capítulo analisamos o engenho SOLAP no qual este trabalho está inserido, o GOLAPA, além de outras propostas e trabalhos relacionados.

3. Trabalhos Relacionados

3.1. GOLAPA (GEOGRAPHICAL ON-LINE ANALYTICAL PROCESSING ARCHITECTURE)

GOLAPA [8, 18, 25] é uma arquitetura de software que pretende ser baseada em tecnologias abertas e extensíveis, e otimizada para suporte à decisão em um contexto multidimensional-geográfico.

Para o primeiro objetivo, Java e XML foram utilizadas como tecnologias de implementação. Para o segundo, uma arquitetura baseada na utilização de um DWG foi utilizada. A primeira escolha justifica-se pela ampla abrangência de ambas as tecnologias e a segunda por ser a alternativa que mais se adéqua para aplicações de suporte à decisão (alto desempenho e baixa freqüência de atualização de dados).

As camadas da arquitetura GOLAPA são distribuídas conforme a Figura 3.1. Ela dispõe de 3 camadas essenciais para o suporte multidimensional-geográfico: as camadas I, II e III, que tratam os dados, serviços e interface gráfica de um sistema com capacidades de processamento tanto analítico quanto geográfico, respectivamente.

Além dessas, existem duas camadas de suporte, A e B, que tratam dos dados operativos e da construção do DWG respectivamente.

A organização em camadas desta arquitetura preza pelo máximo de coesão possível entre os componentes de cada camada, bem como o mínimo acoplamento entre elas, o que é essencial em termos de qualidade de software. Além disso, na organização em camadas, cada uma delas faz uso dos recursos da camada imediatamente inferior e provê serviços para a camada imediatamente superior [26].

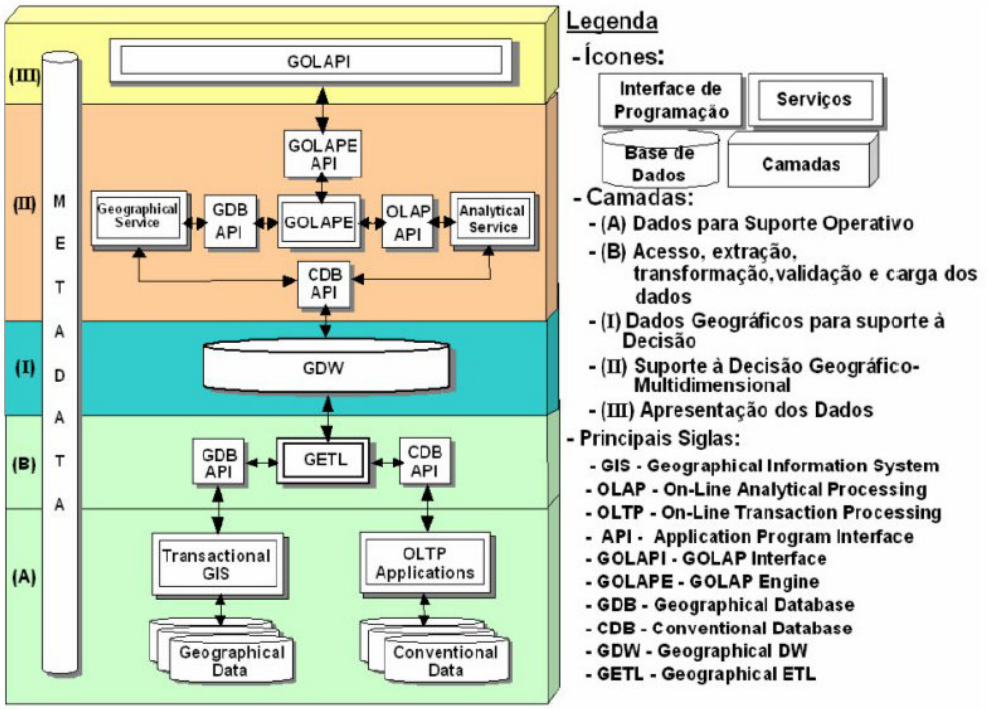


Figura 3.1 arquitetura GOLAPA [8]

3.2. *OUTRAS PROPOSTAS*

Outros trabalhos também visam à integração entre processamento multidimensional e geográfico.

O MapWarehouse [40] é um dos trabalhos que tem como objetivo essa integração e implementa o modelo lógico de um DWG sobre um SGBD objeto-relacional. Uma das limitações desse trabalho é a falta de uma linguagem de consulta específica para DWG, tornando as consultas mais complexas extremamente complicadas.

Outra abordagem é proposta em [41]. SOVAT possibilita a visualização dos resultados de consultas multidimensionais em mapas e gráficos, porém armazena as informações multidimensionais e geográficas em bases distintas, o que é uma desvantagem.

O MapCube [42] é um trabalho para processamento multidimensional-geográfico baseado no operador Cube. A principal diferença entre estes dois operadores é que enquanto o Cube apresenta os resultados em Tabelas, o MapCube os apresenta em tabelas e mapas.

Além destes, vários outros trabalhos [43, 44, 45] presentes na literatura objetivam esta integração, apresentam ferramentas para SOLAP, bem como conceitos relacionados a este tópico.

3.3. *GEOMDQL*

Vários trabalhos [33, 34, 35, 36] apresentam como soluções para consultas espaciais linguagens de consulta adicionais, sendo a maioria delas extensão da linguagem SQL padrão. A linguagem GeoMDQL (Geographic

Multidimensional Query Language) [6] também está inserida no contexto de consultas espaciais, mas, ao contrário das anteriores, é uma extensão da linguagem MDX [27], para prover suporte para o uso de operadores espaciais. Essa alternativa tenta aproximar a linguagem ainda mais do propósito analítico-dimensional, visto que a linguagem MDX é voltada para processamentos dessa natureza.

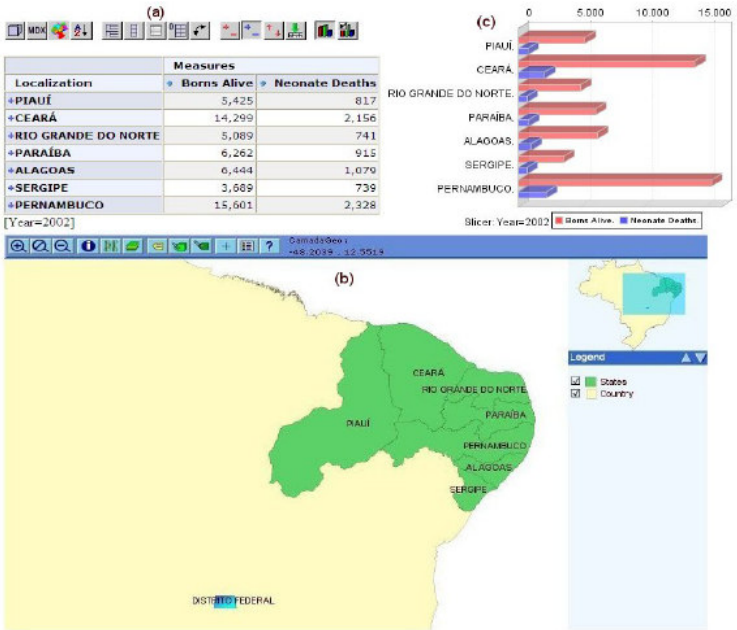
Portanto, como uma de suas principais contribuições, a linguagem GeoMDQL especifica e implementa uma linguagem geográfica multidimensional, que permite a realização de consultas envolvendo tanto operadores analítico-multidimensionais quanto operadores espaciais.

No Quadro 01 mostramos um exemplo da linguagem GeoMDQL. Ela inclui o operador espacial at_northeast_of e retorna os nascidos vivos e óbitos neonatais dos estados que ficam ao nordeste do distrito federal ocorridos no ano de 2000.

Quadro 01: Um exemplo de consulta GeoMDQL.

```
SELECT [Measures].[Nascidos Vivos],[Measures].[Obitos Neonatais]
ON COLUMNS, [Localizacao].[Estado].MEMBERS ON ROWS FROM [BRSaude] WHERE (([Time].[Year].[2002]) and (AT_NORTHEAST_OF([Localizacao].[Estado], [Localizacao].[Estado].[DISTRITO FEDERAL])))) ON MAP
```

Figura 3.2 Resultado da consulta GeoMDQL do Quadro 01.



4. Tecnologias Envolvidas

4.1. MONDRIAN

Mondrian [7] é um servidor OLAP Open Source, escrito em Java, que utiliza a linguagem de consulta MDX e que pode ser utilizado em conjunto com diversos SGBD no desenvolvimento de aplicações.

O Mondrian é projetado conforme a arquitetura ROLAP e estruturado em quatro camadas, que vão desde a interface com o usuário, até o repositório de informações [7, 9], a saber:

- Camada de apresentação - determina o que o usuário enxerga em seu monitor e como pode interagir para obter novas informações. Existem várias maneiras dos dados multidimensionais serem apresentados para os usuários, incluindo *PivotTables* (uma forma de tabela interativa), gráficos de linhas, barras ou fatias. No entanto, todas essas formas de apresentação têm em comum os conceitos multidimensionais de dimensões, medidas e células, sobre os quais esta camada recebe as requisições dos usuários e devolve seus resultados;
- Camada Dimensional - executa parsing, validação e execução das consultas MDX submetidas pela camada de apresentação. As consultas são tratadas em diferentes fases. Os eixos das dimensões são computados primeiramente, e então os valores das células de cada eixo. Metadados presentes nesta camada descrevem o modelo dimensional e como este se mapeia no modelo relacional;
- Camada Estrela - esta é uma camada que tem basicamente função de melhorar o desempenho do sistema, armazenando uma estrutura de cache que armazena na memória um conjunto de medidas associadas às respectivas dimensões. Quando os dados solicitados não estão disponíveis nessa estrutura ou não podem ser dela derivados, a requisição é repassada à camada de armazenamento; e
- Camada de armazenamento - consiste no sistema de gerenciamento de banco de dados relacional.

Todos estes componentes podem estar presentes na mesma máquina, ou serem distribuídos. As camadas 2 e 3 (Dimensional e Estrela) compõem o servidor do Mondrian e devem estar presentes na mesma máquina. A camada de armazenamento pode estar situada em outra máquina e ser acessada remotamente através de uma conexão JDBC [38].

O Mondrian também fornece uma API (Application Programming Interface) para aplicações clientes executarem consultas. Ela é semelhante a API JDBC, sendo a linguagem de consulta, MDX, a maior diferença.

Por ser escrito em Java, é independente de plataforma. Além disso, é disponibilizado como software livre, podendo ser estendido para se adaptar a diferentes requisitos, como por exemplo, no caso de sua extensão para processamento de consultas geográficas e geográfico-multidimensionais.

4.2. MDX

MDX [27] é uma linguagem de consulta para servidores OLAP.

A linguagem MDX provê uma sintaxe rica, poderosa e ao mesmo tempo simples para consultar e manipular dados armazenados nos servidores OLAP de forma bastante flexível. Ela possui uma vasta quantidade de operadores analíticos e utiliza expressões compostas de identificadores, valores e funções que são avaliados pelo servidor OLAP para obter os objetos (por exemplo, membros) ou escalares (por exemplo, um número).

MDX representa para os servidores OLAP o que a linguagem SQL representa para os SGBD Relacionais. No entanto ela não representa uma extensão da linguagem SQL e pode-se diferenciar as duas em vários pontos. Como exemplo, nos SGBD relacionais tabelas são utilizadas como fontes de dados, enquanto MDX utiliza-se do conceito de cubos.

MDX tornou-se um padrão de mercado para consultas multidimensionais, sendo bem aceita em aplicações analíticas. Ela foi adotada por uma grande variedade de companhias. Tanto companhias que trabalham com OLAP no lado servidor como Applix, Microstrategy, SAS, SAP, Whitelight, NCR quanto companhias que trabalham com OLAP no lado cliente, como Panorama Software, Proclarity, AppSource, Cognos, Business Objects, Brio Technology, Crystal Reports, Microsoft Excel, Microsoft Reporting Services, entre outros.

O Quadro 02 apresenta uma consulta MDX básica. A consulta retorna como resultado o conjunto que contém um balanço geral sobre as vendas de uma certa cadeia de lojas, durante os trimestres do ano de 1997.

Quadro 02: Um exemplo de consulta MDX.

```
SELECT Measures.MEMBERS ON COLUMNS,
{[Time].[1997].children} ON ROWS
FROM [Sales]
```

Esta consulta pode ser entendida da seguinte maneira:

- A clausula SELECT seleciona as medidas numéricas do negócio (membros da dimensão Medida), e o trimestres do ano de 1997 ([Year].[1997].children), membro da dimensão Ano (Year); e
- A cláusula FROM especifica que a fonte de dados é o cubo Vendas (Sales).

Seu resultado é mostrado na Figura 4.1, a seguir:

Figura 4.1 Resultado da consulta MDX do Quadro 02.

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Q1	66.291,00	55.752,24	R$ 139.628,35	21588	83.876,11
Q2	62.610,00	52.964,22	R$ 132.666,27	20368	79.702,05
Q3	65.848,00	55.904,87	R$ 140.271,89	21453	84.367,02
Q4	72.024,00	61.005,90	R$ 152.671,62	23428	91.665,72

4.3. MONDRIAN USER-DEFINED FUNCTIONS

O servidor OLAP Mondrian fornece facilidades de extensão. Extensões tanto do seu mecanismo de funcionamento, visto que é Open Source, quanto da linguagem de consulta MDX.

Estas extensões à linguagem de consulta MDX no Mondrian tomam a forma de Função na sintaxe MDX e são conhecidas como User-Defined Functions (Funções definidas pelo usuário).

Elas são implementadas utilizando-se código Java, devem possuir um construtor público e implementar a interface mondrian.spi.UserDefinedFunction .

A interface mondrian.spi.UserDefinedFunction possui 7 métodos, além do construtor, que são utilizados na definição de uma UDF, são estes:

- getName – Onde é definido o nome pelo qual a UDF será chamada em uma consulta MDX;
- getDescription – Método que define a descrição da UDF;
- getSyntax – Define a sintaxe da UDF para o *Parser* do Mondrian. Geralmente Syntax.Function, ou seja, tem a sintaxe de uma função f ( );
- getReturnType – Define o tipo de retorno da UDF;
- getParameterTypes – Define os tipos dos argumentos a serem recebidos pela UDF na consulta MDX;

- Execute – Essa função realiza o trabalho propriamente dito dentro da UDF, ela aplica as operações aos argumentos recebidos e retorna um resultado; e

- getReservedWords – Especifica uma lista de palavras reservadas usadas na UDF. Essa lista geralmente é vazia quando a função não possui palavras reservadas.

Destes métodos, 4 são mais importantes para este trabalho, visto que os outros, em sua maioria utilizam os valores default. São eles: (1) getName, (2) getReturnType, (3) getParameterTypes, (4) Execute. Os três últimos fazem utilização de tipos do mondrian (escritos em java) que são convertidos para tipos multidimensionais e vice-versa. Esses tipos são utilizados para que os parâmetros recebidos pela UDF (através da consulta MDX) sejam utilizados, manipulados e retornados, utilizando dentro da UDF código Java.

Os tipos nativos com suporte no servidor OLAP Mondrian incluem:

- BooleanType – representa expressões do tipo booleano;
- CubeType – representa um cubo ou um cubo virtual;
- NumericType – representa tipos numéricos;
- DimensionType – representa uma dimensão;
- LevelType – representa um nível;
- MemberType – representa um membro;
- SetType – representa conjuntos; e
- StringType – representa uma String.

Estes tipos, ao serem recebidos pela UDF, são avaliados e convertidos para tipos nativos de Java, como por exemplo, um NumericType, que é convertido para um double, ou um MemberType, que é convertido para um objeto do tipo List.

Apesar da grande quantidade de tipos nativos, novos tipos podem ser declarados no Mondrian através da implementação da interface Type [7].

Além da implementação da interface mondrian.spi.UserDefinedFunction, as UDF precisam ser registradas no arquivo XML que define o esquema do cubo de dados, como mostra o trecho de código XML do Quadro 03, que registra a função LowDistance e descreve o caminho de sua classe nos pacotes do Mondrian.

Quadro 03: Definição de uma consulta UDF no esquema do cubo de dados.

```
<Schema>
 …
  <UserDefinedFunction name="LowDistance"
className="mondrian.udf.LowDistance"/>
</Schema>
```

Após a implementação da função e do seu registro dentro do esquema XML, a UDF poderá ser utilizada em consultas MDX no servidor OLAP Mondrian.

O próximo capítulo apresenta as funções implementadas neste trabalho, mostra alguns exemplos destas sendo utilizadas em consultas, e por fim apresenta mais detalhes sobre a implementação das funções criadas.

5. Extensão do Servidor Mondrian com UDF Espacial

5.1. DEFINIÇÃO E SINTAXE DAS FUNÇÕES

5.1.1. LowDistance

Esta função ordena os membros de um conjunto em ordem crescente pela distância a um determinado membro deste conjunto. O primeiro argumento deve ser o "Unique Name" de um membro do conjunto. O segundo argumento é o conjunto de membros que será ordenado. Caso dois ou mais membros retornados tenham a mesma distância em relação ao membro passado como referencial na consulta, estes são apresentados no resultado em ordem alfabética.

- Sintaxe

  LowDistance(<MemberName>, <MemberSet>)

- Argumentos

  MemberName: Representa o "Unique Name" de um membro de um nível de uma dimensão geográfica. Exemplo: *[Localizacao].[Estado].[Distrito Federal]*

  MemberSet: Representa o "Unique Name" de um conjunto de uma dimensão geográfica. Exemplo: [*Localizacao].[Estado].members*

- Exemplo

O Quadro 04 mostra uma consulta que obtém a população dos estados, ordenados pela distância ao Distrito federal.

Quadro 04: Um exemplo de consulta LowDistance.

```
SELECT {[Measures].[Populacao]} on columns,
{LowDistance([Localizacao].[Estado].[Distrito Federal],
[Localizacao].[Estado].members)} on rows
FROM [BRSaude]
```

5.1.2. TopDistance

Esta função é bem parecida com a LowDistance, ela também ordena os membros de um conjunto em ordem crescente pela distância a um determinado membro deste conjunto, mas ela retorna apenas os N primeiros resultados, onde N é um argumento passado pelo usuário. O primeiro argumento deve ser o "Unique Name" de um membro do conjunto. O segundo argumento é o conjunto de membros que será ordenado. O terceiro argumento representa quantidade de membros a serem retornados na consulta. Caso dois ou mais membros retornados tenham a mesma distância em relação ao membro passado como referencial na consulta, estes são apresentados no resultado em ordem alfabética.

- Sintaxe

  TopDistance(<MemberName> , <MemberSet>, <Numeric Expression>)

- Argumentos

  MemberName: Representa o "Unique Name" de um membro de um nível de uma dimensão geográfica.

  MemberSet: Representa o "Unique Name" de um conjunto de uma dimensão geográfica.

  Numeric Expression: Uma expressão numérica que representa a quantidade de membros a serem retornados na consulta.

- Exemplo

O Quadro 05 mostra uma consulta que obtém a população dos estados, ordenados pela distância ao Distrito federal, mas que mostra apenas os 5 primeiros estados.

Quadro 05: Um exemplo de consulta TopDistance.

```
SELECT {[Measures].[Populacao]} on columns,
{TopDistance([Localizacao].[Estado].[Distrito Federal],
[Localizacao].[Estado].members), 5} on rows
FROM [BRSaude]
```

5.1.3. Drillout

Esta função retorna, em ordem alfabética, os membros (do mesmo nível) que fazem divisas com um determinado membro. O único argumento da função Drillout deve ser o "Unique Name" de um membro.

- Sintaxe

  Drillout(<MemberName>)

- Argumentos

  MemberName: Representa o "Unique Name" de um membro de um nível de uma dimensão geográfica.

- Exemplo

O Quadro 06 mostra uma consulta que obtém a população dos municípios vizinhos a Recife.

Quadro 06: Um exemplo de consulta Drillout.

```
SELECT {[Measures].[Populacao]} on columns,
{Drillout([Localizacao].[Municipio].[Recife] } on rows
FROM [BRSaude]
```

5.1.4. Touches

Esta função retorna, em ordem alfabética, os membros que fazem divisas com um determinado membro.

- Sintaxe

  Touches(<MemberName>, <MemberSet>)

- Argumentos

  MemberName: Representa o "Unique Name" de um membro de um nível de uma dimensão geográfica.

  MemberSet: Representa o "Unique Name" de um conjunto de uma dimensão geográfica.

- Exemplo

O Quadro 07 mostra uma consulta que obtém a população dos municípios vizinhos a Recife.

Quadro 07: Um exemplo de consulta Touches.

```
SELECT {[Measures].[Populacao]} on columns,
{Drillout([Localizacao].[Municipio].[Recife] } on rows
FROM [BRSaude]
```

5.1.5. RankArea

Esta função retorna os membros de um conjunto ordenados pelo tamanho de sua área. O único argumento da função RankArea deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

  RankArea(<MemberSet>)

- Argumentos

  MemberSet: Representa o "Unique Name" de um conjunto de uma dimensão geográfica.

- Exemplo

O Quadro 08 mostra uma consulta que obtém a população dos estados no ano de 2002, ordenando o resultado pela área dos estados.

Quadro 08: Um exemplo de consulta RankArea.

```
SELECT {[Measures].[Populacao]} on columns,
RankArea([Localizacao].[Estado].Members) on rows
FROM [BRSaude]  WHERE [Tempo].[Todos].[2002]
```

5.1.6. MaxArea

Esta função retorna o membro de um conjunto que possuir a maior área. O único argumento da função MaxArea deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

  MaxArea(<MemberSet>)

- Argumentos

  MemberSet: Representa um conjunto de membros

- Exemplo

  O Quadro 09 mostra uma consulta que obtém a população do estado de maior área no ano de 2002.

Quadro 09: Um exemplo de consulta MaxArea.

```
SELECT {[Measures].[Populacao]} on columns,
DrilldownLevel(MaxArea([Localizacao].[Estado].Members)) on rows
FROM [BRSaude]
WHERE [Tempo].[Todos].[2002]
```

5.1.7. MinArea

Esta função retorna o membro de um conjunto que possuir a menor área. O único argumento da função MinArea deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

  MinArea(<MemberSet>)

- Argumentos

  MemberSet: Representa um conjunto de membros

- Exemplo

O Quadro 10 mostra uma consulta que obtém a quantidade de óbitos neonatais do estado de menor área no ano de 2002.

Quadro 10: Um exemplo de consulta MinArea.

```
SELECT {[Measures].[Obitos Neonatais]} on columns,
{MinArea([Localizacao].[Regiao].members)} on rows
FROM [BRSaude]
```

5.1.8. RankPerimeter

Esta função retorna os membros de um conjunto ordenados pelo tamanho de seu perímetro. O único argumento da função RankPerimeter deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

RankPerimeter (<MemberSet>)

- Argumentos

MemberSet: Representa o "Unique Name" de um conjunto de uma dimensão geográfica. Exemplo: [Localizacao].[Estado].members

- Exemplo

O Quadro 11 mostra uma consulta que obtém a quantidade de óbitos neonatais dos estados, ordenando o resultado de acordo com o perímetro dos estados.

Quadro 11: Um exemplo de consulta RankPerimeter.

```
SELECT {[Measures].[Obitos Neonatais]} on columns, {RankPerimeter
([Localizacao].[Estado].members)} on rows
FROM [BRSaude]
```

5.1.9. MaxPerimeter

Esta função retorna o membro de um conjunto que possuir o maior perímetro. O único argumento da função MaxPerimeter deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

  MaxPerimeter(<MemberSet>)

- Argumentos

  MemberSet: Representa um conjunto de membros

- Exemplo

O Quadro 12 mostra uma consulta que obtém a quantidade de óbitos neonatais da zona climática de maior perímetro.

Quadro 12: Um exemplo de consulta MaxPerimeter.

```
SELECT {[Measures].[Obitos Neonatais]} on columns,
{MaxPerimeter([Clima].[Zona].members)} on rows
FROM [BRSaude]
```

5.1.10. MinPerimeter

Esta função retorna o membro de um conjunto que possuir o menor perímetro. O único argumento da função MinPerimeter deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

  MinPerimeter (<MemberSet>)

- Argumentos

MemberSet: Representa um conjunto de membros

- Exemplo

O Quadro 13 mostra uma consulta que obtém a quantidade de nascidos vivos na região de menor perímetro.

Quadro 13: Um exemplo de consulta MinPerimeter.

```
SELECT {[Measures].[Nascidos Vivos]} on columns,
{MinPerimeter([Localizacao].[Regiao].members)} on rows
FROM [BRSaude]
```

5.1.11. RankLength

Esta função retorna os membros de um conjunto ordenados pelo tamanho de seu comprimento. O único argumento da função RankLength deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

  RankLength (<MemberSet>)

- Argumentos

  MemberSet: Representa o "Unique Name" de um conjunto de uma dimensão geográfica. Exemplo: [Localizacao].[Estado].members

5.1.12. MaxLength

Esta função retorna o membro de um conjunto que possuir o maior comprimento. O único argumento da função MaxLength deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

  MaxLength (<MemberSet>)

- Argumentos

MemberSet: Representa o "Unique Name" de um conjunto de uma dimensão geográfica. Exemplo: [Localizacao].[Municipio].members

5.1.13. MinLength

Esta função retorna o membro de um conjunto que possuir o menor comprimento entre todos os membros. O único argumento da função MinLength deve ser o "Unique Name" de um conjunto de membros.

- Sintaxe

  MinLength (<MemberSet>)

- Argumentos

  MemberSet: Representa o "Unique Name" de um conjunto de uma dimensão geográfica. Exemplo: [Localizacao].[Estado].members

5.2. ALGORITMO DE IMPLEMENTAÇÃO

Aqui apresentamos o algoritmo de implementação das funções citadas na seção anterior. As funções recebem como entrada objetos multidimensionais (por exemplo: membros e conjuntos), além de escalares (como inteiros, por exemplo) e gera na saída um resultado contendo objetos de um BD multidimensional a serem utilizados no restante da consulta MDX.

Conceitualmente, o algoritmo consiste das seguintes etapas:

1. A função UDF é disparada pelo *parser* do mondrian, recebendo como parâmetros os operadores multidimensionais envolvidos.

2. Os tipos dos parâmetros são validados.

3. Os parâmetros, após serem validados, são recuperados e encaminhados ao método responsável pelo seu tratamento dentro da UDF.

4. Com as informações obtidas por esses parâmetros multidimensionais, a UDF conecta-se ao banco de dados geográfico-relacional, de onde extrai as informações a serem retornadas pela UDF.

5. Essas informações são tratadas e retornadas para o restante da consulta MDX.

5.3. *DETALHES DE IMPLEMENTAÇÃO*

As implementações das funções citadas na seção anterior seguem o modelo de implementação de User Defined Functions definido na documentação do Mondrian e apresentados na seção 4.3. [7] Esse modelo padrão deve ser seguido para que o servidor Olap Mondrian possa reconhecer que aquelas funções são definidas pelo usuário e é uma forma de extensão do conjunto de funções nativas. Dessa forma, podemos utilizá-las nas consultas MDX.
Neste trabalho, porém, além das funcionalidades e facilidades com suporte no Mondrian, foram utilizados operadores espaciais do PostGIS para que as UDF tivessem um propósito multidimensional-geográfico.
O mecanismo de funcionamento das UDF implementadas neste trabalho pode ser visto na Figura 5.1 a seguir:

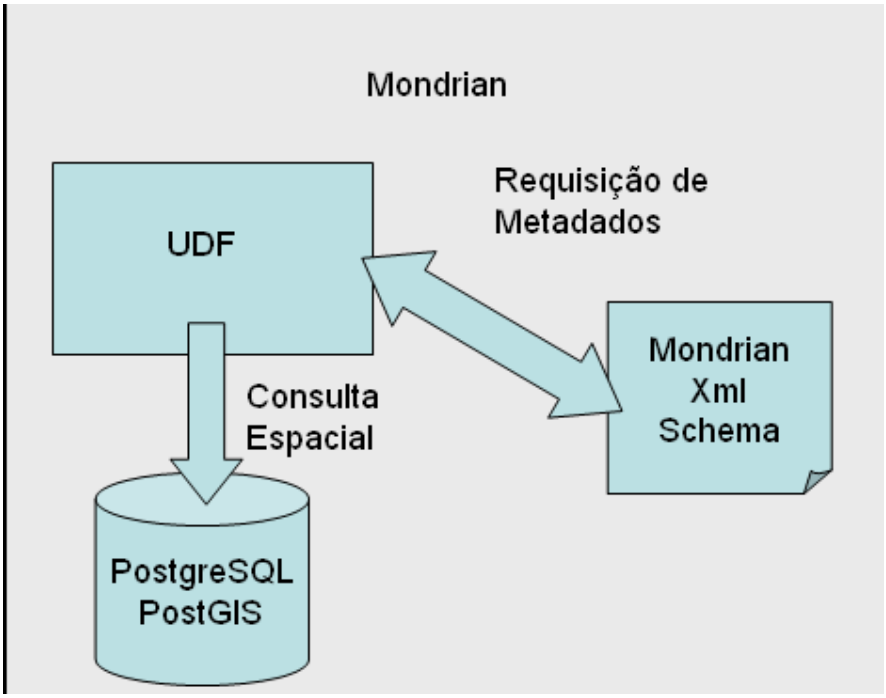


Figura 5.1 Mecanismo de funcionamento das UDF implementadas neste trabalho.

No primeiro momento, a UDF requisita metadados sobre as informações recebidas como seus argumentos. Estes metadados mapeiam os atributos multidimensionais recebidos em tabelas e atributos do banco de dados geográfico.

Após receber estas informações, a UDF monta uma consulta Geográfica na linguagem SQL com operadores geográficos do PostGIS e a executa, através de uma conexão JDBC, extraindo do banco os dados necessários.

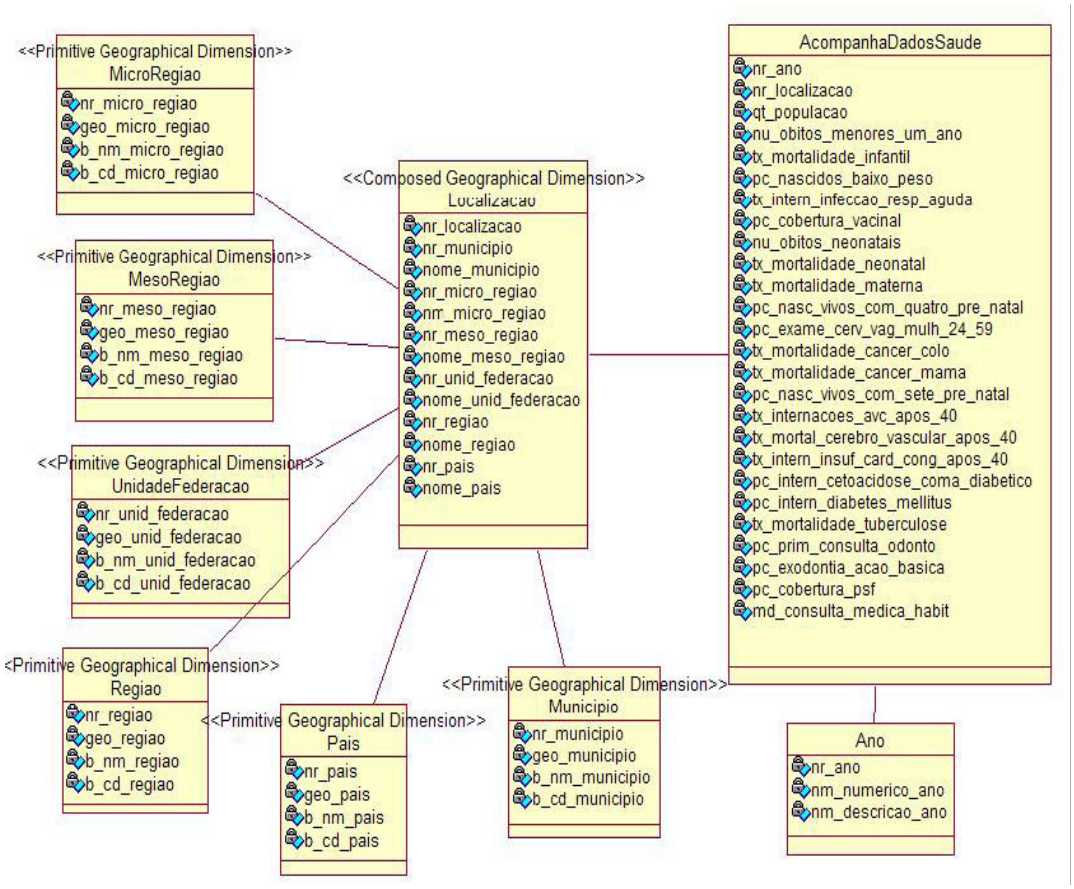
Após a execução da consulta, estes dados são manipulados, reorganizados no formato multidimensional e retornados para o Mondrian, que é o responsável pelo restante da consulta MDX.

Como também pode ser visto na arquitetura, as UDF deste trabalho utilizam como SGBD o PostgreSQL (com sua extensão geográfica PostGIS). No entanto a estrutura das UDF é independente do SGBD, necessitando apenas de uma conexão JDBC, ou seja, este modelo pode ser estendido para funcionar com uma grande quantidade de SGBD.

6. Estudo de Caso

6.1. *Data Warehouse do DataSus*

Como apresentação de resultados deste trabalho, utilizaremos como estudo de caso o Data Warehouse Geográfico com dados sobre o Sistema Único de Saúde (SUS) brasileiro. [28] Quanto à aplicação geográfica, os dados foram adquiridos do Instituto Brasileiro de Geografia e Estatística (IBGE). Estes dados dizem respeito a geometria do país, das regiões brasileiras, dos estados, das meso-regiões, das micro-regiões e dos municípios. A justificativa para a realização deste estudo de caso foi por este DW ser o mesmo utilizado no projeto GOLAPA, no qual este trabalho está inserido.



Os dados analíticos utilizados para a construção do DWG produzem informações sobre saúde da mulher, o controle de doenças e a saúde bucal, taxas de mortalidade infantil, óbitos neonatais, a taxa de mortalidade materna, o percentual de nascidos abaixo do peso, entre outros.

Consequentemente, este DWG pode ajudar autoridades nos processos de planejamento, administração e avaliação de políticas públicas de saúde, bem como pode ser um meio eficiente de elaborar metas para melhorar os serviços de saúde pública disponíveis para a população brasileira, uma vez que permite a análise tanto analítica como geográfica com a visualização dos dados por meio de tabelas, gráficos e mapas.

6.2. *EXEMPLOS DE CONSULTAS*

Esta seção apresenta algumas consultas em execução sobre o DW apresentado na seção anterior.

Função Utilizada: LowDistance
Descrição: Exibir a população dos estados brasileiros, ordenando estes em relação a menor distância em relação a Pernambuco.
SELECT {[Measures].[Populacao]} on columns, {LowDistance([Localizacao].[Estado].[PERNAMBUCO], [Localizacao].[Estado].members)} on rows FROM [BRSaude]

	[Measures].[Populacao]
[Localizacao].[All].[Brasil].[Nordeste].[BAHIA]	93,354,533
[Localizacao].[All].[Brasil].[Nordeste].[PERNAMBUCO]	56,406,406
[Localizacao].[All].[Brasil].[Nordeste].[PIAUÍ]	20,225,738
[Localizacao].[All].[Brasil].[Nordeste].[CEARÁ]	53,457,291
[Localizacao].[All].[Brasil].[Nordeste].[PARAÍBA]	24,439,547
[Localizacao].[All].[Brasil].[Nordeste].[ALAGOAS]	20,161,181
[Localizacao].[All].[Brasil].[Nordeste].[SERGIPE]	12,906,060
[Localizacao].[All].[Brasil].[Nordeste].[RIO GRANDE DO NORTE]	19,913,663
[Localizacao].[All].[Brasil].[Nordeste].[MARANHÃO]	40,524,335
[Localizacao].[All].[Brasil].[Norte].[TOCANTINS]	8,472,992
[Localizacao].[All].[Brasil].[Sudeste].[MINAS GERAIS]	128,211,275
[Localizacao].[All].[Brasil].[Centro-Oeste].[GOIÁS]	36,507,426
[Localizacao].[All].[Brasil].[Norte].[PARÁ]	45,115,701

Função Utilizada: MaxArea
Descrição: Obter a população de cada sub-região do estado brasileiro de maior área no ano de 2002.
select {[Measures].[Populacao]} ON COLUMNS, DrilldownLevel(MaxArea([Localizacao].[Estado].Members)) ON ROWS from [BRSaude] where [Tempo].[Todos].[2002]

[Tempo].[Todos].[2002]	[Measures].[Populacao]
[Localizacao].[All].[Brasil].[Norte].[AMAZONAS]	2,961,804
[Localizacao].[All].[Brasil].[Norte].[AMAZONAS].[CENTRO AMAZONENSE]	2,290,296
[Localizacao].[All].[Brasil].[Norte].[AMAZONAS].[NORTE AMAZONENSE]	121,646
[Localizacao].[All].[Brasil].[Norte].[AMAZONAS].[SUDOESTE AMAZONENSE]	312,808
[Localizacao].[All].[Brasil].[Norte].[AMAZONAS].[SUL AMAZONENSE]	237,054

Função Utilizada: Drillout
Descrição: Obter o número de nascidos vivos e de óbitos maternos nos vizinhos de Pernambuco com mais de 10 milhões de habitantes.
select {[Measures].[Nascidos Vivos], [Measures].[Obitos Maternos]} ON COLUMNS, Intersect({Drillout([Localizacao].[Estado].[PERNAMBUCO])}, {Filter([Localizacao].[Estado].Members, ([Measures].[Populacao] > 10000000))}) ON ROWS from [BRSaude]where [Tempo].[Todos].[2002]

[Tempo].[Todos].[2002]	[Measures].[Nascidos Vivos]	[Measures].[Obitos Maternos]
[Localizacao].[All].[Brasil].[Nordeste].[BAHIA]	23,380	15,308

Função Utilizada: Drillout
Descrição: Obter o número de nascidos vivos e de óbitos maternos nos municipios ao redor da região metropolitana de Belo Horizonte.
select {[Measures].[Nascidos Vivos], [Measures].[Obitos Maternos]} ON COLUMNS, {Drillout([Localizacao].[All].[Brasil].[Sudeste].[MINAS GERAIS].[METROPOLITANA DE BELO HORIZONTE].[BELO HORIZONTE].[Belo Horizonte])} ON ROWS from [BRSaude] where [Tempo].[Todos].[2002]

[Tempo].[Todos].[2002]	[Measures].[Nascidos Vivos]	[Measures].[Obitos Maternos]
[Localizacao].[All].[Brasil].[Nordeste].[BAHIA].[SUL BAIANO].[ILHEUS-ITABUNA].[Santa Luzia]	534	
[Localizacao].[All].[Brasil].[Sudeste].[MINAS GERAIS].[METROPOLITANA DE BELO HORIZONTE].[BELO HORIZONTE].[Vespasiano]	166	100
[Localizacao].[All].[Brasil].[Sudeste].[MINAS GERAIS].[METROPOLITANA DE BELO HORIZONTE].[BELO HORIZONTE].[Brumadinho]	40	
[Localizacao].[All].[Brasil].[Sudeste].[MINAS GERAIS].[METROPOLITANA DE BELO HORIZONTE].[BELO HORIZONTE].[Contagem]	941	700
[Localizacao].[All].[Brasil].[Sudeste].[MINAS GERAIS].[METROPOLITANA DE BELO HORIZONTE].[BELO HORIZONTE].[Ibirité]	269	
[Localizacao].[All].[Brasil].[Sudeste].[MINAS GERAIS].[METROPOLITANA DE BELO HORIZONTE].[BELO HORIZONTE].[Nova Lima]	96	

7. Conclusão

7.1. *Considerações Finais*

Ao contextualizar as dificuldades encontradas na integração envolvendo sistemas de processamento de dados analíticos e sistemas geográficos e algumas soluções propostas para este problema (arquitetura GOLAPA e linguagem GeoMDQL), este trabalho se propôs a implementar algumas das funções definidas pela linguagem GeoMDQL utilizando-se de uma extensão do servidor OLAP Mondrian.

Nos três primeiros capítulos foram feitos levantamentos dos elementos envolvidos e que serviram de fundamentação teórica para o desenvolvimento deste trabalho.

Os capítulos seguintes apresentaram e detalharam o conceito de UDF (funções definidas pelo usuário), que forma o conceito central deste trabalho. Desde os princípios básicos, até a abordagem desenvolvida.

7.2. *Contribuições*

Os resultados deste trabalho mostram a viabilidade da implementação de algumas funções da linguagem GeoMDQL para estender o Mondrian com recursos geográficos através da utilização do recurso do servidor OLAP Mondrian chamado UDF.

Assim, o objetivo central, de "Estender o Servidor OLAP Mondrian com UDF Envolvendo Operadores Espaciais", pôde ser alcançado.

7.3. *Trabalhos Futuros*

A partir desta experiência notamos que várias outras consultas propostas em [6], inclusive extensões das implementadas neste trabalho, podem ser implementadas de forma bastante natural.

Além disso, recomendamos, como foi sugerido, a generalização do mecanismo aqui implementados para o PostgreSQL + PostGIS.

8. Bibliografia

[1] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F. e Pirahesh, H. (1997). "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data Mining and Knowledge Discovery", 1(1):29–53

[2] Han, J., Koperski, K. e Stefanovic, N. (1997). "GeoMiner: A System Prototype for Spatial data Mining." In ACM SIGMOD'97

[3] Kouba, Z., Matousek, K. e Miksovsk, P.(2000). "On data warehouse and gis integration. In DEXA '00: Proceedings of the 11th International Conference on Database and Expert Systems Applications", pages 604–613, London, UK

[4] Shekhar, S., Lu, C.T., Tan, X. e Chawla, S. (2000). "Map cube: A visualization tool for spatial data warehouse".

[5] Rivest, S., B´edard, Y., Proulx, M., Nadeau, M., Hubert, F., e Pastor, J.. (November 2005) "Solap technology: Merging business intelligence with geospatial technology for interactive spatio-temporal exploration and analysis of data." ISPRS Journal of Photogrammetry e Remote Sensing, pages 17–33.

[6] Silva, J. (2006). "GEOMDQL: Uma Linguagem de Consulta Geográfica E Multidimensional", PhD thesis Proposal. Universidade Federal de Pernambuco, Recife, PE, BR.

[7] Mondrian. Mondrian official home page, http://mondrian.sourceforge.net/, Maio 2008.

[8] Fidalgo, R. N. (2005). "Uma Infra-Estrutura para Integração de Modelos, Esquemas e Serviços Multidimensionais e Geográficos". PhD thesis, Centro de Informática - Universidade Federal de Pernambuco, Recife, PE, BR.

[9] Medeiros, V. N. (2006). "Concepção e Validação de Algoritmos para Processamento de Consultas Geográfico-Multidimensionais em um Ambiente de Data Warehouse Geográfico", Master thesis. Universidade Federal de Pernambuco, Recife, PE, BR.

[10] Worboys, M. e Duckham, M. (2004). "GIS: A Computing Perspective", 2nd Edition. CRC Press

[11] M. A. Casanova G. Câmara, A. S. Hemerly, G. C. Magalhães e C. M. B. Medeiros. (1996). "Anatomia de sistemas de informação geográfica, http://www.dpi.inpe.br/geopro/livros/anatomia.pdf"

[12] Stolze, K.. "Sql/mm spatial - the standard to manage spatial data in a relational database system".(2003). In 10th Conference on Database Systems for Business, Technology and Web

[13] W. H. Inmon. (2002). "Building the Data Warehouse", 3rd Edition. Wiley.

[14] An esri white paper. (March 1998). "Environmental Systems Research Institute Inc. Spatial data warehousing for hospital organizations".

[15] "IBM Corporation. Metro Nashville maps its future with db2 spatial data warehouse." (2002).

[16] Kimball, R., Reeves L., Ross M., e Thornthwaite W. (1998). "The Data Warehouse Lifecycle Toolkit." John Wiley & Sons.

[17] Silva, E.C.S. e Campos, M.L.M.(1998). "Integração de sistemas de informação geográficas e ferramentas olap. In Congresso para Usuários de Geoprocessamento da América Latina."

[18] Silva, J., Fidalgo, R. N., Times, V. C e Barros, R. S. M. (2004). "Towards a web service for geographic and multidimensional processing. In VI Brazilian Symposium on GeoInformatics, pages 2–17"

[19] Thomsen, E. (1997). "OLAP solutions: building multidimensional information systems." John Wiley & Sons, Inc., New York, NY, USA,

[20] Bispo, C. A. F. (1998). "Uma análise da nova geração de Sistemas de Apoio à Decisão". Master Thesis, São Carlos, USP.

[21] Campos, M. L. M. (2000) "Data warehouse: Estratégias, arquitetura e ferramentas." Anais dos Tutoriais e Minicursos do XV Simpósio Brasileiro de Banco de Dados (SBBD).

[22] Güting, R. H. (1994). "An introduction to spatial database systems." The VLDB Journal, 3(4):357–399.

[23] PostgreSQL. Postgresql official home page, http://www.postgresql.org/, Agosto 2007.

[24] Open Geospatial Consortium. Open geospatial consortium open gis catalog services specification, http://www.opengis.org,

[25] Silva, J., Fidalgo, R. N., Times, V. C e Barros, R. S. M. (2005). "Providing geographic-multidimensional decision support over the web. In 7th Asia-Pacific Web Conference (APWeb), pages 477–488"

[26] Mendes, A. (2002). "Arquitetura de Software - Desenvolvimento Orientado para Arquitetura." Campus, 1 edition.

[27] Microsoft Corporation. MDX Overview. Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/enus/ olapdmad/agmdxbasics 90qg.asp, 2004.

[28] Silva J., Times V. C., Salgado A.C., Medeiros V.N., e Fidalgo R. N. (2006). ”An open source and web based framework for geographic and multidimensional processing.” In The 21st Annual ACM Symposium on Applied Computing, Track on Advances in Spatial and Image-based Information Systems (ACM SAC ASIIS’06), pages 63–67

[29] Rivest, S., B´edard, Y., Proulx, M., Nadeau. (2001). “Toward better support for spatial decision making: Defining the characteristics of spatial on-line analytical processing (solap).” Geomatica, 55(4):539–555.

[30] Open Geospatial Consortium. Ogc geography markup language (gml) 2.0, document number: 01-029.2001, http://www.opengis.netigml/ol-029/gml2.html, 2001.

[31] Egenhofer, M. e Herring, J. (1991). “Categorizing binary topological relationships between regions, lines and points in geographic databases”. Technical report, Department of Surveying Engineering, University of Maine.

[32] Pourabbas, E. e Rafanelli, M. (2002). “A pictorial query language for querying geographic databases using positional and OLAP operators.” SIGMOD Record, 31(2):22–27.

[33] Open Geospatial Consortium. (1997). "OpenGIS simple features specification for SQL". Technical report, Open Geospatial Consortium.

[34] Open Geospatial Consortium. (2001). "Filter encoding implementation specification." Technical report, Open Geospatial Consortium.

[35] Egenhofer, M. J. (1994). "Spatial sql: A query and presentation language". IEEE Transactions on Knowledge and Data Engineering, 6(1):86–95.

[36] Chen H., Wang F., Sha J., e Yang S. (2000). "Geosql: A spatial query language of objectoriented gis. In Proceedings of the 2nd International Workshop on Computer Science and Informationn Technologies".

[37] Câmara, G., Casanova, M. A., Hemerly A. S., Magalhães G. C., Medeiros C. M. B. (1996). “Anatomia de Sistemas de Informação Geográfica.” SBC X Escola de Computação.

[38] Sun. Jdbc technology official home page, http://java.sun.com/products/jdbc/, Novembro 2005.

[39] Fayyad, U., Shapiro, G., Smyth, P. “From data mining to knowledge Discovery: An overview.” (1996) In: Advances in Knowledge Discovery and Data Mining, AAAI Press / The MIT Press, MIT, Cambridge, Massachusetts, and London, England.

[40] Sampaio, M.C., Sousa A. G., e Baptista C.S. (2006). “Towards
a logical multidimensional model for spatial data warehousing and olap”. In DOLAP ’06:
Proceedings of the 9th ACM international workshop on Data warehousing and OLAP, pages 83–90, New York, NY, USA, ACM Press.

[41] Scotch M., e Parmanto B. (2005). “Sovat: Spatial olap visualization and analysis tool.” In HICSS ’05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS’05) - Track 6, page 142

[42] Shekhar, S., Lu, C.T., Tan, X., e Chawla S. (2000). “Map cube: A visualization tool for spatial data warehouse”.

[43] Han J., Koperski K., and Stefanovic N. (1997). “GeoMiner: A System Prototype for Spatial data Mining.” In ACM SIGMOD’97.

[44] Ferreira, A .C. “Um modelo para suporte à integração de analises multidimensionais e espaciais.” (2002). Master’s thesis, UFRJ, Rio de Janeiro - Brasil.

[45] Pedersen, T.B., e Tryfona, N. (2001). “Pre-aggregation in spatial data warehouses”. In SSTD ’01: Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases, pages 460–480, London, UK,. Springer-Verlag.

Data e Assinaturas

Recife, 17 de junho de 2008

_____

Robson do Nascimento Fidalgo (Orientador)

_____

Joel da Silva (Co-Orientador)

_____

Fábio Rocha de Pinho (Aluno)