Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Informática

Um *Framework* para *Data Warehouse* Espacial

Felipe Menezes Cardoso

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Sistemas de Informação e Banco de Dados

Marcus Costa Sampaio e Cláudio de Souza Baptista
(Orientadores)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

C268f Cardoso, Felipe Menezes

Um framework para data warehouse espacial / Felipe Menezes Cardoso.─ Campina Grande, 2009.

124 f..: il. color.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

Referências.

Orientadores: Prof. Dr. Marcus Costa Sampaio.
Prof. Dr. Cláudio de Souza Baptista.

1. Banco de Dados 2. Data Warehouse 3. OLAP Espacial I. Título.

CDU 004.65(043)

"UM FRAMEWORK PARA DATA WAREHOUSE ESPACIAL"

FELIPE MENEZES CARDOSO

DISSERTAÇÃO APROVADA EM 24.08.2009

CLÁUDIO DE SOUZA BAPTISTA, PH.D
Orientador(a)

ULRICH SCHIEL, DR.
Examinador(a)

ROBSON DO NASCIMENTO FIDALGO, DR.
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Data Warehouses (DW) e ferramentas OLAP (Online Analytical Processing) são duas tecnologias amplamente utilizadas para auxiliar gestores no processo de tomada de decisões estratégicas. Pesquisas recentes buscaram a integração de dados espaciais ao ambiente de Data Warehouse, o que fez surgir o termo Data Warehouse Espacial. Em sua maioria, os trabalhos na área de DW Espacial se concentram na definição de novos modelos de dados multidimensionais e na implementação de novas ferramentas OLAP. A necessidade de novas ferramentas OLAP é justificável, visto que ferramentas OLAP tradicionais não são preparadas para lidarem com dados espaciais, sendo necessária uma nova categoria de ferramenta OLAP com características especiais. Esta nova categoria, referida pelo termo Spatial OLAP (SOLAP), visa integrar funcionalidades das ferramentas OLAP com funcionalidades tradicionalmente encontradas nos Sistemas de Informações Geográficas (SIG). Apesar do grande número de ferramentas SOLAP encontradas na literatura, acreditamos que a integração OLAP-SIG não é totalmente satisfatória pelo seguinte motivo: não existe ferramenta SOLAP que combine a utilização de operadores multidimensionais e espaciais, e seja destinada a usuários finais, isto é, destinada a usuários gestores - os principais interessados em tais ferramentas. Por isso, este trabalho propõe uma nova ferramenta SOLAP, chamada Mapwarehouse. O Mapwarehouse permite que usuários finais definam seus esquemas de Data Warehouse Espacial e realizem consultas aos Data Warehouses Espaciais criados a partir destes esquemas. Para isso, a ferramenta dispõe de linguagens gráficas para definição de esquemas e consultas, implementadas na forma de um conjunto de interfaces gráficas de alto nível de abstração. A ferramenta proposta segue a filosofia de framework, o que permite que programadores possam estendê-la e utilizá-la em tecnologias distintas de Sistemas Gerenciadores de Banco de Dados (SGBD). Visando sua validação, o framework foi estendido para os SGBDs Oracle 10g e PostgreSQL. Além disso, foram desenvolvidos dois estudos de caso distintos: um destinado à análise da distribuição agrícola do estado da Paraíba e outro destinado à análise dos atendimentos realizados pelo SAMU (Serviço de Atendimento Móvel de Urgência) do município de Aracaju-Sergipe.

Abstract

Data Warehouse (DW) and OLAP tools (Online Analytical Processing) are two technologies widely used to help managers in decision support process. Recent researches attempted to integrate spatial data into Data Warehouse environment. Most of the work related to spatial DW focus on the definition of new data models and implementation of new OLAP tools. The need for new OLAP tools is justifiable, since traditional OLAP tools are not prepared to deal with spatial data, so a new category of OLAP tool with special features comes up. This new category is referred by the term Spatial OLAP (SOLAP). Spatial OLAP aims to integrate features of OLAP tools with features traditionally found in the Geographic Information Systems (GIS). Despite the large number of SOLAP tools found in literature, we believe that OLAP-GIS integration is not fully satisfactory for the following reason: there is no tool that combines the use of spatial and multidimensional operators and is intended to end users. Therefore, this work proposes a new SOLAP tool called Mapwarehouse. Mapwarehouse lets end-users define their own Spatial Data Warehouse schemas and query Spatial Data Warehouses created from these schemes. For this, the tool has a graphical language for defining schemas and queries, implemented as a set of graphical interfaces on a high level of abstraction.

The proposed framework follows the framework philosophy, which allows programmers to extend it and use it in different technologies of Database Managment Systemas (DBMS). Aiming to validate it, the framework was extended to Oracle 10g and PostgreSQL. Furthermore, we developed two case studies: one for the analysis of the agricultural distribution of state of Paraíba and another for the analysis of the care provided by the SAMU at Aracaju city, Sergipe.

Agradecimentos

Este trabalho só foi possível graças ao apoio direto e indireto de outras pessoas e entidades. Por isso, gostaria de agradecer:

A Deus por ter me dado a saúde necessária para vencer os obstáculos da vida.

Aos meus pais, pelo carinho, amor, confiança e uma vida inteira de trabalho para me dar a educação necessária para que eu chegasse até aqui.

Aos meus orientadores, Professores Cláudio Baptista e Marcus Sampaio, pela paciência e ensinamentos transmitidos ao longo da orientação.

A minha namorada Laira, por todo o amor.

Aos meus grandes amigos e colegas de mestrado Gilson, Fernando e Augusto, pela momentos alegres e convivência diária no apartamento.

Aos novos amigos: Danillo, Yuri, Hugo, Welmisson, Halley, Marquinhos, Saulo, Guiga, Preto, Barata, pela amizade e momentos de alegria tanto na universidade como fora dela.

Ao LSI, pela infraestrutura disponibilizada.

Aos demais professores e funcionários da COPIN, em especial, à Aninha, pela gentileza e presteza.

Por fim, a Capes, pelo apoio financeiro.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A grande competitividade do mercado atual impõe às grandes corporações o uso de ferramentas que agilizem seus processos de negócio. Neste cenário, sistemas de informação (SI) são cada vez mais utilizados no dia-a-dia dessas corporações, promovendo melhorias nos produtos e serviços ofertados a partir de um controle operacional mais eficiente. Estes sistemas de informação são conhecidos pelo termo OLTP (*On-line Transaction Processing*) - Processamento de Transações *On-line* [1]. Se por um lado os sistemas OLTP auxiliam o controle operacional, por outro se tornam ineficientes do ponto de vista gerencial. Isto ocorre devido a uma característica específica de tais sistemas: lidam com grandes volumes de dados, em formatos heterogêneos e na forma normalizada, o que torna inviável - ou até mesmo impossível - a extração de informações para auxiliar os gestores no processo de tomada de decisões estratégicas.

O processo de tomada de decisões estratégicas é uma tarefa essencial para que as corporações cheguem a um lugar de destaque no mercado, visto que não basta apenas ter um controle operacional eficiente, é necessário também analisar comportamentos e prever tendências mercadológicas. Neste contexto, os Sistemas de Apoio à Decisão (SAD) [2] têm um papel de extrema importância, pois provêm mecanismos rápidos e eficientes que auxiliam os gestores no processo de tomada de decisão.

Uma tecnologia comumente utilizada na implantação de SADs é o *Data Warehouse* (DW) [3]. O DW está inserido em um cenário mais amplo, conhecido como *Data Warehousing* (armazenagem de dados), que será descrito na próxima seção.

1.1 *Data Warehousing*

Em um ambiente de *Data Warehousing*, como o ilustrado na Figura 1.1, os dados oriundos dos sistemas OLTP, juntamente com outras fontes de dados externas, servem com fontes de dados para o *Data Warehouse* (DW). Estes sistemas, neste contexto chamados de sistemas fonte, armazenam dados que serão integrados ao DW pelo processo de ETL (*Extract, Transform, Load*) [4]. O processo ETL tem como objetivo extrair as informações dos sistemas fontes, transformá-las (padronizando informações de bases de dados diversas) e por fim, carregá-las no DW, integrando-as com informações históricas já existentes.

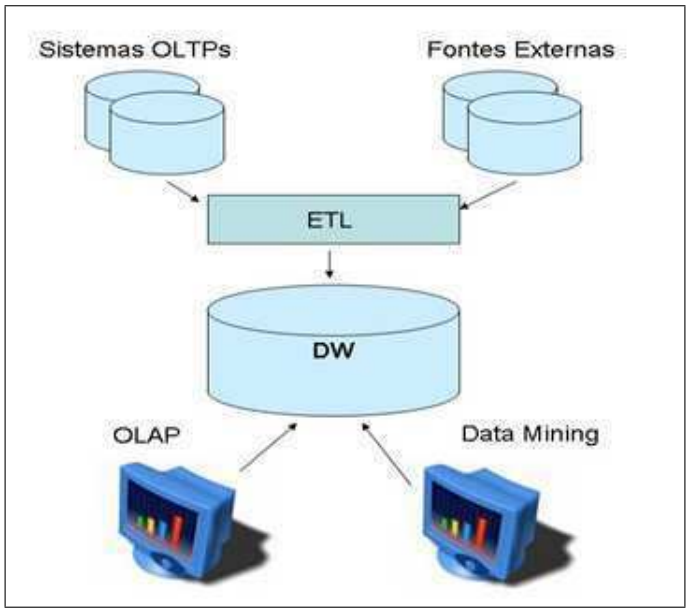

Figura 1.1: Ambiente de Data Warehousing

A partir dos dados armazenados no DW, os gestores podem realizar o processamento analítico das informações utilizando ferramentas especiais, denominadas ferramentas OLAP (*On-line Analytical Processing*) [5].

Nas subseções seguintes serão detalhados os componentes DW e OLAP, ilustrados na Figura 1.1.

1.1.1 *Data Warehouse*

O *Data Warehouse*, elemento central em um ambiente de *Data Warehousing*, é um banco de dados com características diferenciadas, especialmente projetado para armazenar informa-

ções de vários sistemas fontes de um ambiente corporativo. Segundo Inmon [6], um DW "é uma coleção de dados orientada por temas, integrado, variante no tempo e não-volátil, que dá apoio às decisões da administração". Cada uma dessas características serão descritas a seguir:

- **Orientado por tema:** dizer que um DW é orientado por temas significa que ele lida com informações estreitamente relacionadas ao negócio da empresa. Considerando uma empresa atacadista, por exemplo, pode-se ter um DW de tema **Vendas**. Este DW armazenaria várias informações sobre as vendas da empresa, permitindo aos gestores analisar estas informações e descobrir tendências e comportamentos relacionados ao negócio.

- **Integrado:** como elucidado anteriormente, o DW armazena informações oriundas de diversos sistemas fontes. Estas informações podem ter a mesma semântica, porém assumirem valores distintos em cada sistema. É papel do DW integrar estas informações em um único ambiente, para que possam assumir um valor uniforme.

- **Variante no tempo:** os dados armazenados no DW são relativos a determinado período de tempo. Este aspecto temporal é muito importante, visto que, à medida que novos dados são armazenados no DW, tem-se várias visões do negócio ao longo do tempo, o que permite que se possa analisar comparativamente o comportamento do negócio em diferentes períodos.

- **Não-Volátil:** ao contrário dos sistemas OLTPs, nos quais os dados antigos são constantemente atualizados e até mesmo excluídos, um DW armazena informações históricas da organização. Por isso, apenas inserções e consultas são permitidas neste ambiente.

Um DW é um banco de dados multidimensional modelado a partir dos conceitos de fatos, medidas, dimensões e hierarquia de dimensões. O fato é o objeto de análise do DW, enquanto as medidas são atributos dos fatos na forma de variáveis numéricas. Considerando um DW de tema **Vendas**, por exemplo, pode-se ter como medidas, as variáveis numéricas: **Quantidade Vendida** e o **Valor da Venda**. Por outro lado, as dimensões caracterizam os fatos e servem como critérios para a agregação de medidas. Possíveis exemplos de dimensões para o DW Vendas seriam: **Produto** (representando um produto vendido), **Filial** (informações sobre

a filial na qual a venda foi realizada) e **Tempo** (dimensão temporal representando em que momento se deu a venda). Ressalta-se que um DW sempre terá uma dimensão temporal justamente para permitir que o negócio seja analisado comparativamente ao longo do tempo.

Outro importante conceito referente à modelagem de DW é o conceito de hierarquia. Dimensões são organizadas em hierarquias, sendo permitido uma ou mais hierarquias por dimensão. Hierarquias são compostas por níveis, onde cada nível representa uma forma como os dados podem ser analisados. Considerando, por exemplo, a dimensão **Tempo** do DW **Vendas**, pode-se ter a seguinte hierarquia: Dia -> Mês -> Semestre -> Ano. A partir desta hierarquia, usuários gestores poderiam visualizar o total de vendas por mês (nível Mês), ou se preferirem uma informação mais agregada (sumarizada), poderiam visualizar o total de vendas por ano (nível Ano).

Conceitualmente, o modelo de dados de um DW pode ser enxergado como um cubo de dados, no qual os eixos são as dimensões e as células são os fatos. A figura 1.2 ilustra um cubo de dados que representa um DW projetado para análise das vendas de determinada empresa.

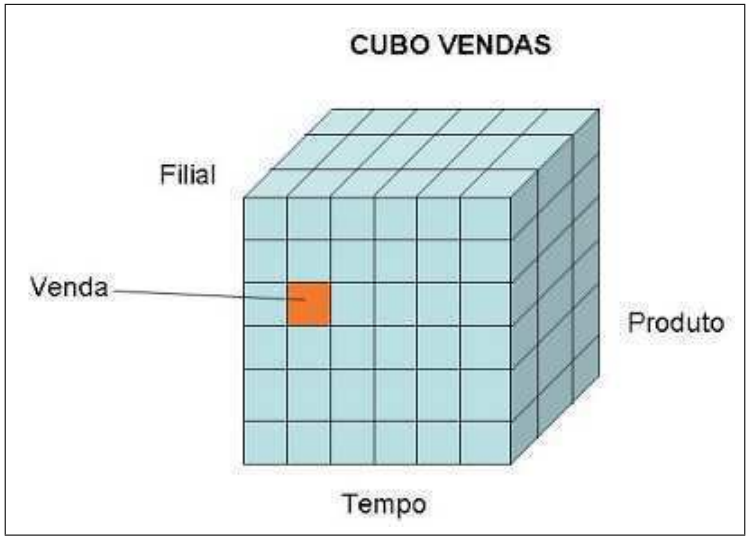

Figura 1.2: Cubo Vendas

Cubos de dados como o da figura 1.2 são comumente implementados em SGBDs relacionais a partir de um esquema especial, denominado esquema em estrela (*star schema*) [3]. Em um esquema em estrela conforme o da figura 1.3, a tabela de fatos (estrela), posiciona-se ao centro, contendo as medidas importantes para análise de negócio. Por outro lado, as tabelas

de dimensões (os planetas), contém informações descritivas que vão dar características aos fatos, além de servirem como critério para agregação das medidas. Note que as hierarquias estão implícitas nas dimensões, como por exemplo, a hierarquia **Filial** -> **Cidade** -> **Estado** -> **Região** da dimensão Filial, e a hierarquia **Dia** -> **Mês** -> **Semestre** -> **Ano** da dimensão Tempo.

Outro aspecto dos esquemas em estrela é que as tabelas de dimensão se relacionam com a tabela de fatos a partir de relacionamentos 1:N. Neste caso, a chave-primária da tabela de fatos é a composição das chaves-estrangeiras das dimensões.

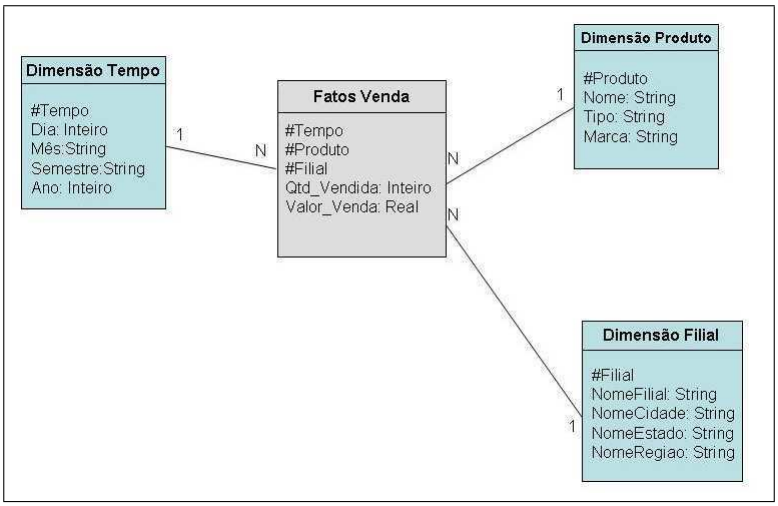

Figura 1.3: Esquema em Estrela

Data Warehouses são explorados por uma categoria de ferramentas apropriadas para a abordagem multidimensional. Estas ferramentas são chamadas de ferramentas OLAP, as quais são descritas na próxima subseção.

1.1.2 Ferramentas OLAP

Como elucidado anteriormente, *Data Warehouses* são bancos de dados com características específicas, o que os diferem de outros bancos de dados convencionais. Estas características dificultam o uso de ferramentas de geração de relatórios tradicionais. Neste ambiente, são necessárias ferramentas que implementem funcionalidades específicas para lidar com a abordagem multidimensional. Estas ferramentas são denominadas ferramentas OLAP (*On-line Analytical Processing*) [5].

Uma ferramenta OLAP é o "principal cliente"de um *Data Warehouse*. Com ela, gestores podem realizar consultas *ad-hoc* ao DW, de forma rápida e interativa utilizando um conjunto de operações especiais denominadas de operações OLAP.

Operações OLAP permitem aos usuários navegarem e explorarem dimensões e suas hierarquias, viabilizando a análise das informações do DW em vários níveis de detalhe e perspectivas. Exemplos de operações OLAP são descritos a seguir:

- **Roll-up**: realizar uma operação de *Roll-up* significa ir de um nível mais baixo a um nível mais alto na hierarquia de determinada dimensão. O resultado desta operação é uma informação mais agregada (ou sumarizada). Considerando que um usuário gestor esteja analisando as vendas de sua empresa por mês. A partir de uma operação de *Roll-up*, ele poderá visualizar as vendas em um nível mais agregado, por exemplo, por semestre, ou ano.

- **Drill-down**: a operação *Drill-down* tem efeito oposto à *Roll-up*, ou seja, o usuário tem uma informação agregada e deseja descobrir detalhes sobre a mesma. Considerando que um usuário gestor esteja analisando o total de vendas por região e deseja analisar o desempenho de cada estado separadamente. Esta análise pode ser realizada a partir de uma operação de *Drill-down* na hierarquia Filial.

- **Slice**: o *Slice* seleciona as dimensões que serão projetadas em determinada consulta, omitindo as dimensões restantes que não forem importantes naquele momento. Considerando, por exemplo, o esquema em estrela da Figura 1.3, um usuário gestor pode visualizar o total de vendas considerando apenas os membros das dimensões Produto e Filial e omitir a utilização da dimensão Temporal.

- **Dice**:o *Dice* permite restringir os valores dos membros de determinada dimensão a partir de um predicado. Por exemplo, um usuário gestor deseja analisar o total de vendas no ano de 2009. Esta restrição (Ano = 2009), caracteriza um *Dice* realizado no nível Ano da dimensão Tempo. A ferramenta totalizará apenas as vendas que ocorreram no ano de 2009.

- **Pivot**: a operação de *Pivot* altera a perspectiva de visualização do cubo. Com isso, o usuário pode visualizar a mesma informação em perspectivas distintas.

1.2 DW Espacial

Estima-se que 80% dos dados armazenados em bancos de dados contenham algum tipo de componente com contexto espacial [7], como por exemplo a rua, o bairro ou o CEP de um endereço de determinada loja. *Data Warehouses* convencionais, tratam estes componentes no formato alfanumérico. Neste ambiente, os componentes com contexto espacial não são georreferenciados, isto é, representados segundo algum sistema de referência espacial, como o sistema de coordenadas (latitude/longitude), por exemplo [8].

A utilização de componentes espaciais georreferenciados (ou simplesmente dados espaciais) em ambientes de *Data Warehousing* traz grandes benefícios no processo de tomada de decisão, visto que permitem descobrir novas informações a respeito dos fatos a partir da realização de consultas mais abrangentes. Questões do tipo: **"Qual o total de vendas de produtos considerando os bairros que estão a uma distância Y da filial X?"** são possíveis de serem respondidas, mostrando aos gestores informações até então desconhecidas.

Diante deste benefício, algumas propostas foram feitas visando integrar dados espaciais ao ambiente de *Data Warehousing*, surgindo assim o termo *Data Warehouse* Espacial (*Spatial Data Warehouse*) [9].

Ainda não existe um consenso a respeito dos conceitos relacionados a *Data Warehouse* Espacial: diferentes autores propõem diferentes abordagens para o uso do dado espacial em esquemas de DW. Contudo, sabe-se que o ideal é integrar dados convencionais e espaciais em um único ambiente. Para esta tarefa, os bancos de dados espaciais (*Spatial Databases*) são ideais, pois implementam um conjunto de funções e algoritmos que facilitam o armazenamento, a recuperação e a manipulação de objetos espaciais [8].

Embora não exista um consenso na literatura, pode-se enxergar um DW Espacial como uma extensão natural de um DW convencional, levando a novos conceitos como dimensão espacial, hierarquia espacial e medida espacial, além dos conceitos já existentes. Estes conceitos são apresentados a seguir, segundo o ponto de vista de diferentes autores.

1.2.1 Dimensão Espacial

Para Han *et al* [10], o que caracteriza uma dimensão como sendo espacial ou não espacial, é o tipo de dado de seus membros, isto é, se seus membros são geometrias georreferenciadas

ou apenas dados alfanuméricos. Neste sentido, os autores apontam a existência de três tipos de dimensões em um DW Espacial:

- **Dimensão Convencional**: em uma dimensão convencional, os níveis de agregação são representados na forma textual, como exemplo a dimensão Filial do esquema da figura 1.3, organizada segundo a hierarquia NomeFilial -> NomeCidade -> NomeEstado -> NomeRegião (Ex: Bodocongó -> Campina Grande -> Paraíba -> Nordeste). Ou seja, mesmo a dimensão tendo um contexto espacial - já que representa lugares - é considerada uma dimensão convencional.

- **Dimensão Espacial**: em uma dimensão espacial, todos os níveis são representados por geometrias georeferenciadas (ou objetos espaciais) como polígonos, pontos, linhas, etc. Um conjunto de níveis representados por objetos espaciais forma uma hierarquia espacial. Por exemplo, a dimensão Filial do esquema em estrela da figura 1.3, poderia ser organizada segundo a hierarquia espacial: GeometriaFilial -> GeometriaCidade -> GeometriaEstado -> GeometriaRegião, onde os níveis GeometriaFilial, GeometriaCidade, GeometriaEstado, GeometriaRegião seriam polígonos que representariam a filial, sua cidade, seu estado e sua região, respectivamente.

- **Dimensão Mista**: uma dimensão mista é uma dimensão onde os níveis de agregação podem ser representados tanto na forma textual como a partir de geometrias georreferenciadas, por exemplo, uma dimensão formada pelos níveis: GeometriaFilial -> NomeCidade -> NomeEstado -> GeometriaRegião.

Em contrapartida, Bédard *et al* [9] apresentam outra categorização para dimensões de um DW. Para os autores, uma dimensão pode ser convencional - quando não apresentar contexto espacial - ou espacial, esta última podendo ser de três tipos:

- **Dimensão Espacial Não Geométrica**: É uma dimensão com um contexto espacial cujos membros são representados por valores alfanuméricos. Como exemplo, a dimensão Filial do esquema em estrela da figura 1.3.

- **Dimensão Espacial Geométrica**: em uma dimensão espacial geométrica todos os níveis de agregação são representados por geometrias georreferenciadas.

- **Dimensão Espacial Mista**: uma dimensão espacial mista é uma dimensão cujos níveis de agregação são formados por dados alfanuméricos e geometrias referenciadas. Por exemplo, uma dimensão organizada pela hierarquia GeometriaFilial -> NomeCidade -> NomeEstado -> GeometriaRegião.

1.2.2 Medida Espacial

Um DW Espacial possui, além de medidas numéricas (convencionais), medidas espaciais. Medidas espaciais são objetos espaciais que representam fatos relacionados a determinado domínio de negócio [8]. Supondo, por exemplo, um DW Espacial destinado à análise da distribuição agrícola de determinado estado, poderia ser de interesse dos gestores analisar as áreas plantadas (polígonos georreferenciados) de determinado tipo de plantação, além de sua correlação com outras variáveis como o município em que se encontra a plantação, o tipo de solo, a precipitação, etc. Neste caso, as áreas plantadas são exemplos de medidas espaciais.

1.2.3 Ferramentas *Spatial OLAP*

As ferramentas OLAP convencionais não são adequadas para lidarem com dados espaciais. Estes dados são manipulados por uma categoria específica de sistemas: os chamados Sistemas de Informações Geográficas (SIG). Um SIG, como é popularmente conhecido, é um sistema computacional projetado para coletar, armazenar, recuperar, manipular e visualizar dados espaciais, ou seja, dados que representam objetos e fenômenos em que a localização geográfica é uma característica inerente à informação e indispensável para analisá-la [11].

Apesar de auxiliarem o processo de tomada de decisão, os SIGs não estão preparados para a abordagem multidimensional dos *Data Warehouses*, sendo necessária uma nova categoria de ferramentas que combine algumas funcionalidades dos SIGs com as funcionalidades das ferramentas OLAP. Diante desta necessidade surgiu o termo Spatial OLAP (SOLAP) [12].

Como definido por Rivest *et al* [12], SOLAP é uma plataforma visual construída para permitir a análise espaço-temporal e exploração de dados com facilidade e rapidez, seguindo uma abordagem multidimensional composta por níveis de agregação disponíveis na forma de tabelas, gráficos e mapas.

Essa nova categoria de ferramenta representa um grande desafio do ponto de vista de implementação, visto que a característica multidimensional combinada ao uso de dados espaciais traz como consequência a necessidade de repensar a forma como as dimensões serão exploradas, como as medidas serão agregadas, além de aspectos relacionados à forma como resultados de consultas serão representados e apresentados aos usuários [13].

Nestas ferramentas, operações OLAP como *Slice e Dice*, devem ser realizadas tanto em dimensões convencionais como em dimensões espaciais. Por isso, operadores de comparação devem ser combinados a operadores espaciais, a fim de prover maior flexibilidade na formulação de consultas. Da mesma forma, operações de *Roll-up e Drill-Down* devem ser repensadas, visto que a agregação de medidas espaciais deve ser feita com novas funções de agregação, tais como: *geometric union, geometric intersection ou convex hull* [8].

1.3 Motivação

Várias propostas de ferramentas SOLAP com características e funcionalidades distintas estão disponíveis na literatura [14], [15], [16], [17], [18], [19]. Estas ferramentas têm em comum o objetivo de integrar características dos SIGs com funcionalidades OLAP, provendo um ambiente único para análise multidimensional-espacial. Entretanto, os trabalhos divergem nos seguintes sentidos: (i) com relação a quantidade de funcionalidades implementadas; (ii) com relação ao uso ou não de tecnologias abertas: algumas propostas utilizam tecnologias abertas, enquanto outras tecnologias proprietárias; (iii) com relação à abrangência dos modelos de dados que implementam: algumas ferramentas implementam modelos de dados que consideram o uso de dados espaciais apenas nas dimensões, enquanto outras implementam modelos de dados mais abrangentes, onde dados espaciais são utilizados tanto em dimensões, como na forma de medidas; (iv) com relação ao uso de operações espaciais na realização de consultas.

Acreditamos que uma ferramenta SOLAP ideal deve possuir as seguintes características:

1. Deve apresentar resultados de consultas na forma de mapas, gráficos e tabelas, permitindo, inclusive, que mapas sejam manipulados através de funções como *zoom in*, *zoom out* e *pan*.

2. Deve ser voltada para Web para que possa ser acessada através da Internet.

3. Deve utilizar tecnologias abertas, permitindo que seja implantada e estendida com baixo custo.

4. Deve implementar um modelo de dados que integre totalmente dados espaciais e não espaciais, permitindo a criação de aplicações SOLAP mais poderosas.

5. Deve combinar o uso de operações OLAP com operações espaciais permitindo a formulação de consultas mais abrangentes.

6. Deve ser simples, para que seja possível sua utilização pelos principais interessados em tais ferramentas: os usuários não especialistas, isto é, usuários gestores.

Apesar do grande número de proposta de ferramentas SOLAP, não observamos dentre as encontradas, uma que contemple de forma satisfatória as características que julgamos serem ideais a uma ferramenta SOLAP. Mais especificamente, não encontramos uma ferramenta que combine a utilização de operadores multidimensionais e operadores espaciais, e que seja destinada a usuários não especialistas, isto é, os gestores. Esse fato serviu como motivação para este trabalho, que visa o desenvolvimento de uma nova ferramenta SOLAP destinada especialmente a usuários gestores, que combina o uso de operações multidimensionais com operações espaciais permitindo que consultas complexas e abrangentes sejam definidas e executadas. A ferramenta proposta permite que usuários gestores definam seus próprios esquemas de DW Espacial e realizem consulta a DW Espaciais criados a partir destes esquemas. Para isto, a ferramenta dispõe de linguagens gráficas para definição de esquemas e consultas, implementadas na forma de um conjunto de interfaces gráficas de alto nível de abstração. Nossa proposta de ferramenta foi implementada seguindo a filosofia de *framework*, o que permite que programadores possam estendê-la e utilizá-la em tecnologias distintas de Sistemas Gerenciadores de Banco de Dados (SGBD).

1.4 Objetivos

Os objetivos desta dissertação foram divididos em geral e específicos.

1.4.1 Objetivo Geral

O objetivo geral é desenvolver um *framework* para *Data Warehouse* Espacial com o qual usuários gestores possam definir conceitualmente esquemas de DW Espacial e realizar consultas a DW Espaciais criados a partir destes esquemas.

1.4.2 Objetivos Específicos

Visando alcançar o objetivo geral, os seguintes objetivos específicos foram traçados:

- Definição de um metamodelo conceitual multidimensional espacial.
- Implementação do *framework* com tecnologias abertas e bem difundidas.
- Extensão do *framework* para os SGBDs Oracle 10g e PostgreSQL.
- Validação do *framework* e das extensões implementadas a partir de dois estudos de caso.

1.5 Estrutura da dissertação

O documento de dissertação está estruturado da seguinte forma:

No capítulo 2, são apresentados alguns trabalhos relacionados com suas principais características. Ao final do capítulo é feita uma breve discussão a partir de um comparativo entre os trabalhos.

O capítulo 3 traz a proposta de *framework* para DW Espacial, no qual serão apresentados os requisitos que guiaram seu desenvolvimento, sua arquitetura, aspectos relacionados à implementação de duas extensões do *framework*, além das linguagens gráficas de definição de esquemas e consultas.

No capítulo 4, são apresentados dois estudos de caso desenvolvidos com o auxílio do *framework*: um destinado à análise da distribuição agrícola do estado da Paraíba e outro destinado à análise dos atendimentos realizados pelo SAMU (Serviço de Atendimento Móvel de Urgência) do município de Aracaju-Sergipe.

Por fim, o capítulo 5 traz as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Este capítulo tem como objetivo apresentar alguns trabalhos relacionados ao tema de *Data Warehouse* Espacial encontrados na literatura. Mais especificamente, serão apresentadas algumas propostas recentes de ferramentas SOLAP, assim como seus respectivos modelos de dados, visando cobrir tanto questões referentes à implementação de ferramentas SOLAP, como também questões relacionadas aos conceitos de DW Espacial.

Para isso, os trabalhos a seguir serão descritos da seguinte forma:

1. Uma apresentação resumida observando as principais características de cada proposta.
2. Um exemplo ilustrativo, quando houver.
3. Críticas: elucidando aspectos positivos e negativos de cada trabalho.

2.1 JMap Spatial Olap

O termo SOLAP (Spatial OLAP), originalmente definido por Bédard em [14], designa ferramentas especiais que combinam funcionalidades dos sistemas de informações geográficas com características das ferramentas OLAP. Estas ferramentas estão inseridas em um ambiente chamado de *Spatial Data Warehousing*, cujo elemento central é o DW Espacial [9]. Em [12] e [20], Rivest *et al* apresentam detalhes deste ambiente, caso em que são apresentados os conceitos relacionados a DW espacial, além de características importantes que devem ser atendidas pelas ferramentas SOLAP.

Para os autores, um DW Espacial possui, além de dimensões convencionais, três tipos de dimensão espaciais: dimensão espacial não geométrica, cujos níveis de agregação possuem um contexto espacial porém, representados na forma descritiva, como por exemplo, nome de bairros, cidades ou estados; dimensão espacial geométrica, cujos níveis de agregação são objetos espaciais que podem ser apresentados na forma de mapas temáticos; e dimensão espacial mista, cujos membros podem ser representados tanto na forma descritiva, como também por objetos espaciais. Os autores definem ainda dois tipos de medidas espaciais: o primeiro tipo é um conjunto de objetos espaciais formados a partir de operações espaciais como união espacial, interseção espacial, etc. O segundo tipo são medidas espaciais formadas por operadores métricos ou topológicos, como por exemplo, à distância, área, etc.

Com relação a ferramentas SOLAP, uma série de características desejáveis são ressaltadas pelo autores, entre elas a necessidade de implementação de operadores SOLAP, como *spatial drill-down* e *spatial roll-up*, para permitir a exploração de dimensões espaciais geométricas e mistas, além da utilização de mapas e diagramas para apresentação de resultados de consultas.

Considerando estes conceitos e características, criou-se a JMAP Spatial OLAP [20], uma ferramenta SOLAP que contempla funcionalidades SIGs e OLAP. Segundo os autores, a JMap Spatial OLAP é a primeira ferramenta comercial a combinar as duas tecnologias, sendo fruto de uma parceria com a empresa Kheops [21].

A JMap OLAP é dividida em dois módulos: um módulo de administração, formado por *wizards* para configuração do banco de dados utilizado pela ferramenta e um módulo de visualização que permite aos usuários finais visualizarem e explorarem os dados multidimensionais. A ferramenta implementa os três tipos de dimensão espacial definidos por Bédard e apenas o primeiro tipo de medida espacial.

A figura 2.1 apresenta a interface gráfica da ferramenta JMAP, ilustrando uma aplicação SOLAP destinada à análise de incidência de doenças respiratórias na província de Quebec, Canadá. Na aplicação de exemplo, as seguintes dimensões são consideradas: Doença, Tipo de caso (incidência, morte ou hospitalização), Faixa etária, Sexo, Território e Tempo. Além das seguintes medidas: número de casos, faixa padronizada, figura comparativa e indicadores estatísticos.

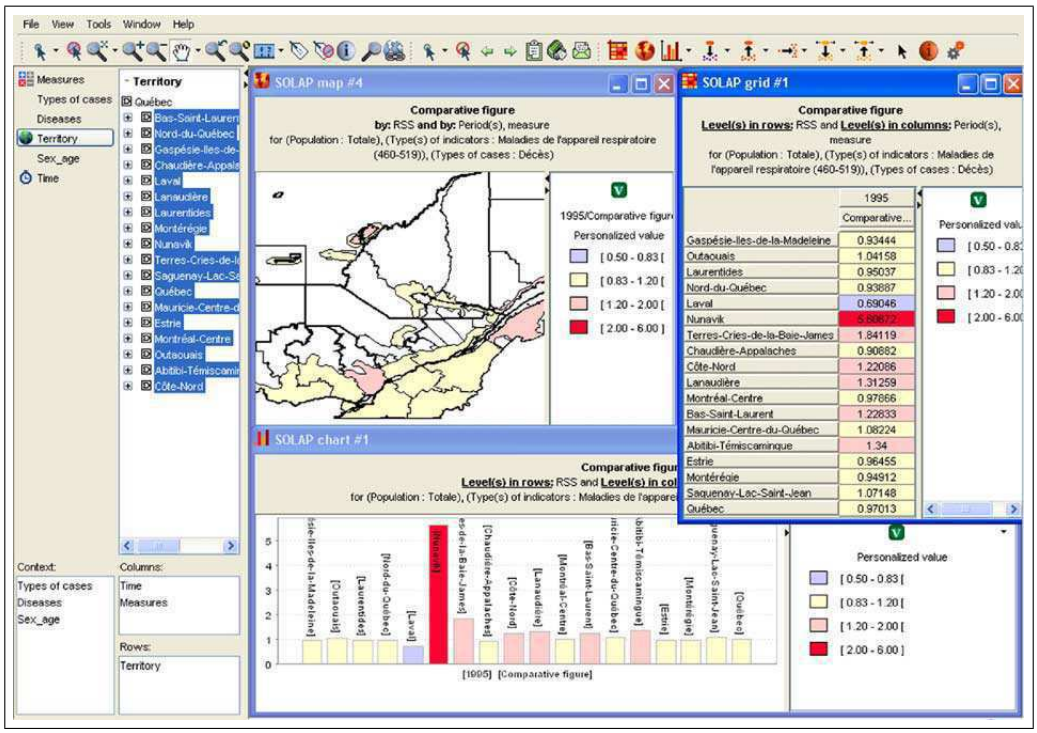

Figura 2.1: Interface da JMAP Spatial Olap

A JMAP Spatial OLAP permite a visualização de dados em diferentes tipos de mapas e gráficos, além da forma tabular. Ao todo, 7 tipos de gráficos são implementados, entre eles, gráficos de barras, pontos, pizza, etc. Ressalta-se, ainda, que existe sincronismo entre os componentes da interface, ou seja, uma ação no componente tabular, por exemplo, será automaticamente refletida nos demais componentes.

A ferramenta utiliza a abordagem federada de DW Espacial. Nesta abordagem, os dados multidimensionais não são armazenados no mesmo ambiente dos dados espaciais. No caso específico do JMAP, os dados analíticos são armazenados em um SGBD relacional segundo um esquema em estrela e ponteiros fazem a associação entre os dados armazenados no DW e arquivos contendo geometrias.

Outro aspecto observado a partir de um demo disponibilizado em [22], é que a ferramenta não permite o uso de operadores espaciais, isto é, operadores métricos ou topológicos.

2.2 GeoWOlap

A GeoWOLAP [15] é uma ferramenta SOLAP voltada para Web que implementa o modelo de dados multidimensional GeoCube [23] [24], cuja principal característica é permitir a modelagem de medidas espaciais como objetos complexos, isto é, entidades descritas por um conjunto de atributos geométricos, numéricos ou temáticos. Esta característica traz como consequência outras particularidades do modelo, como por exemplo, a utilização de relacionamentos N:M entre fatos e dimensões, o tratamento de medidas e dimensões de forma simétrica e a utilização de funções *ad-hoc* defindas pelos usuários para a realização de agregação de medidas.

Do ponto de vista da arquitetura, a GeoWOLAP é dividida em três camadas, como ilustrada na figura 2.2. A camada de armazenamento (*Warehouse*) utiliza o SGBD Oracle 10g com sua extensão espacial Oracle Spatial [25]. Nesta camada são armazenadas as dimensões e tabela de fatos, além de visões materializadas para otimizar o desempenho de consultas. Na camada central, uma extensão do servidor OLAP Mondrian foi utilizada para tratar a questão da agregação de medidas espaciais. Por fim, a camada de interface utiliza os componentes JPivot [26] e o MapXtreme Java [27], sendo utilizados para a apresentação de resultados de consultas na forma tabular e cartográfica respectivamente.

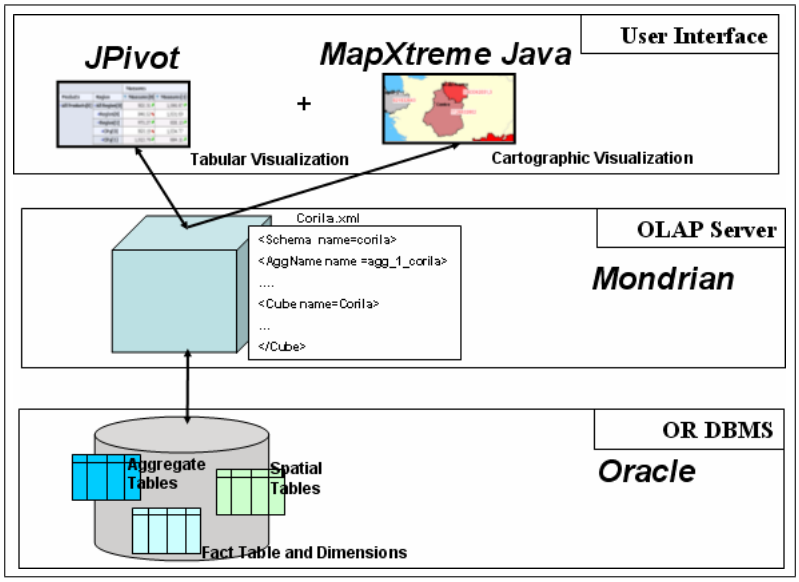

Figura 2.2: Arquitetura da GeoWOlap

Como exemplo ilustrativo, é apresentada uma aplicação SOLAP referente ao modelo

da figura 2.3, cujo objeto de análise é a *unità barenale*, parte de uma lagoa de Venice que fica exposta por um período de tempo enquanto a maré está baixa. O objetivo é analisar a poluição da *unità barenale* e sua correlação com outras variáveis, tais como, o tempo, o tipo de poluente e a profundidade da lagoa no momento da medição.

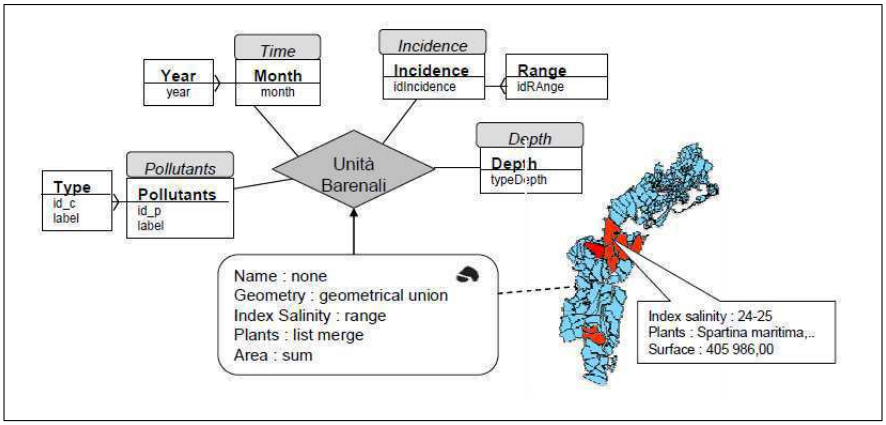

Figura 2.3: Aplicação SOLAP: Lagoa de Venice

A aplicação SOLAP consiste em quatro dimensões (Poluente, Tempo, Incidência e Profundidade), além da medida *unità barenale* representada por um objeto complexo composto por uma geometria, nome, índice de salinidade, lista de plantas e área. A figura 2.4 apresenta a interface da aplicação na ferramenta GeoWOLAP.

A interface gráfica da GeoWOLAP é composta por 4 (quatro) painéis que visam satisfazer um conjunto de requisitos de funcionalidades definidos em [13] como sendo prioritários em ferramentas dessa categoria, por exemplo, a apresentação de resultados na forma tabular e cartográfica e o sincronismo entre os componentes na interface. A descrição de cada painel da ferramenta é apresentada a seguir:

- Barra de ferramentas OLAP: permite a utilização de funcionalidades OLAP, tais como seleção de cubo (dimensões e medidas), drill-down, roll-up, etc.
- Barra de ferramenta SIG: implementa funcionalidades SIG, permitindo, por exemplo, realização de operações de *zoom*, pan, distância entre pontos, impressão de maps, etc.
- Pivot Table: painel para a visualização e análise dos resultados de consultas na forma tabular, implementado com JPivot.

- Visão do Mapa: painel para a visualização e análise dos resultados de consultas na forma cartográfica, implementado com o MapExtreme.

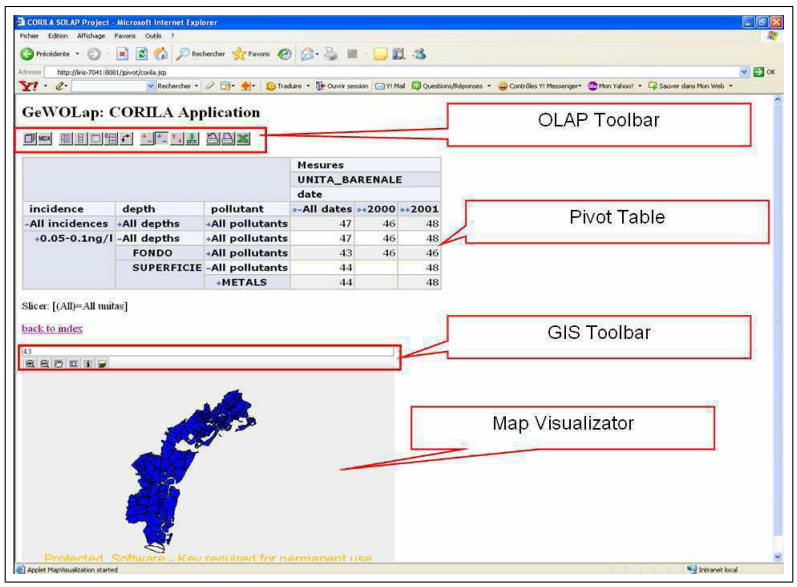

Figura 2.4: Interface da GeoWOlap

Um aspecto que pode ser observado na GeoWOLAP é a forte dependência da ferramenta com relação a tecnologias proprietárias, visto que a ferramenta utiliza o SGBD Oracle e a API MapExtreme para visualização de mapas. Além disso, observou-se é que o GeoWOLAP não utiliza operações espaciais (métricas ou topológicas) para restringir valores em dimensões espaciais, sendo esta característica elencada pelos autores como trabalho futuro.

2.3 SOVAT

SOVAT (*Spatial OLAP Visualization and Analysis Tool*) [16] [28], é outra proposta de ferramenta SOLAP. Sua implementação foi guiada por dois requisitos principais: prover uma interface gráfica que esconde dos usuários finais detalhes de consultas e integrar funcionalidades SIG e OLAP em um único ambiente gráfico, provendo diferentes representações dos dados a serem analisados.

A arquitetura da ferramenta é dividida em três camadas:

- Camada de dados: a camada de dados utiliza a abordagem federada de *DW* espacial

sendo utilizado o SGBD o Microsoft SQL Server 2000 para armazenar dados convencionais. Estes dados são associados à outra base de dados contendo dados espaciais.

- Camada de aplicação: a camada de aplicação tem como objetivo integrar dados do DW com dados espaciais, além de implementar mecanismos para analisar estes dados integrados.

- Camada de cliente: a camada de cliente é composta por uma aplicação (*desktop*) *standalone* implementada na plataforma .NET. A interface gráfica da SOVAT permite que o usuário defina consultas de forma fácil a partir de simples interações, além de apresentar resultados de consultas em gráficos e mapas.

A SOVAT foi desenvolvida especialmente para o domínio de saúde pública tendo como objetivo auxiliar profissionais e pesquisadores dessa área no processo de tomada de decisão. Para isto, a ferramenta implementou um esquema multidimensional composto pela medida **População afetada** e por um conjunto de dimensões como Idade, Sexo, Raça, Nível Educacional, Região (Urbana ou Rural), Diagnóstico, Peso de Nascimento, Geografia (Localização) e Ano. A Figura 2.5 ilustra a interface gráfica da SOVAT para o domínio de aplicação em questão.

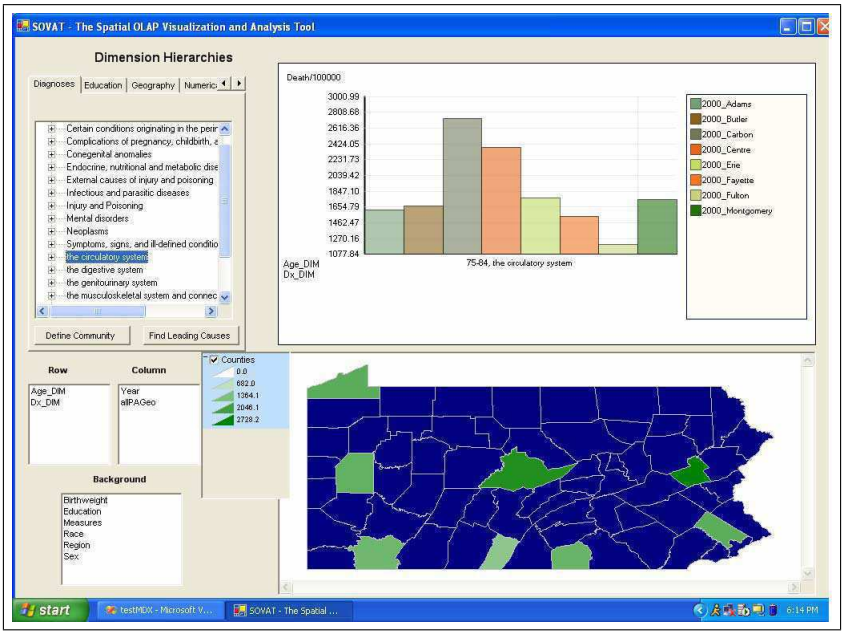

Figura 2.5: Interface da SOVAT

Scotch *et al* [29] realizaram ainda um estudo de usabilidade da SOVAT com profissionais

da área de saúde visando identificar possíveis problemas relacionados à usabilidade da ferramenta. Além disso, foi apresentado como característica marcante da ferramenta um novo operador SOLAP denominado *drill-out*, que visa resolver consultas envolvendo detecção de fronteiras de geometrias, como por exemplo, consultas do tipo "Quais as regiões fazem fronteira com a região X?", ou "Qual o maior índice de câncer das regiões que fazem fronteira com determinada região?".

O resultado deste último trabalho foi uma extensão da ferramenta com melhorias na interface gráfica, ilustrada na Figura 2.6.

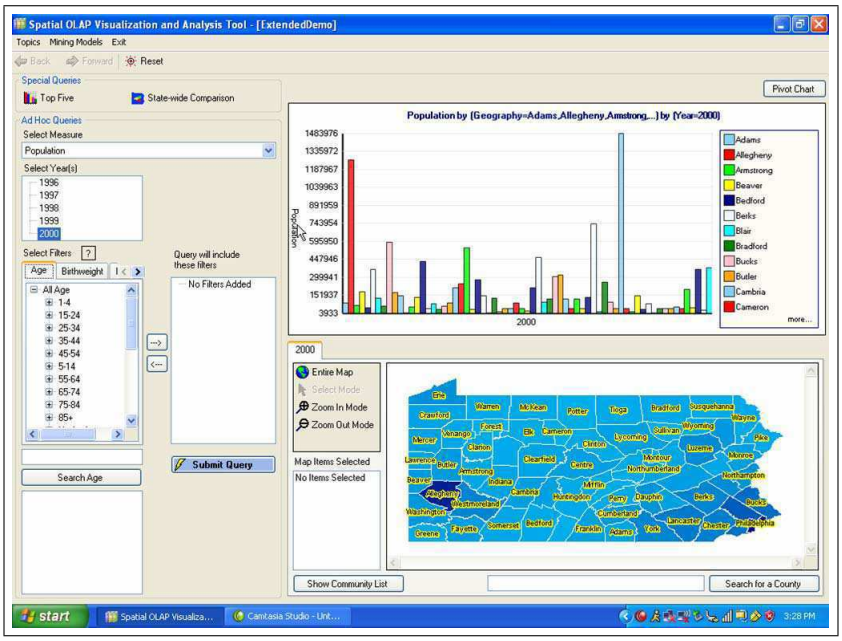

Figura 2.6: Interface da Extended SOVAT

Como explanado anteriormente, a SOVAT é uma ferramenta implementada especialmente para profissionais e pesquisadores da área de saúde pública, o que a torna específica para este domínio de aplicação. Além disso, observa-se o uso de tecnologias proprietárias tanto na camada de *Data Warehouse* quanto na camada de cliente. Estas características aumentam o custo da ferramenta, além de torná-la dependente de uma plataforma específica de sistema operacional.

Outro aspecto é que o modelo de dados implementado pela ferramenta não contempla a utilização de medidas espaciais, o que pode considerado uma limitação da ferramenta.

2.4 Piet

A ferramenta PIET, proposta em [18], é outra proposta de ferramenta SOLAP que utiliza a abordagem federada de DW Espacial. Em linhas gerais a PIET utiliza dois bancos de dados distintos: um *Data Warehouse* convencional que armazenada dados alfanuméricos e um banco de dados geográfico que armazena mapas. Estes dois bancos de dados formam as chamadas partes OLAP e SIG, respectivamente.

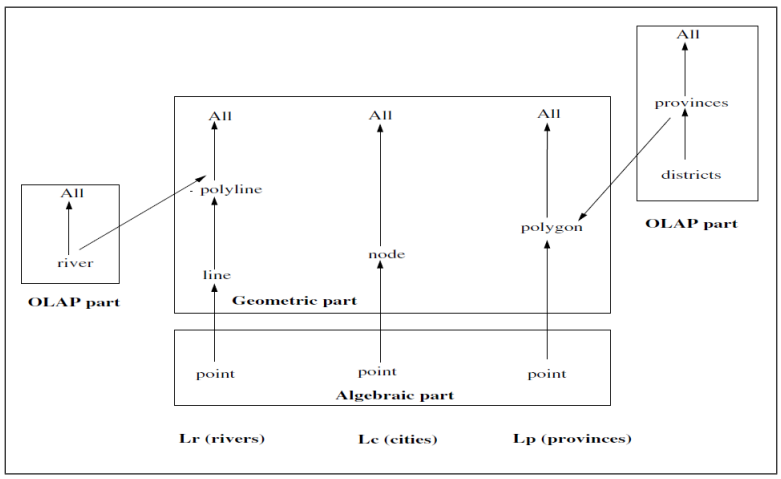

Figura 2.7: Arquitetura do PIET

Como pode ser observado na Figura 2.7, a parte SIG é subdividida em duas outras partes: algébrica e geométrica. A subdivisão algébrica representa os infinitos pontos que compõem uma camada temática do mapa, enquanto a subdivisão geométrica armazena apenas identificadores. Estes identificadores são associados entre si para formar novos elementos geométricos. Por exemplo, dois pontos P1 e P2 podem ser associados para formar uma linha L1, que por sua vez, pode ser associada à outra linha L2 e formar uma multi-linha. Esta associação se dá a partir de grafos direcionados que indicam quais elementos formam outros elementos geométricos. O resultado é um grande grafo que representa todo o mapa geográfico da aplicação. Além disso, os grafos servem para associar um elemento geométrico com seu correspondente na parte OLAP.

Para melhorar o desempenho de consultas que envolvam operações espaciais, a ferramenta realizada a pré-computação de cobertura (*overlay*), a partir de uma técnica que os autores chamaram de *subpolygonization*.

Basicamente, a arquitetura da PIET é dividida em três módulos, descritos a seguir:

- Configuração de dados brutos: responsável por juntar dados numéricos e espaciais e armazená-los em um *Data Warehouse* ou em um Mapa.

- Gerador de dados pré-calculados: tem como objetivo a geração de dados pré-calculados, como por exemplo, um conjunto de subgeometrias formadas pelo processo de *subpolygonization*, metadados que descrevem a estrutura do DW e do mapa, além da associação entre os elementos geométricos e as informações contidas no DW.

- Processador de consulta: que realiza o processamento de quatro tipos de consultas: consulta apenas geométrica, consulta com agregação geométrica, consulta OLAP e consulta SIG-OLAP.

Do ponto de vista da implementação, a Piet utiliza um conjunto de soluções abertas, como o SGBD PostgreSQL, o servidor OLAP Mondrian, além de APIs para apresentação de mapas e tabelas. Sua arquitetura é dividida em três camadas, como ilustrada na Figura 2.8:

Na camada de dados encontra-se o *Data Warehouse*, implementado no SGBD PostgreSQL, e outro banco de dados contendo mapas que utiliza o PostgreSQL com sua extensão espacial, PostGis [30]. Além disso, a camada de dados contém arquivos de sistema com informações relacionadas ao esquema da aplicação.

Na camada central da arquitetura encontra-se a máquina virtual Java, responsável pela execução do *engine* Piet e Mondrian. Por fim, na camada de cliente encontram-se as interfaces gráficas utilizadas pelos usuários para a realização de consultas.

A camada de cliente é composta por duas aplicações distintas. Uma aplicação Web, chamada de Piet-Web que utiliza um conjunto *frameworks* Java e outra aplicação desktop, chamada Piet-Jump, baseada no *framework* Jump. A aplicação Web é especialmente destinada a realização de consultas OLAP, enquanto a aplicação desktop tem como foco a realização de consultas com operações espaciais.

Para validação da ferramenta PIET foi proposta uma aplicação SOLAP objetivando analisar as vendas de produtos em lojas da Bélgica. A interface gráfica para esta aplicação pode ser observada na Figura 2.9.

A ausência de um único ambiente gráfico para realização de consultas é um aspecto negativo da ferramenta PIET. Apesar de os autores informarem que é possível a realização

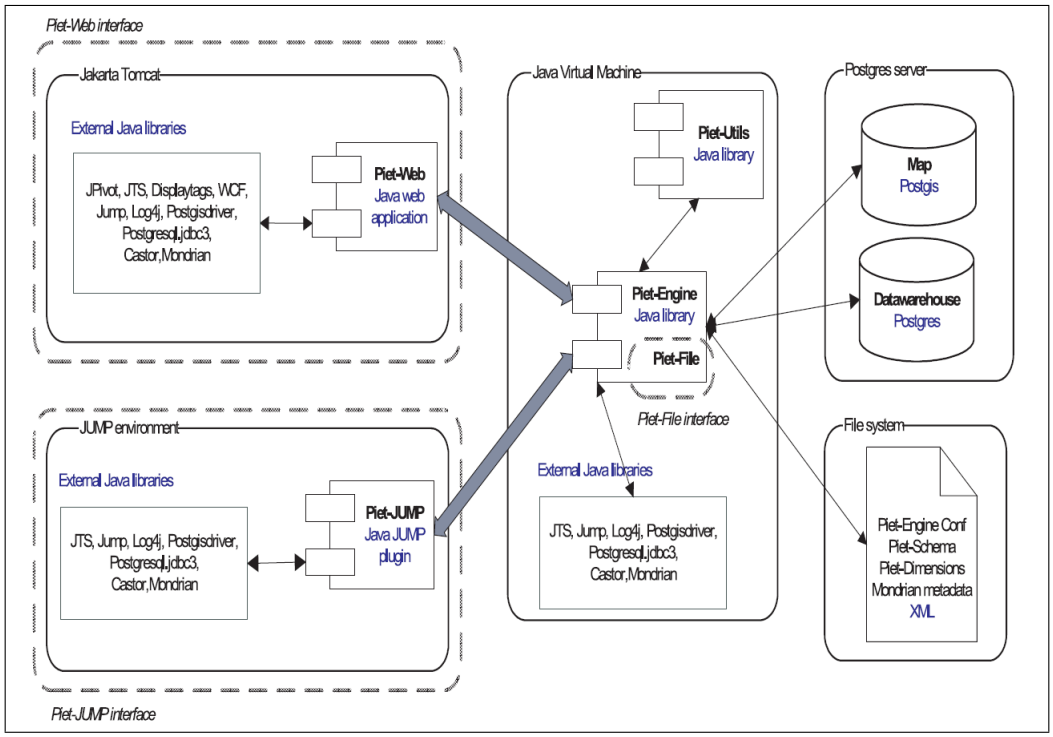

Figura 2.8: Arquitetura do PIET

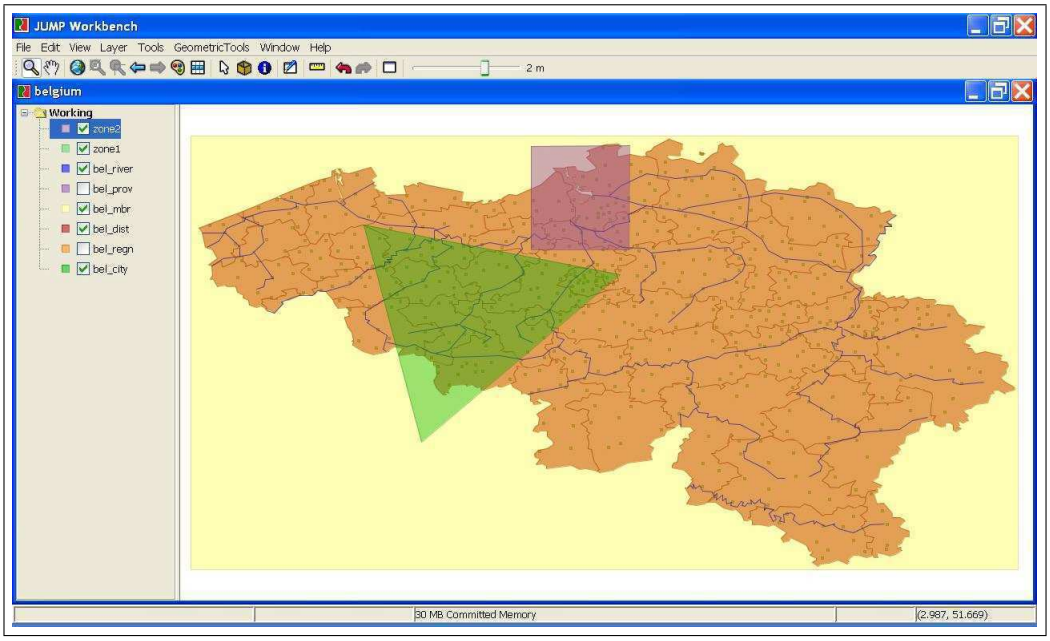

Figura 2.9: Interface da PIET - aplicação desktop

de consultas OLAP juntamente com operações espaciais, não fica claro como estas consultas podem ser definidas e apresentadas na interface gráfica da ferramenta.

2.5 GeoMDQL

A GeoMDQL, [19] [31], é uma linguagem de consulta geográfica e multidimensional que implementa um conjunto de operações para permitir que consultas sejam realizadas em um ambiente de suporte a decisões geográfico e multidimensional. GeoMDQL é uma variação da linguagem de consulta multidimensional MDX, estendida para prover novos operadores que lidam com dados espaciais. GeoMDQL está inserida no contexto da GolapWare, a qual pode ser enxergada como uma instância da arquitetura GOLAPA, ilustrada na Figura 2.10.

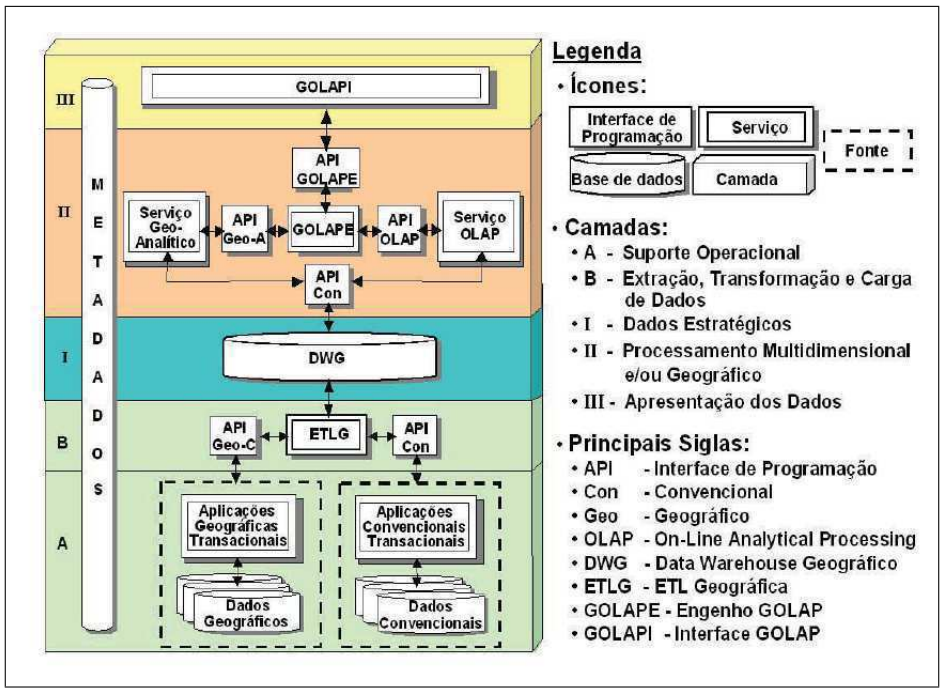

Figura 2.10: Arquitetura GOLAPA

A GOLAPA (*Geographical On-Line Analytical Processing Architecture*) é uma arquitetura de software que prevê modelos para DW Espacial, metadados para integração, serviços para integração de processamento geográfico e multidimensional, além de uma ferramenta para modelagem de esquemas de DW Espacial.

GolapWare implementa as camadas I, II e III da arquitetura GOLAP, as quais provêm respectivamente, dados, serviços e interface com o usuário. Cada camada de GolapWare

será descritas a seguir.

Na camada I de GolapWare encontra-se o DW Espacial, implementado no SGBD PostgreSQL com extensão espacial (PostGis). A modelagem de esquemas, sua validação e a geração de *scripts* lógicos para a criação do DW Espacial foi feita com a ferramenta GeoDW-Case [32], ilustrada na figura 2.11.

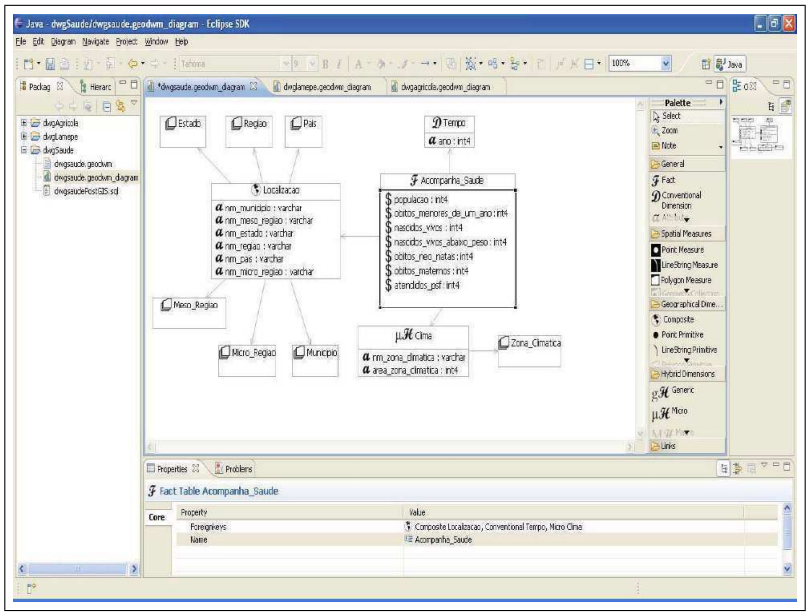

Figura 2.11: GeoDWCase

Na camada II encontram-se os mecanismos para o processamento geográfico e multidimensional, implementados a partir de uma extensão do servidor OLAP Mondrian [33] para lidar com dados espaciais e interpretar consultas na sintaxe da GeoMDQL.

Por fim, na camada III encontram-se os componentes de interface com o usuário. Esta última camada foi implementada em duas abordagens distintas:

- Abordagem Web: utiliza tecnologias como JSP (*Java Server Pages*), SVG (*Scalable Vector Graphics*), HTML (*HyperText Markup Language*), destinada a usuários que queiram realizar consultas geográficas e multidimensionais pela Internet.

- Abordagem *stand-alone*: uma aplicação cliente *desktop* baseada na integração e extensão das tecnologias *Java Plugin Framework*, *OpenJump* [34] e *JRubik* [35]. A interface gráfica da abordagem *desktop* pode ser observada na Figura 2.12. O painel (A) da Figura 2.12 é a área na qual as consultas na sintaxe GeoMDQL são especificadas.

O Painel (B) exibe os resultados de consultas na forma tabular. Por sua vez, no Painel (C) são exibidos os mapas resultantes de consultas. O painel (D) permite a construção de gráficos.O painel (E) permite selecionar consultas anteriormente armazenadas. O Painel (F) permite explorar as dimensões, fatos, hierarquias da aplicação. Por fim, o painel (G) permite se trabalhar com outros catálogos armazenados pela interface.

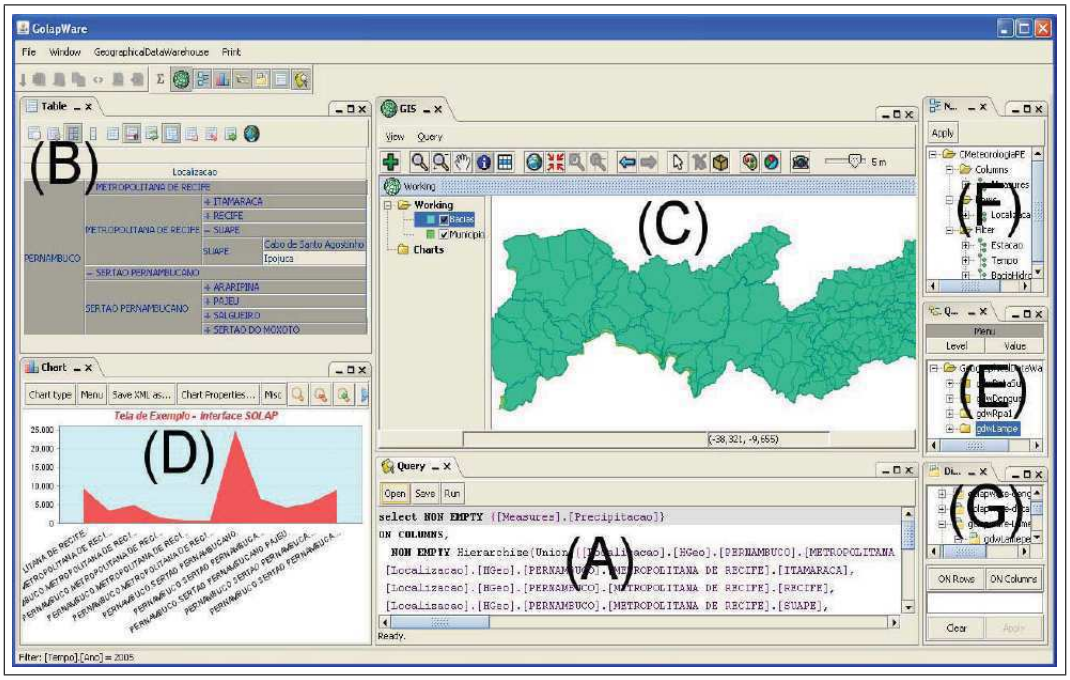

Figura 2.12: Interface GolapWare - aplicação desktop

Como pode ser notado, GolapWare não dispõe de meios pelos quais os usuários possam especificar suas consultas a partir de interações simples com a interface gráfica. Ao invés disso, o usuário deve informar uma consulta na sintaxe da GeoMDQL para que se possam realizar consultas geográficas e multidimensional. Apesar de ser tão simples quando a MDX, a GeoMDQL não pode ser considerada uma linguagem destinada a usuários não especialistas, visto que deve-se conhecer sua gramática e sintaxe para realizar consultas corretamente.

2.6 Mapwarehouse

Em [17], Sampaio *et. al* propõem um modelo de dados lógico multidimensional que integra estreitamente dados espaciais ao ambiente de *Data Warehousing* (vide Figura 2.13).

Com isso, o metamodelo prevê a modelagem de esquemas de DW espacial contendo tanto dimensões espaciais como medidas espaciais.

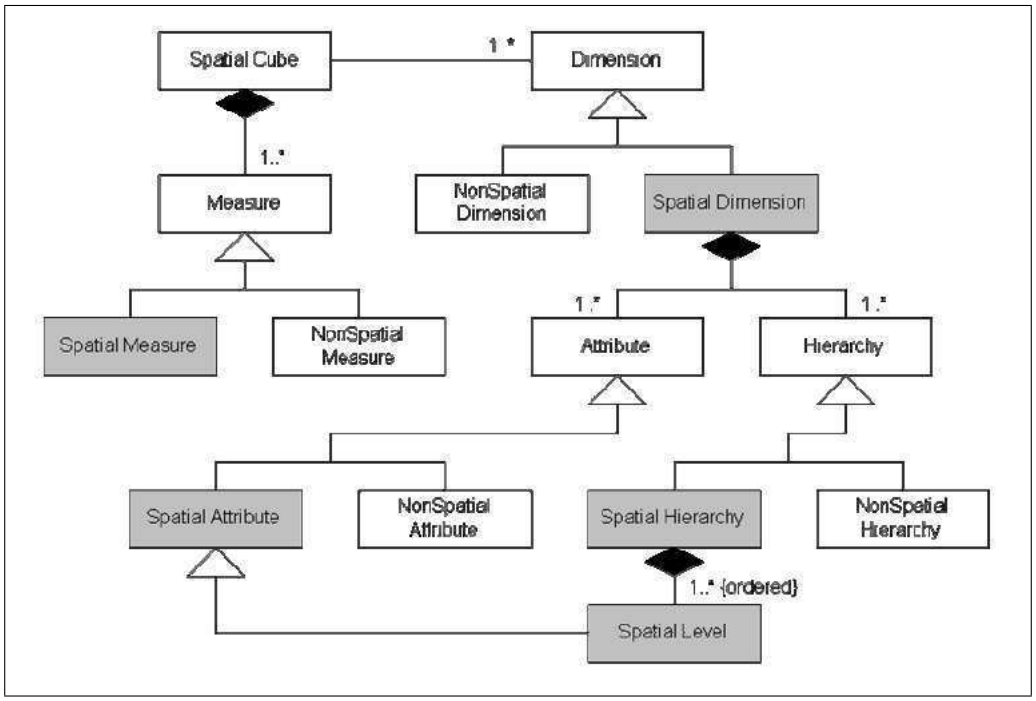

Figura 2.13: Metamodelo Espacial Multidimensional

Para validar o modelo lógico proposto, desenvolveu-se um protótipo de ferramenta SOLAP, denominado Mapwarehouse, o qual utiliza o SGBD Oracle e sua extensão espacial, Oracle Spatial. O protótipo desenvolvido implementa um estudo de caso referente à distribuição agrícola do Estado da Paraíba. O esquema do estudo de caso é composto pelas seguintes dimensões: **Tempo**, **Localização** (dimensão espacial), **Precipitação**, **Solo** e **Tipo de plantação**, além da medida numérica **Quantidade plantada** e da medida espacial **Área plantada**.

A interface gráfica do protótipo Mapwarehouse foi projetada visando permitir a definição de suas consultas analítico-espaciais a partir de menus e janelas interativas. A partir de cliques na interface o usuário pode definir suas consultas e submetê-las ao processador de insultas da aplicação.

A Figura 3.2 ilustra a interface inicial do protótipo Mapwarehouse, na qual foi definida a seguinte consulta: *quais as áreas de plantação (medida espacial), e suas as respectivas quantidades (medida numérica), do estado da Paraíba por Microrregião e por Região (Roll-up do nível Microregião para o nível Região) do mês de Janeiro de 2003 até o mês de Maio*

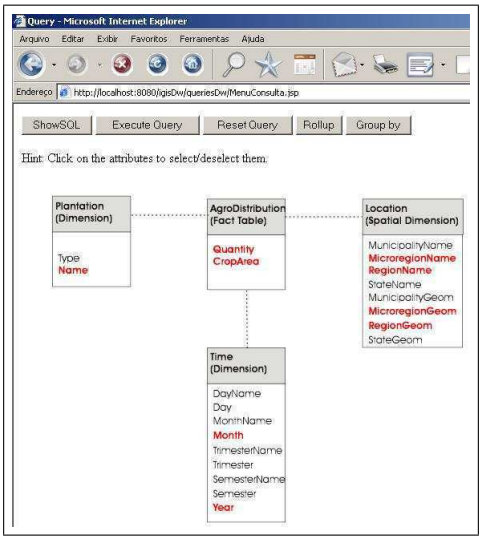

Figura 2.14: Interface de definição de consulta do Mapwarehouse

de 2003, mês a mês, que estejam dentro de uma janela espacial. Seu resultado pode ser observado na Figura 2.15.

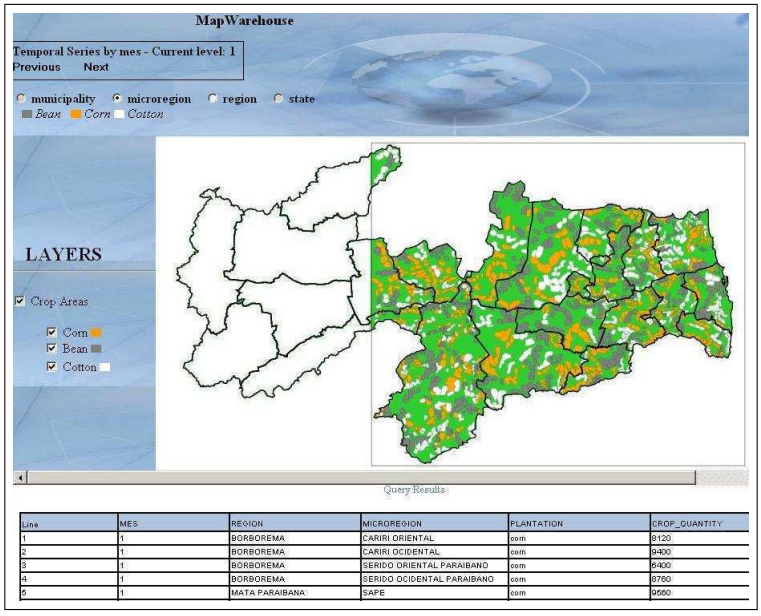

Figura 2.15: Exemplo de resultado de consulta no Mapwarehouse

Uma característica do protótipo Mapwarehouse é que o mesmo depende totalmente do SGBD Oracle. Além disso, a interface gráfica do protótipo foi desenvolvida especificamente para o estudo de caso Distribuição Agrícola. Estes dois fatores tornam inviável sua utilização em outros estudos de caso e outros SGBDs.

2.7 Discussão

Neste capítulo foram apresentadas propostas recentes que buscam a integração das tecnologias OLAP e SIG. De maneira geral, estas propostas têm como objetivo definir e/ou implementar um ambiente de suporte à decisão mais poderoso que una o melhor das duas tecnologias. Contudo, ainda não existe um consenso por parte dos autores no que se refere a *Data Warehouse* Espacial: as propostas são divergentes tanto com relação aos conceitos básicos relacionados a DW espacial, como também do ponto de vista de implementação.

As principais características observadas nos trabalhos, além de outras características que julgamos serem importantes a um ambiente de DW espacial, estão sintetizadas na Tabela 2.1. A partir desta síntese, pode-se analisar comparativamente as propostas.

	JMap	GeoWOlap	SOVAT	Piet	GolapWare	Mapwarehouse
Abordagem Integrada		x			x	x
Medidas Espaciais	x	x		x	x	x
Medidas Complexas		x				
Operadores Espaciais					x	x
Interface Web	x	x			x	x
Consulta Conceitual	x	x	x			x
Definição de Esquemas					x	
Tecnologia Aberta	x				x	
Extensível - Domínio	x	x			x	
Extensível - SGBD	x					

Tabela 2.1: Características dos trabalhos relacionados

Observando a Tabela 2.1, percebe-se a diferença entre as propostas, tanto em aspectos relacionados a modelos de dados como também com relação a funcionalidades implementadas.

Do ponto de vista de modelo de dados, nota-se que apenas a proposta GeoWOlap [13], [15] utiliza medidas como objetos complexos. Modelar medidas como objetos complexos pode ser uma característica muito interessante, principalmente quando se utilizam medidas espaciais, visto que um objeto espacial tem um conjunto de informações inerentes a ele,

como por exemplo seu nome, sua área, perímetro, etc. Estas informações podem ser utilizadas na análise de negócio aprimorando o processo de tomada de decisão.

Apenas a GolapWare e o Mapwarehouse utilizam operadores espaciais para a realização de *Slice and Dice* no cubo de dados. Esta característica aumenta demasiadamente o poder da ferramenta SOLAP, visto que amplia o universo de consultas que podem ser realizadas.

Com relação à definição de esquemas, apenas a proposta do GolapWare apresenta uma ferramenta para este fim. Esta ferramenta se chama GeoDWCase - um *plug-in* do ambiente de desenvolvimento integrado Eclipse.

Nota-se ainda, que duas propostas utilizam tecnologias abertas, o que pode ser considerada uma vantagem no caso de uma implantação, visto que tecnologias abertas diminuem o custo de implantação, além de poderem ser modificadas ou estendidas.

Outra característica importante é a possibilidade de utilizar a ferramenta para diferentes domínios de aplicação. Neste sentido, apenas as propostas JMap, GeoWOlap e GolapWare aparentemente permitem esta tarefa. As outras três propostas foram concebidas para domínios específicos.

Por fim, apenas a proposta GolapWare afirma ser possível a utilização da ferramenta em diferentes SGBDs. Entretanto, deve-se observar que um conjunto de funcionalidades da ferramenta foram implementadas na forma de procedimentos armazenados do SGBD PostgreSQL. Por isso, sua utilização em outro SGBD que não seja o PostgreSQL, requer que estes procedimentos armazenados sejam reescritos para o novo SGBD utilizado.

De maneira geral, nenhuma das propostas de integração de OLAP e SIG pode ser considerada satisfatória, seja por questões de funcionalidades implementadas ou seja por conta de limitações dos modelos de dados que estas ferramentas implementam. A proposta que mais se aproxima deste objetivo é a GolapWare. Entretanto, a linguagem de consulta utilizada pela ferramenta (GeoMDQL) é textual, o que certamente dificulta sua utilização pelos não especialistas, pelo fato de ser necessário informar comandos na sintaxe da linguagem GeoMDQL para realização de consultas.

Capítulo 3

Uma Proposta de *Framework* para DW Espacial

Este capítulo tem como objetivo apresentar uma proposta de *framework* para desenvolvimento de aplicações de *Data Warehouse* Espacial destinadas a não especialista, isto é, usuários gestores. O principal objetivo do *framework* é que os gestores possam definir seus próprios cubos de DW Espacial e realizar consultas sobre eles. Para isto, o *framework* deve trabalhar em nível conceitual, no qual detalhes de implementação são abstraídos. Ressalta-se que aspectos relacionado ao ETL estão fora do escopo deste trabalho.

Assim como a proposta definida por Sampaio *et al* [17], nosso *framework* também se chamará **Mapwarehouse**. Entretanto, deve-se ressaltar que não se trata de uma melhoria do protótipo anterior, nosso Mapwarehouse é uma proposta nova.

O restante deste capítulo está dividido da seguinte forma: na seção 3.1 são apresentados os requisitos essenciais que devem ser atendidos pelo *framework* proposto. A seção 3.2 apresenta o projeto arquitetural do *framework* Mapwarehouse, o qual é definido a partir de módulos de *software* e relacionamentos entre os módulos. A seção 3.3 discorre sobre aspectos de implementação de duas extensões do *framework*: Oracle e PostgreSQL. Na seção 3.4 são apresentadas as linguagens gráficas de definição de esquemas de DW espacial e consulta a DW espacial. Por fim, a seção 3.5 traz uma breve discussão sobre o *framework* proposto.

3.1 Requisitos do *Framework*

Nesta seção serão apresentados os requisitos que guiaram o desenvolvimento do *framework* Mapwarehouse. Um destes requisitos, considerado essencial, é obviamente o requisito de ser um *framework*. Segundo Gamma et al [36], um *framework* é "um conjunto de classes cooperantes que constroem um projeto reutilizável para um determinada categoria de software". Em outras palavras, um *framework* dita a arquitetura de software, provendo uma estrutura que pode ser reutilizada no desenvolvimento de aplicações similares.

Frameworks são classificados segundo dois aspectos distintos: (i) com relação a como as particularidades da aplicação são introduzidas no *framework*, ou seja, "como são usados"e (ii) com relação ao tipo de aplicações que utilizam o *framework*, ou seja, "onde são usados"[37], [38]. Do ponto de vista do aspecto (i), um *framework* pode ser:

- Focado em Herança: *frameworks* desse tipo, também chamados de *white-box* ou *architecture-driven*, são estendidos a partir da definição de sub-classes e sobrescrita de métodos abstratos. Sua implementação é conhecida e pode ser modificada pelos usuários.

- Focado em Composição: estes *frameworks*, também chamados de *black-box* ou *data-driven*, possuem um conjunto de funcionalidades implementadas que não são visíveis aos usuários que os utilizam. Apenas suas interfaces são conhecidas.

- Híbrido: é uma combinação das características dos *frameworks white-boxes* e *black-boxes*. A maioria dos *frameworks* se enquadram nesta categoria.

Do ponto de vista do aspecto (ii), um *framework* pode ser de suporte, de domínio ou de aplicação [37]:

- *Frameworks* de suporte: provêm serviços em nível de sistema operacional, por exemplo, acesso a arquivos, *drivers* de dispositivos, etc.

- *Frameworks* de domínio: conhecidos como *frameworks* horizontais, encapsulam funcionalidades aplicáveis a uma gama de aplicações de domínios distintos. Por este motivo, a parte genérica do framework é mínima. Exemplos de *frameworks* horizontais são os destinados a desenvolvimento de interfaces gráficas.

- *Frameworks* de aplicação: conhecidos como *frameworks* verticais, encapsulam funcionalidades aplicáveis a aplicações de um domínio específico, por exemplo aplicações de geoprocessamento, aplicações de recursos humanos, etc.

No caso específico do *Mapwarehouse*, a "filosofia"de *framework* está na possibilidade de estendê-lo para que o mesmo seja utilizado com diferentes SGBDs para o desenvolvimento de aplicação SOLAP. O Mapwarehouse é considerado híbrido e vertical. Híbrido pelo fato de possuir um conjunto de classes que encapsulam funcionalidades e não podem ser alteradas (*back-box*) e ao mesmo tempo possui um conjunto de classes e métodos abstratos que devem ser implementados para contemplar características específicas de cada SGBD. Por outro lado, é considerado vertical pois é destinado apenas ao desenvolvimento de aplicações SOLAP.

O projeto arquitetural do Mapwarehouse foi baseado no uso de diferentes padrões de projetos (*Design Patterns*) que provêm reutilização de código e capacidade de extensão.

Outro requisito fundamental é que as aplicações desenvolvidas com o Mapwarehouse sejam destinadas especialmente aos gestores. Deve ser permitido a esses usuários definir e consultar um DW Espacial em alto nível de abstração. Para tornar isto possível, o Mapwarehouse implementa um novo metamodelo conceitual multidimensional, descrito na subseção seguinte.

3.1.1 Metamodelo do *framework*

O metamodelo do Mapwarehouse, ilustrado pelo diagrama UML da Figura 3.1, foi considerando duas características básicas:

- Integração: o metamodelo proposto integra totalmente dados espaciais e dados não espaciais ao DW. Com isso, pode-se modelar aplicações utilizando dados espaciais tanto em dimensões como em medidas.

- Simplicidade: o metamodelo considera um DW Espacial uma extensão natural de um DW convencional. Neste sentido, os conceitos do metamodelo são os mesmos dos tradicionais, ou seja, dimensões, hierarquias, fatos e medidas, sendo estes conceitos estendidos para lidarem com dados espaciais.

Segundo o metamodelo, um DW é composto por classes *Fact* e classes *Dimension* associadas por um relacionamento N:M (*linked_to*). Entretanto o relacionamento (*linked_to*)

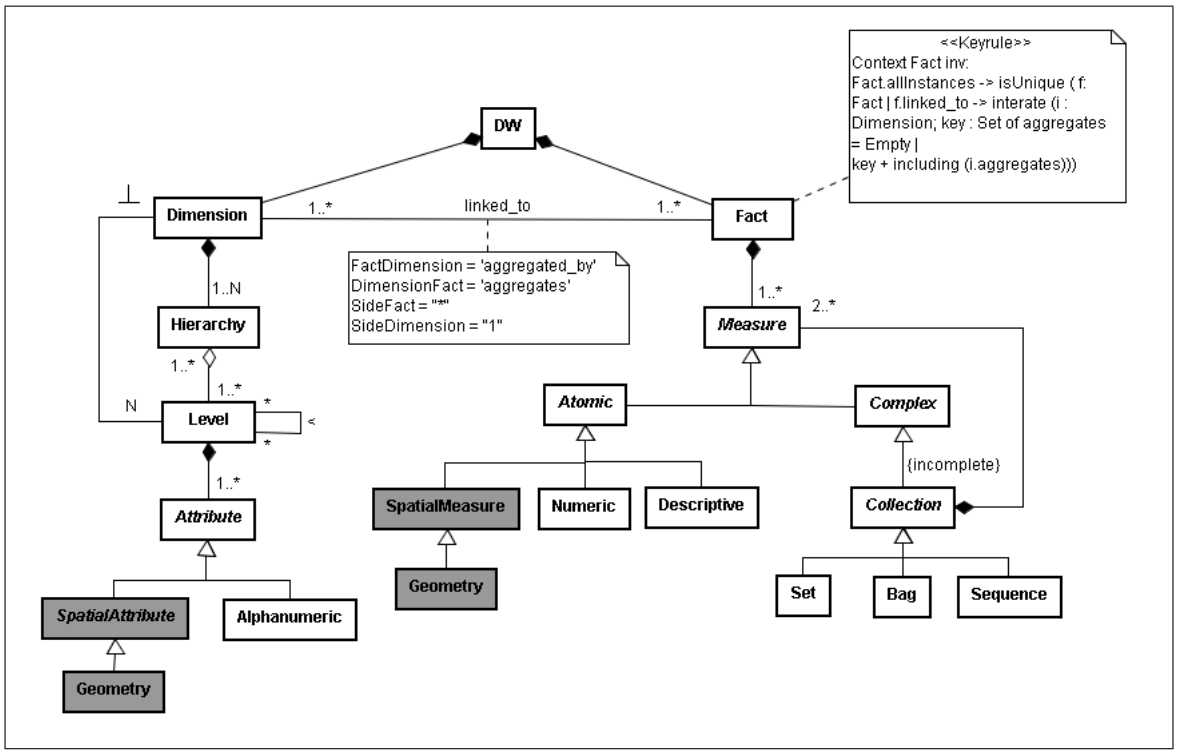

Figura 3.1: Metamodelo do Mapwarehouse

não captura toda a essência da associação entre as classes *Fact* e *Dimension*. Para cada par *Fact-Dimension* deve existir um relacionamento 1:N, *Dimension* agrega (*aggregates*) *Fact*, representando semanticamente que uma dimensão é um critério de agregação de medidas. Um conjunto de relacionamentos *Dimension aggregates Fact* forma a chave da classe *Fact*, a qual é formalizada pela expressão OCL (*Object Constraint Language*) da caixa de anotações.

Uma classe *Fact* é composta por uma ou várias classes *Measure*. Esta pode ser *Atomic* ou *Complex*, como segue:

- Classe *Atomic*: a classe *Atomic* pode ser *Numeric*, *Descritive* ou *SpatialMeasure*. Uma classe *Numeric* representa um medida numérica convencional enquanto uma classe *Descriptive* representa uma pseudo-medida que serve apenas para descrever textualmente outras medidas. A classe *SpatialMeasure* é implementada por uma classe *Geometry*, representando um dos elementos geométricos definidos pelo OGC (*Open Geospatial Consortium*) [39] (i.e. *Point, LineString, Polygon, MultiPoint, MultiLineString* ou *MultiPolygon*).

- Classe *Complex*: a classe *Complex* visa atender a outro requisito do *framework*, que é

o de permitir a definição de medidas como objetos complexos. Uma medida complexa pode ser uma coleção no sentido restrito, ou seja, representada por uma das subclasses *Set*, *Bag* ou *Sequence*, ou uma coleção no sentido amplo, quando alguma de suas medidas não é subclasse de *Collection*. Esta semântica está representada pela restrição {*incomplete*}. Em outras palavras, uma medida complexa não é atômica, mas não é necessariamente uma coleção (*set, bag* ou *sequence*).

Uma classe Dimenson é composta por pelo menos uma ou N classes *Hierarchy*. Por sua vez, uma classe *Hierarchy* contém uma ou várias classes *Level* com um auto-relacionamento M:N. O símbolo < indica que classes *Level* formam relacionamentos do tipo "parte-de"entre si, ou seja, um nível pode compor outros níveis. Se uma *Dimension* tiver N > 1 classes *Hierarchy*, a própria classe Dimension é o nível mais baixo de todas as hierarquias. Isto é representado a partir do símbolo _|_, que indica o papel da classe *Dimension* no relacionamento 1:N com a classe *Level*.

A classe *Level* é composta por classes *Attribute*. Uma classe *Attribute* pode ser *Alphanumeric*, representando atributos convencionais, ou *SpatialAttribute*, representado por um dos elementos geométricos do OGC.

Nota-se que o metamodelo do Mapwarehouse é uma extensão simples de um metamodelo de DW convencional. No metamodelo, os conceitos de fato e dimensão foram estendidos para lidarem com dados espaciais. Em outras palavras, a única inovação do modelo são as classes *SpatialMeasure* e *SpatialAttribute*(destacadas na Figura 3.1) que representam elementos geométricos.

Além dos requisitos apresentados relacionados ao modelo de dados, os seguintes requisitos funcionais guiaram o desenvolvimento do *framework*:

- Definição de Medidas: refere-se a todas as funcionalidades relacionadas a atividade de definição de medidas, como: criar nova medida, alterar medidas existentes, excluir medidas, etc.

- Definição de Dimensões: refere-se a todas as funcionalidade relacionadas a atividade de definição de dimensões, como: criar nova dimensão, alterar dimensões existentes, excluir dimensões, além de funcionalidades relacionadas às hierarquias das dimensões,

como: criar nova hierarquia, adicionar nível, alterar ordem dos níveis de determinada hierarquia, excluir hierarquias, etc.

- Mapeamento conceitual-lógico: deve ser possível mapear um esquema definido a partir das interfaces gráficas do *framework* para um esquema lógico de algum SGBD específico.

- Definição de Consulta: refere-se a todas as atividades relacionados à definição de consultas. Deve ser possível definir consultas a partir da interface gráfica de maneira intuitiva, sem a necessidade de conhecimento de linguagens lógicas de consulta (e.g. SQL).

- Salvar Consulta: tem como objetivo armazenar consultas definidas pelos usuários para posterior utilização, gerando um repositório de relatórios gerenciais.

- Restrição espacial: o *framework* deve possibilitar o uso de operações espaciais (e.g. *touches, intersects, etc.*), visando aumentar a abrangência das consultas executadas.

- Operação OLAP Espacial: deve ser possível navegar nas hierarquias, mesmo que as hierarquias contenham dados espaciais.

- Sincronismo entre Tabelas e Mapas: a apresentação de resultados de consultas deve ser feita em tabelas e mapas de forma sincronizada. Uma ação em uma tabela deve ser refletida no mapa e vice-versa.

- Funções espaciais: deve ser possível a manipulação de mapas a partir de funções espaciais como *zoom in*, *zoom out* e *pan*, visando uma melhor visualização de resultados de consultas.

Foram considerados também os seguintes requisitos não funcionais:

- Uso de tecnologias abertas: um requisito não funcional essencial é a utilização de tecnologias abertas, o que permite que outros desenvolvedores possam estender o *framework*, além de diminuir o custo de implantação.

- Interface Web: a proposta é que o *framework* seja totalmente voltado para Web, permitindo que usuários possam acessar remotamente suas aplicações a partir de um navegador.

- Abordagem Integrada: o *framework* deve ser implementado segundo a abordagem integrada de DW espacial.

3.2 Arquitetura

Em nível macro, a arquitetura do *framework* Mapwarehouse divide-se em três camadas: Cliente, Aplicação e *Data Warehouse*, como ilustrada na Figura 3.2. A seguir, serão descritas cada uma destas camadas.

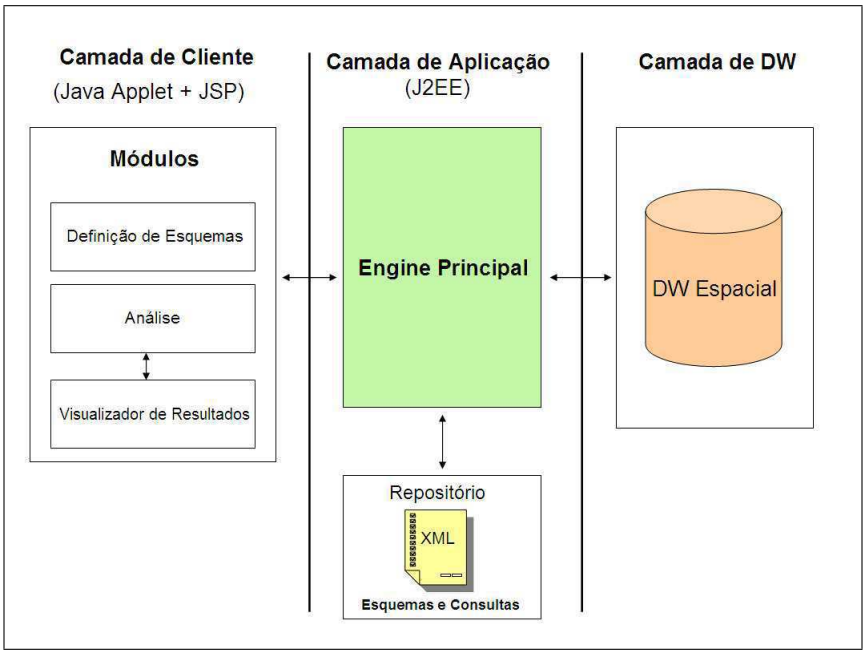

Figura 3.2: Macro Arquitetura do Mapwarehouse

- **Camada de Cliente:** a camada de cliente compreende um conjunto de interfaces gráficas com as quais usuários podem definir seus esquemas de DW espacial e realizadas consultas a *Data Warehouses* espaciais criados a partir destes esquemas. Estas interfaces são agrupadas em três módulos distintos: (i) módulo de definição de esquemas, que permite a definição de esquemas conceituais de DW Espacial e o mapeamento desses esquemas conceituais para esquemas lógicos de um SGBD específico; (ii) módulo de análise, que permite que consultas conceituais sejam definidas e executadas; (iii) módulo visualizador de resultados, que é composto por interfaces para apresentação de mapas e tabelas.

Ressalta-se que o *framework* Mapwahouse tem como requisito a utilização de tecnologias abertas, além da necessidade de ser voltado para Web. Por isso, utilizou-se as tecnologias Java Applet e Java Server Pages em seu desenvolvimento. Maiores detalhes sobre as interfaces gráficas dos módulos de definição e análise são apresentados na seção 3.4.

- **Camada de Aplicação:** esta camada é considerada a principal na arquitetura do Mapwarehouse. É nela que está inserido o *engine* principal responsável por todo o funcionamento da ferramenta. O *engine* principal serve como *interface* entre a camada do cliente e o *Data Warehouse*. Assim como a camada de cliente, esta camada foi implementada na tecnologia Java. Detalhes sobre o *engine* principal são apresentados na subseção seguinte.

- **Camada de *Data Warehouse*:** a última camada da arquitetura é a camada de DW. No *Data Warehouse* estão armazenados os dados analíticos e espaciais utilizados no processo de tomada de decisão. O *framework* é independente de SGBD, sendo possível a utilização de diferentes tecnologias e fabricantes. Entretanto, deve ser lembrado que um dos requisitos não funcionais do Mapwarehouse é utilizar a abordagem integrada de DW, na qual dados analíticos e espaciais são armazenados em um mesmo ambiente. Por isso, é essencial que o SGBD escolhido tenha a capacidade de armazenar e manipular dados espaciais.

3.2.1 *Engine* Principal

O engine do Mapwarehouse, ilustrado na Figura 3.3, é o componente principal na arquitetura do Mapwarehouse. Ele é o responsável pelo processamento e geração de resultados de consultas definidas pelos usuários, sendo a interface entre a camada de cliente e o *Data Warehouse*. O engine é composto por um conjunto de módulos e submódulos, descritos a seguir.

Controlador

O módulo Controlador é a interface entre as camadas de cliente e aplicação, sendo responsável por receber solicitações de clientes e retornar resultados. Quando um cliente faz uma

solicitação, o gerenciador de operações identifica o tipo de requisição solicitada e delega sua execução para um módulo responsável por atendê-la. Dentre os tipos de operações atendidas estão, as operações relacionadas à definição de esquemas e consultas (e.g. salvar esquemas, alterar consultas, etc), as operações de manipulação de mapas (e.g. *zoom in*, *zoom out*, pan), a operação de mapeamento de esquema e as operações relacionadas à execução de consultas em geral.

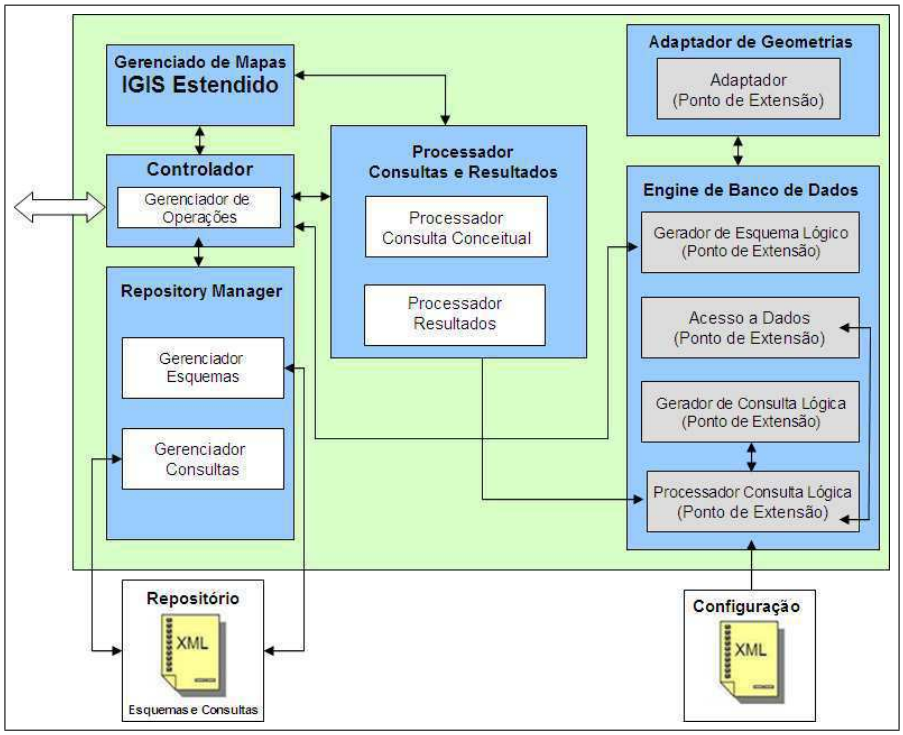

Figura 3.3: Engine Principal

Gerenciador de repositório

O Gerenciador de repositório divide-se em dois submódulos: **Gerenciador de Esquemas** e **Gerenciador de Consultas**. O Gerenciador de Esquemas recebe delegações do Módulo Controlador para realização de operações relacionadas à definição de esquemas, por exemplo: abrir esquema, salvar esquema, adicionar dimensão, alterar dimensão, etc. Por outro lado, o Gerenciador de Consultas gerencia as consultas definidas pelos usuários para que possam ser utilizadas em ocasiões futuras. Ambos, esquemas e consultas definidas são armazenadas em arquivos XML (*Extensible Markup Language*) em um repositório em disco localizado no servidor de aplicação.

Processador de Consultas e Resultados

O Processador de Consultas e Resultados é responsável pelo processamento e geração de resultados de consultas solicitadas pelos usuários. Quando o módulo Controlador identifica que determinada operação refere-se à execução de consulta, ele encaminha a solicitação para o Processador de Consultas e Resultados, que identifica qual o tipo de consulta a ser executada e aciona outros módulos que serão utilizados na sua execução. Este módulo é composto por dois submódulos descritos a seguir.

- **Processador de Consulta Conceitual**: cada consulta solicitada pelo usuário chega ao Processador de Consultas e Resultados no formato conceitual (descrito na subseção 3.4). Cabe ao Processador de Consulta Conceitual realizar um pré-processamento na consulta, formatando-a para que seja encaminhada ao Módulo Engine de Banco de Dados, que é o responsável pela sua execução propriamente dita.

- **Processador de Resultados**: este submódulo é responsável pela formatação dos resultados de uma consulta. Uma vez realizado o processamento e execução de uma consulta, deve-se formatar os resultados que serão apresentados aos usuários, como por exemplo, atribuir cores a elementos de mapas e gráficos.

Basicamente, o Mapwarehouse atende quatro tipos de consultas:

- Consulta somente Tabular: são consultas OLAP que não dados espaciais, consequentemente não existe a geração de mapas. Neste tipo de consulta, o processador interage apenas com o *Engine* de Banco de Dados (descrito mais adiante) e o resultado é representado em forma de tabela.

- Consulta somente com Mapas: são consultas espaciais características dos Sistemas de Informação Geográfica, como por exemplo: "Retornar os bairros que estão adjacentes à filial X". Para resolver este tipo de consulta, o processador interage tanto com o *Engine* de Banco de Dados como também com o módulo Gerenciador de Mapas (descrito mais adiante), responsável pela geração dos mapas que serão apresentados aos usuários.

- Consulta Mista: é o tipo mais complexo de consulta que um usuário pode solicitar. É a combinação entre a consulta tabular e a consulta com mapas, ou seja, envolve tanto dados analíticos como dados espaciais. Os resultados de consultas mistas são representados em Mapas e Tabelas.

- Consulta com Série Temporal: consultas com série temporal envolvem a dimensão temporal do DW como critério de agregação de medidas. Um exemplo de consulta com série temporal seria: "retornar o total de vendas das filiais, por cada ano". Quando utilizam-se dados espaciais em uma consulta com série temporal, o resultado pode ser a geração de vários mapas - um para cada elemento da série. É papel do Processador de Consultas e Resultados identificar que uma solicitação refere-se a uma série temporal e formatar os resultados da consulta de maneira que possa ser apresentada ao usuário de forma satisfatória.

Gerenciador de Mapas

O Gerenciador de Mapas tem como objetivo gerar mapas sob demanda e aplicar funções espaciais (e.g., *zoom in*, *zoom out* e *pan*) sobre eles. É acionado tanto pelo Módulo Processador de Consultas e Resultados como pelo Módulo Controlador. A interação com o Processador de Consultas e Resultados acontece quando um usuário qualquer solicita a execução de determinada consulta. Nestes casos, o Processador de Resultados transfere um conjunto de dados espaciais e numéricos para o Gerenciador de Mapas, que os utiliza para gerar o mapa que representa o resultado da consulta. A interação com o Módulo Controlador é mais simples: uma vez gerado o mapa representando o resultado de determinada consulta, pode-se manipular este mapa a partir de funções espaciais como *zoom in*, *zoom out* e *pan*. Neste caso, as operações são delegadas diretamente pelo Módulo Controlador.

Em sua implementação foi utilizada uma versão estendida do IGIS - um *framework* Web para desenvolvimento de SIGs [40]. O modelo de dados e o engine de geração de mapas são basicamente os mesmos da versão original do IGIS. Entretanto, algumas alterações foram feitas, como por exemplo, a geração de gráficos sobrepostos a mapas e o formato de arquivo gerado pelo engine, que deixou de ser SVG (*Scalable Vector Graphics*) e passou a ser PNG (*Portable Network Graphics*). Além disso, foram reimplementados os mecanismos

para realização de operações espaciais (e.g. *Zoom in*, *Zoom out* e *Pan*) visando um melhor desempenho.

3.2.2 Pontos de Extensão

Os outros dois módulos do *engine* principal, o Adaptador de Geometrias e o Engine de Banco de Dados, são os pontos de extensão do *framework* Mapwarehouse. Pontos de extensão, também conhecidos com "ganchos", permitem que o *framework* seja estendido para atender a determinado requisito. No caso do Mapwarehouse, este requisito é a possibilidade de utilização de qualquer SGBD que lide com dados espaciais.

Os pontos de extensão do Mapwarehouse foram projetados seguindo o padrão IoC (*Inversion of Control*) [41], o que permite que programadores os implementem sem a necessidade de alteração do código do *framework*.

Adaptador de Geometrias

O objetivo do Adaptador de Geometrias é converter objetos de classes Java que representam geometrias em um SGBD específico em objetos de classes que possam ser manipulados pelo Mapwarehouse. Este ponto de extensão assim como sua implementação para os SGBDs Oracle e PostgreSQL também existem no *framework* IGIS, o que permitiu o reaproveitamento no Mapwarehouse. Maiores detalhes sobre a implementação de adaptadores são apresentados na seção 3.4.

Engine de Banco de Dados

Outro ponto de extensão do *framework* é o módulo *Engine* de Banco de Dados. Este módulo implementa basicamente três funcionalidades: a comunicação com a camada de *Data Warehouse*, o mapeamento de consultas conceituais em consultas lógicas e a execução de consultas lógicas em um SGBD específico. Cada uma dessas funcionalidades é desempenha por submódulos, descritos a seguir:

- **Acesso a Dados**: este submódulo tem como único objetivo prover acesso ao banco de dados em que encontra-se o *Data Warehouse*, sendo utilizado pelo submódulo Pro-

cessador de Consultas, para que este possa se comunicar com o SGBD e executar consultas sobre ele.

- **Gerador de Esquema Lógico**: o Gerador de Esquema Lógico realiza o mapeamento de esquemas de DW definidos pelos usuários em esquemas lógicos para algum SGBD específico.

- **Gerador de Consulta Lógica**: o Gerador de Consulta Lógica realiza o mapeamento de consultas conceituais em consultas lógicas, que são na realidade comandos SQL. Apesar de realizar a única tarefa de mapeamento conceitual-lógico, este ponto de extensão é o que demanda maior custo de implementação, visto que para implementá-lo o programador deve lidar com funções implícitas de cada SGBD que deseja ser utilizado com o Mapwarehouse.

- **Processador de Consultas**: quando recebe uma consulta conceitual, este submódulo interage com o Gerador de Consulta Lógica para convertê-la em uma consulta lógica. A partir daí, utiliza o submódulo Acesso a Dados para se comunicar com o SGBD e executar a consulta sobre ele. Após a execução da consulta, o processador transfere os resultados para o módulo Processador de Consultas e Resultados para que os resultados sejam formatados e apresentados ao usuário. Além de executar as consultas solicitadas pelos usuários, este módulo desempenha outras funcionalidades auxiliares como, por exemplo, a geração de *buffer* de geometrias. Maiores detalhes sobre este e os outros pontos de extensão são apresentados na seção 3.4.

3.3 Interface Gráfica Conceitual

Metamodelos de dados conceituais, como o modelo do Mapwarehouse ilustrado na Figura 3.1, fornecem um conjunto de elementos que permitem aos usuários representarem esquemas de aplicações em alto nível de abstração[42]. Estes modelos são amplamente utilizados na modelagem inicial de sistemas computacionais, pois abstraem detalhes de implementação e focam apenas no conceitos necessários para sua representação semântica. Por outro lado, linguagens gráficas permitem a comunicação entre usuário e sistema de forma não textual, ou seja, a partir de interação com interfaces gráficas de usuário (GUI - *Graphical User Interface*)

[43]. É neste contexto que se inserem as linguagens de definição de esquemas e definição de consultas do Mapwarehouse, as quais são consideradas gráficas e conceituais, pois são definidas por um conjunto de interfaces gráficas que permitem a definição de esquemas e consultas segundo metamodelos conceituais.

Esta seção está dividida da seguinte forma: a subseção 3.3.1 discorre sobre a linguagem de definição de esquemas *framework*, enquanto a subseção 3.3.2 apresenta a linguagem de definição de consultas.

3.3.1 Linguagem Gráfica Conceitual para Definição de Esquemas

No ambiente do Mapwarehouse, a tarefa de definição de esquemas compreende as subtarefas de definição de medidas, definição de dimensões e definição de hierarquias de dimensões. A seguir, serão apresentadas cada uma destas subtarefas.

Definição de Medidas

Segundo o metamodelo do Mapwarehouse, uma medida pode ser definida como Atômica (*Atomic*) ou Complexa (*Complex*). Uma medida atômica pode ser numérica ou espacial, sendo esta última representada por um dos elementos geométricos definidos pelo Open Geospatial Consortium. A Figura 3.4 apresenta a interface para definição de medidas atômicas. Os tipos de dados permitidos são: *Integer, Real, Point, LineString, Polygon, MultiPoint, MultiLineString e Multipolygon*.

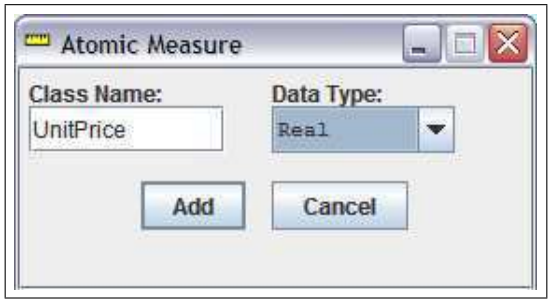

Figura 3.4: Interface para definição de Medida Atômica

Por outro lado, uma medida complexa pode ser um conjunto de outras medidas (inclusive outras medidas complexas) ou um coleção no sentido restrito (i.e. *Set*, *Bag* ou *Sequence*).

A Figura 3.5 (1) ilustra um exemplo de definição de medida complexa composta pelas medidas Submeasure1 (Inteiro) e Submeasure2 (Polígono), enquanto a Figura 3.5 (2) ilustra a definição de uma coleção (sequência de Inteiros).

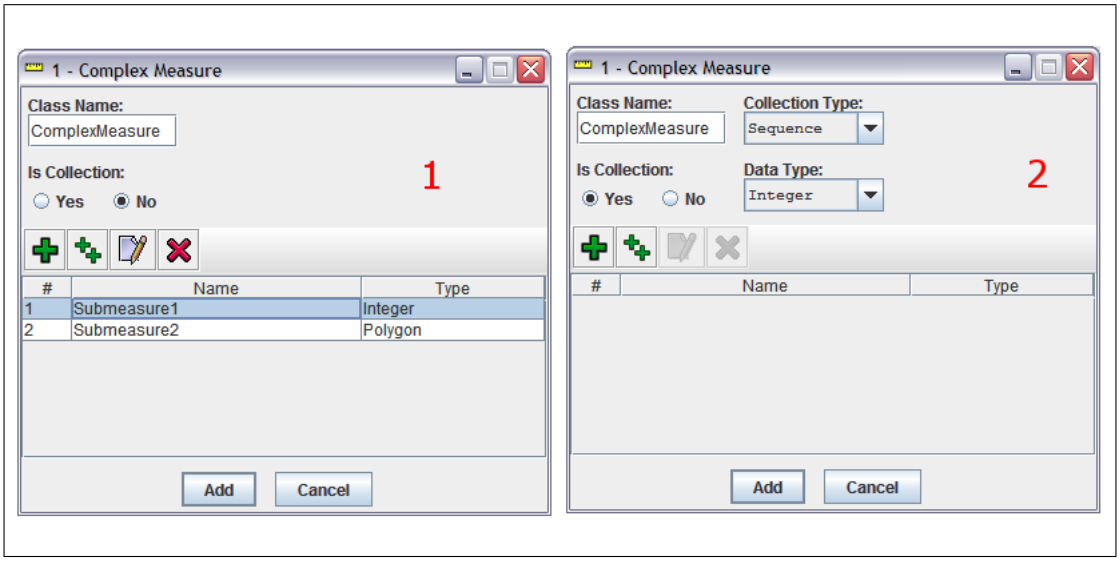

Figura 3.5: Interface para definição de Medidas Complexas

Definição de Dimensões

Segundo o metamodelo do Mapwarehouse, uma dimensão (classe *Dimension*) é composta por um conjunto de níveis (classes *Level*), organizados em pelo menos uma ou mais hierarquias (classes *Hierarchy*). Por sua vez, uma classe *Level* é composta por uma ou mais classes *Attribute*, que podem ser alfanuméricos ou espaciais.

No ambiente do Mapwarehouse, dimensões são definidas a partir de um *Wizard* composto por três etapas. A Figura 3.6 apresenta a interface gráfica da primeira etapa da definição da dimensão *Location*, na qual são informadas o nome da classe *Dimension* (*Location*), uma descrição e os atributos que compõem o primeiro nível da hierarquia (i.e. a própria dimensão). No caso específico do exemplo da Figura 3.6, são informados os atributos *Name* e *Geometry*, que representam respectivamente o nome de determinado local e a geometria que representa este local.

A etapa seguinte à definição da classe *Dimension* é a adição de novas classes *Level*. Esta etapa é ilustrada na Figura 3.7, que representa a adição dos níveis Bairro (*District*) e Cidade (*City*) à dimensão *Location*.

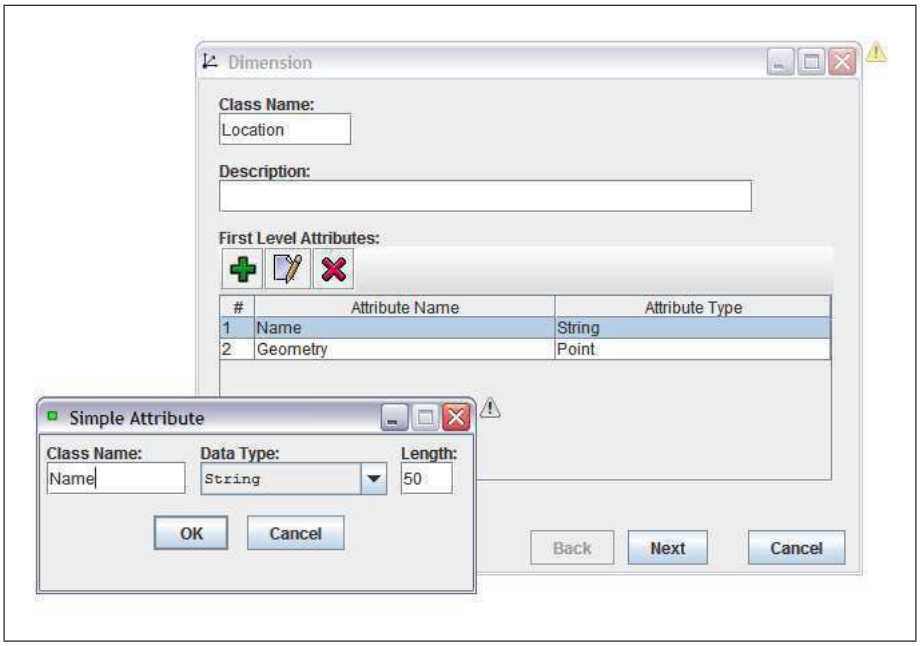

Figura 3.6: *Wizard* para definição de Dimensões

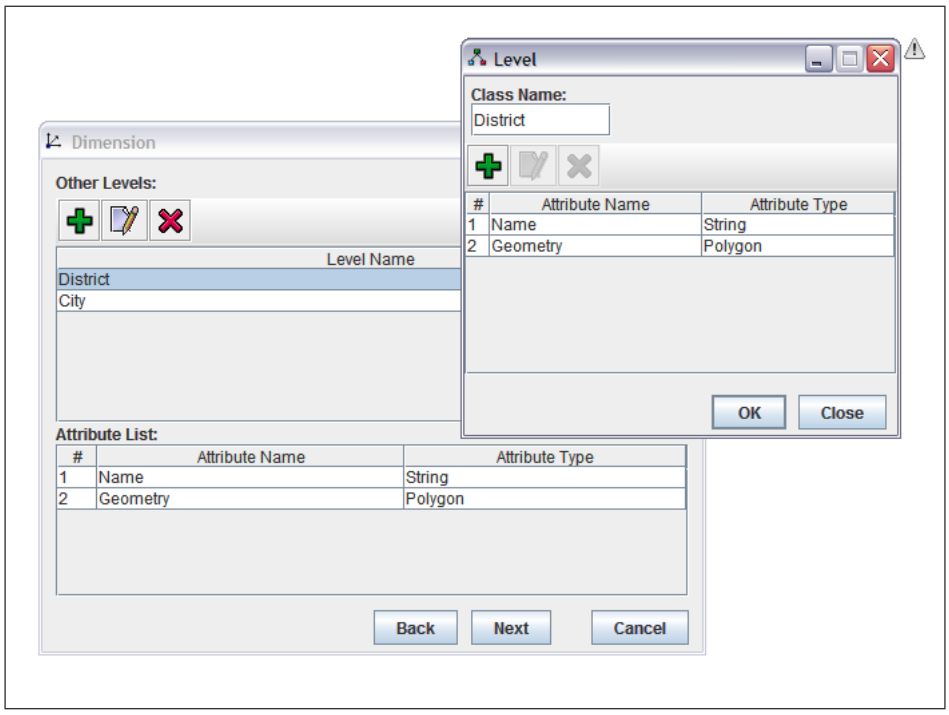

Figura 3.7: Adicionando níveis à dimensão

A etapa seguinte, ilustrada na Figura 3.8, permite a definição de hierarquias. Para facilitar esta tarefa, o Mapwarehouse pressupõe que cada nova classe *Level* adicionada a determinada dimensão faz parte de uma mesma hierarquia. Isto permite que uma hierarquia seja definida de forma automática à medida que novos níveis são adicionados. Entretanto, ressalta-se que esta hierarquia gerada automaticamente pode ser alterada, assim como novas hierarquias podem ser adicionadas à dimensão.

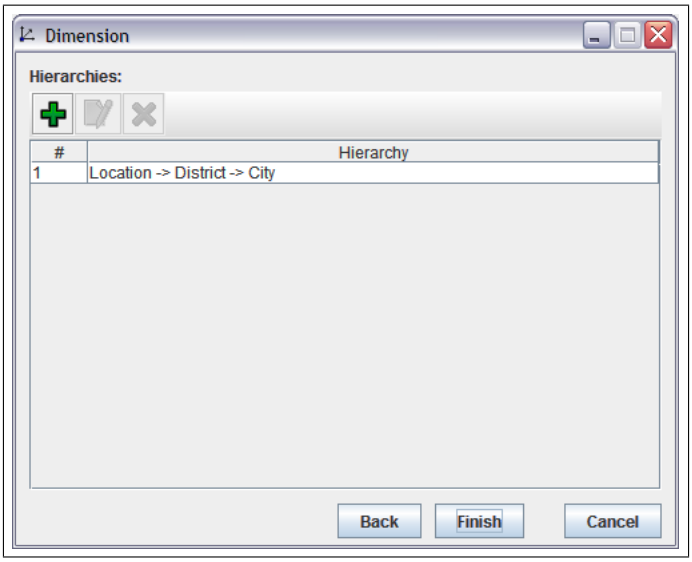

Figura 3.8: Interface de definição de hierarquias

A alteração de hierarquias, assim como a adição de novas hierarquias, é realizada a partir de uma interface gráfica especial, ilustrada na Figura 3.9. Esta interface permite organizar os níveis de determinada hierarquia e visualizá-la em um diagrama conceitual. No exemplo específico da Figura 3.9, observa-se as hierarquia da dimensão *Location*.

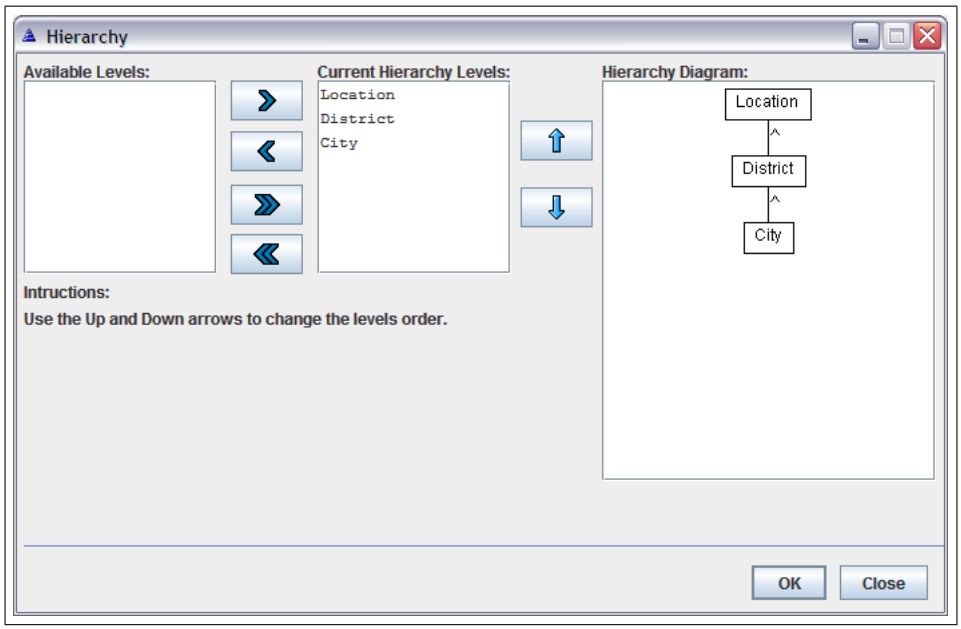

Figura 3.9: Interface de definição de hierarquias

Um esquema definido a partir das interfaces gráficas pode ser visualizado em uma estrutura em forma de árvore como a ilustrada na Figura 3.10. A partir desta árvore e de botões da barra de ferramentas é possível adicionar, alterar ou remover elementos do esquema. Além disso, um dos botões da barra de ferramentas permite a geração de um esquema lógico, resultado do mapeamento conceitual-lógico entre o esquema conceitual definido e um SGBD específico utilizado com o Mapwarehouse. Os detalhes do mapeamento conceitual-lógico serão apresentados no capítulo 4, referente às extensões do *framework*.

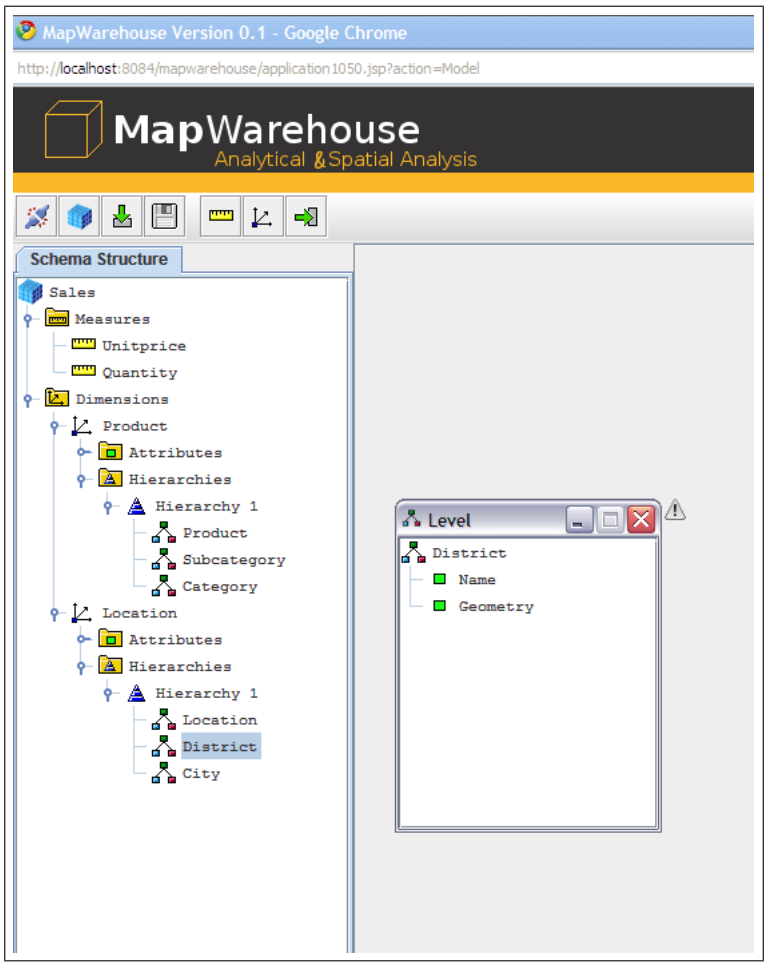

Figura 3.10: Estrutura representando o esquema

3.3.2 Linguagem Gráfica para Consulta

Apesar de serem consideradas padrão para realização de consultas em banco de dados, as linguagens de consulta de nível lógico (e.g. SQL) não são indicadas a usuários não especialista, pois requerem conhecimento significativo para que estes especifiquem suas consultas. Por outro lado, as linguagens de consulta de nível conceitual são ideais para estes usuários, pois abstraem detalhes de implementação.

A linguagem de consulta do Mapwarehouse é implementada a partir de um conjunto de interfaces gráficas, com as quais usuários podem definir consultas conceituais. Estas consultas conceituais são definidas segundo o metadomodelo ilustrado na figura 3.11. A seguir, serão descritas as classes do metamodelo, assim como as interfaces associadas a estas classes.

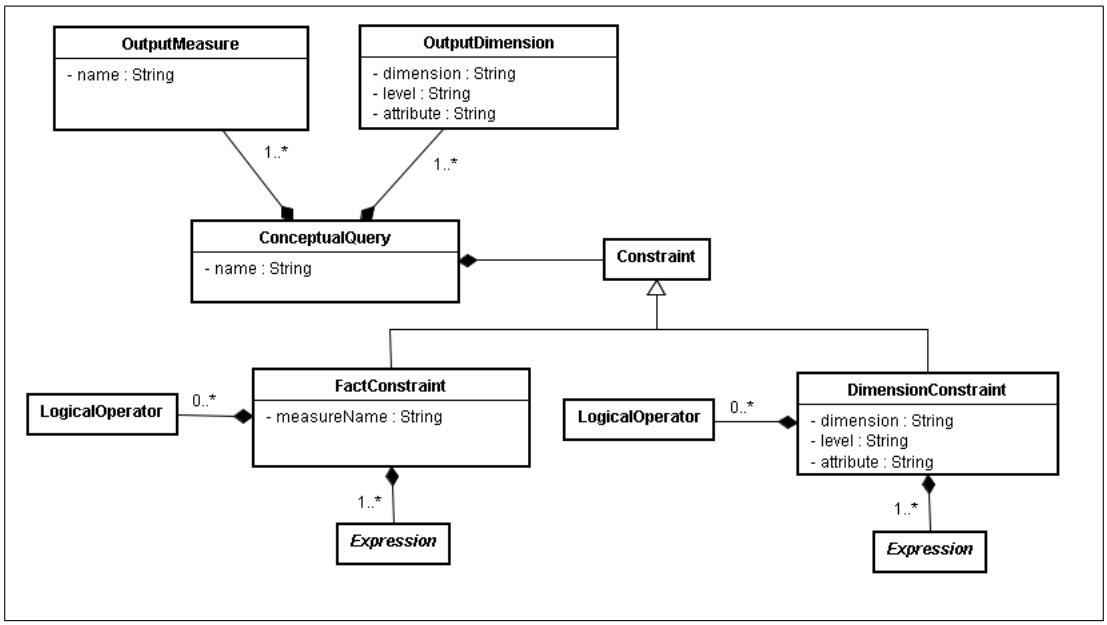

Figura 3.11: Diagrama UML - Consulta Conceitual

A classe *ConceptualQuery*, que representa uma consulta conceitual, é composta por classes *OutputDimension*, *OutputMeasure* e *Constraint*. As classes *OutputDimension* e *OutputMeasure* representam respectivamente, os atributos de dimensões e as medidas que serão exibidas no resultado de consultas. Além disso, uma *ConceptualQuery* é composta por classes *Constraint* quem representam restrições que serão aplicadas a membros de dimensões ou a fatos.

Constraint especializa-se em *FactConstraint* ou *DimensionConstraint*. Classes *FactConstraint* restringem fatos enquanto *DimensionConstraint* restringem membros de dimensões. Ambas, *FactConstraint* e *DimensionConstraint* são compostas por classes *Expression* e classes *LogicalOperation*.

Classes *Expression* representam expressões simples formadas por operadores e operandos. A Figura 3.12 (A) ilustra as subclasses de *Expression* que compõe *DimensionConstraint*, enquanto a Figura 3.12 (B) ilustra as subclasses de *Expression* que compõe *FactConstraint*. A seguir, serão descritas de forma sucinta cada subclasse de *Expression*.

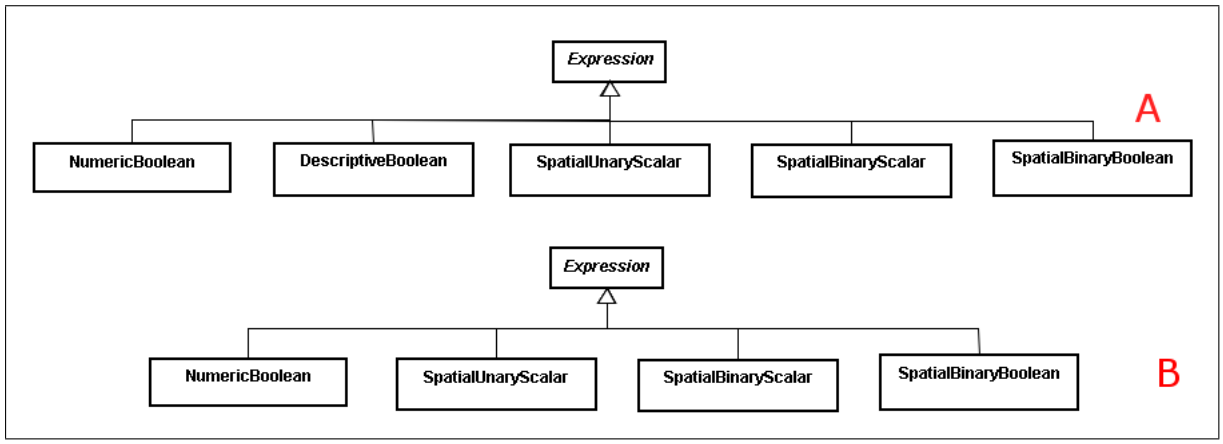

Figura 3.12: Tipos de Expressões

DescriptiveBoolean

A classe *DescriptiveBoolean* representa uma expressão simples aplicada a um membro alfanumérico de determinada dimensão. Uma expressão *DescriptiveBoolean* tem a forma: <Descriptive-Boolean> ::= <Operando> <Operador> <Valor>, onde:

- <Operando>: representa um atributo de determinado nível de uma dimensão.
- <Operador>: é um dos seguintes operadores relacionais: de igualdade (=), diferença (<>) ou tipo (*Like*).
- <Valor>: é um valor literal.

Exemplo: Filial.Nome = 'Bodocongó'.

NumericBoolean

A classe *NumericBoolean* representa uma expressão simples utilizada para restringir tanto membros numéricos de determinada dimensão como também medidas numéricas. Uma expressão *NumericBoolean* tem a forma: <Numeric-Boolean> ::= <Operando> <Operador> <Valor>, onde:

- <Operando>: representa um atributo de determinado nível de uma dimensão.
- <Operador>: um dos seguintes operadores relacionais: maior (>), menor (<), maior ou igual (>=), menor ou igual (<=), de igualdade (=), diferença (<>) ou o operador de conjunto (*IN*).
- <Valor>: é um valor literal numérico. No caso de o <Operador> ser o operador de conjunto (*IN*), <Valor> é um conjunto de valores literais separados por vírgula.

Exemplo: Ano IN {2008, 2009}.

SpatialBinaryBoolean

A classe *SpatialBinaryBoolean* representa uma expressão utilizada para restringir tanto membros espaciais de dimensões como também medidas espaciais. Uma expressão *SpatialBinaryBoolean* é aplicada entre dois elementos geométricos, tendo a forma: <SpatialBinary-Boolean> ::= <Operando-Espacial1>.<Operador>(<Operando-Espacial2>), onde:

- <Operador>: é um dos seguintes operadores espaciais: *Disjoint*, *Equals*, *Touches*, *Contains*, *Covers*, *Overlap*, *Inside*, *Covered By*.
- <Operando-Espacial1>: representa um atributo espacial de determinada dimensão ou uma geometria formada pela agregação de medidas espaciais.
- <Operando-Espacial2>: é uma geometria.

Exemplo: Filial.Geometria.Inside(Geometria).

SpatialUnaryScalar

A classe *SpatialUnaryScalar* representa uma expressão utilizada para restringir membros espaciais de dimensões ou medidas espaciais, sendo aplicada em um único elemento geométrico. Uma expressão *SpatialUnaryScalar* tem a forma: <SpatialUnary-Scalar> ::= <Operando-Espacial>.<Função> <Operador> <Valor>, onde:.

- <Função>: é uma função espacial aplicada a uma geometria que retorna um escalar. Pode ser a função *Length*, que retorna o tamanho de uma geometria, ou *Area*, que retorna a área de uma geometria.
- <Operando-Espacial>: é um atributo espacial de determinada dimensão ou uma geometria formada pela agregação de medidas espaciais.
- <Operador>: é um dos seguintes operadores relacionais: maior (>), menor (<), maior ou igual (>=), menor ou igual (<=), de igualdade (=), diferença (<>) ou o operador de conjunto (*IN*).
- <Valor>: é um valor literal numérico.

Exemplo: Filial.Geometria.Area >= 100.

SpatialBinaryScalar

A classe *SpatialBinaryScalar* representa uma expressão aplicada a dois elementos geométricos a partir da função espacial *Distance*, que retorna um escalar. Uma expressão *SpatialBinaryScalar* tem a forma: <SpatialBinary-Scalar> ::= <Operando-Espacial1>.Distance(<Operando-Espacial2>) <Operador> <Valor>, onde:

- *Distance*: é uma função que retorna uma Distância (*Distance*) entre dois elementos geométricos.
- <Operando-Espacial1>: um atributo espacial de determinada dimensão ou uma geometria formada pela agregação de medidas espaciais
- <Operando-Espacial2>: é uma geometria.

- <Operador>: é um dos seguintes operadores relacionais: maior (>), menor (<), maior ou igual (>=), menor ou igual (<=), de igualdade (=), diferença (<>) ou o operador de conjunto (*IN*).

- <Valor>: é um valor numérico que representa a distância.

Exemplo: Filial.Geometria.Distance(Geometria) <= 5000.

Classes *Constraint* são compostas também por classes *LogicalOperation*, representam os operadores lógicos E (*AND*) e OU (*OR*). A combinação de classes *Expression* com classes *LogicalOperation* permite a formação de expressões mais complexas entre atributos de uma mesma dimensão. Por exemplo, a expressão: (Filial.Geometria.Distance(Geometria) <= 5000 AND Filial.Geometria.Area >= 100).

Interfaces Gráficas

O ponto inicial para a definição de uma consulta conceitual é a interface ilustrada na Figura 3.13, que representa um *Workspace* que permite visualizar uma consulta definida e acessar um conjunto de funcionalidades para sua definição. As funcionalidades acessíveis são: definição de dimensões de saída, definição de medidas de saída, restrições sobre dimensões e definição de operações OLAP. Cada uma dessas funcionalidades são implementadas por uma ou mais interfaces gráficas. A seguir, serão apresentadas as interfaces de cada funcionalidade, associando-as com classes do metamodelo das Figuras 3.11 e 3.12.

Definição de dimensões de saída

A interface da Figura 3.14 tem como objetivo definir quais atributos de determinada dimensão serão exibidos como saída no resultado de uma consulta. Cada atributo selecionado representa uma instância da classe *OutputDimension* do metamodelo da Figura 3.11. Especificamente na Figura 3.14 é ilustrada a seleção dos atributos *Name* e *Geometry*. Neste caso, os dois atributos selecionados serão exibidos no resultada da consulta e o nível *Location* servirá como critério para a agregação das medidas.

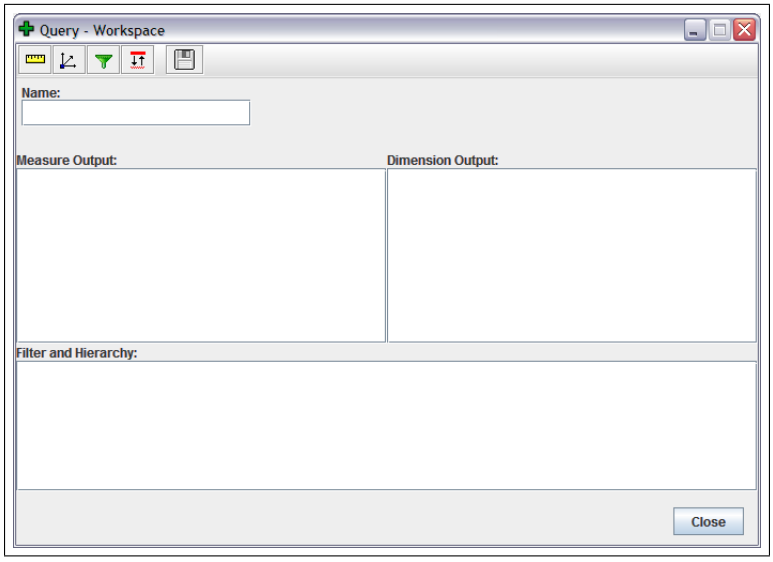

Figura 3.13: *Workspace* de definição de consulta

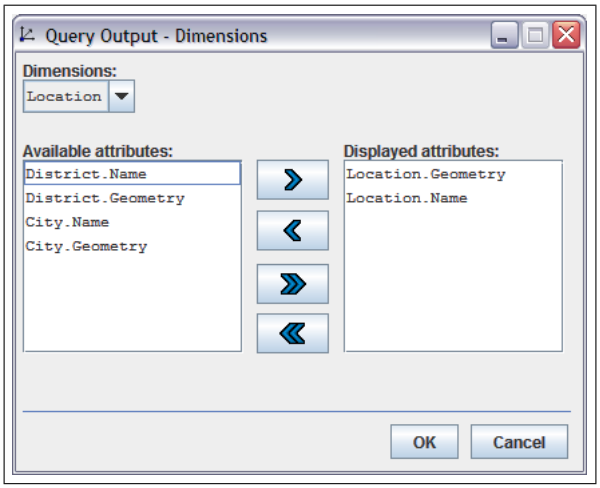

Figura 3.14: Definição de Saídas de Dimensão

Restrições sobre Dimensões

Restrições aplicadas a membros de dimensões, representadas por classes *Constraint* do metamodelo, são definidas pela interface gráfica da Figura 3.15. As restrições são expressões complexas formadas a partir da combinação de expressões simples com operadores lógicos. No caso ilustrado pela Figura 3.15, foi definida uma classe *Constraint* para a dimensão *Product*. A *Constraint* é composta por uma expressão *DescriptiveBoolean* (Product.Name = 'Corolla'). Restrições *DescriptiveBoolean* são comumente encontradas nas ferramentas OLAP tradicionais.

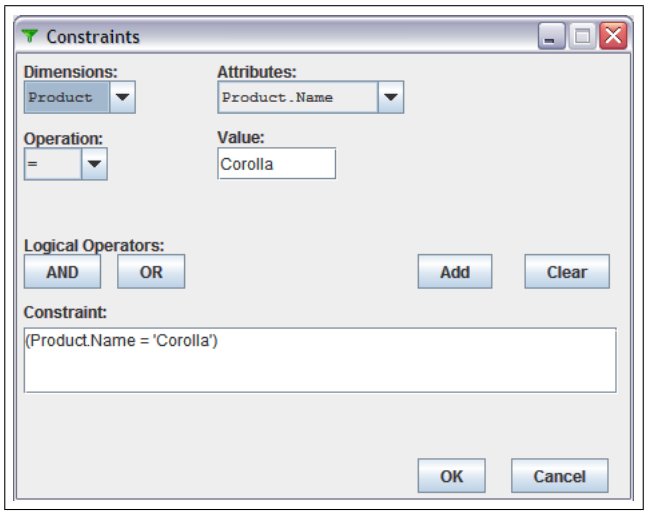

Figura 3.15: Definição Restrições de Dimensão

Por outro lado, a Figura 3.16 ilustra a definição de uma *Constraint* composta por uma expressão *SpatialBinaryBoolean*. Restrições como a *SpatialBinaryBoolean* envolvem a utilização de operações espaciais entre atributos espaciais das dimensões e outras geometrias. Para facilitar sua definição, o Mapwarehouse dispõe de uma interface gráfica chamada de interface de janela espacial (Figura 3.17). Esta interface gráfica permite selecionar uma geometria específica a partir de um clique do mouse sobre o mapa ou a definição de uma janela retangular (*cliping area*) definida a partir da seleção de dois da interface. No caso ilustrado na Figura 3.16, que mostra a restrição (Location.Geometry.Touches('Centro')), a geometria referente ao bairro Centro foi selecionada a partir de um clique no mouse sobre o bairro Centro. Outros detalhe sobre a utilização da janela espacial são apresentados no capítulo 4 desta dissertação.

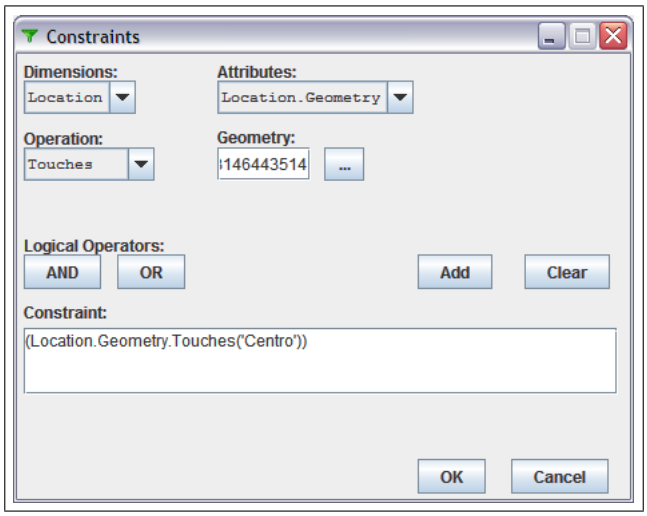

Figura 3.16: Definição Restrições Espacial em uma dimensão

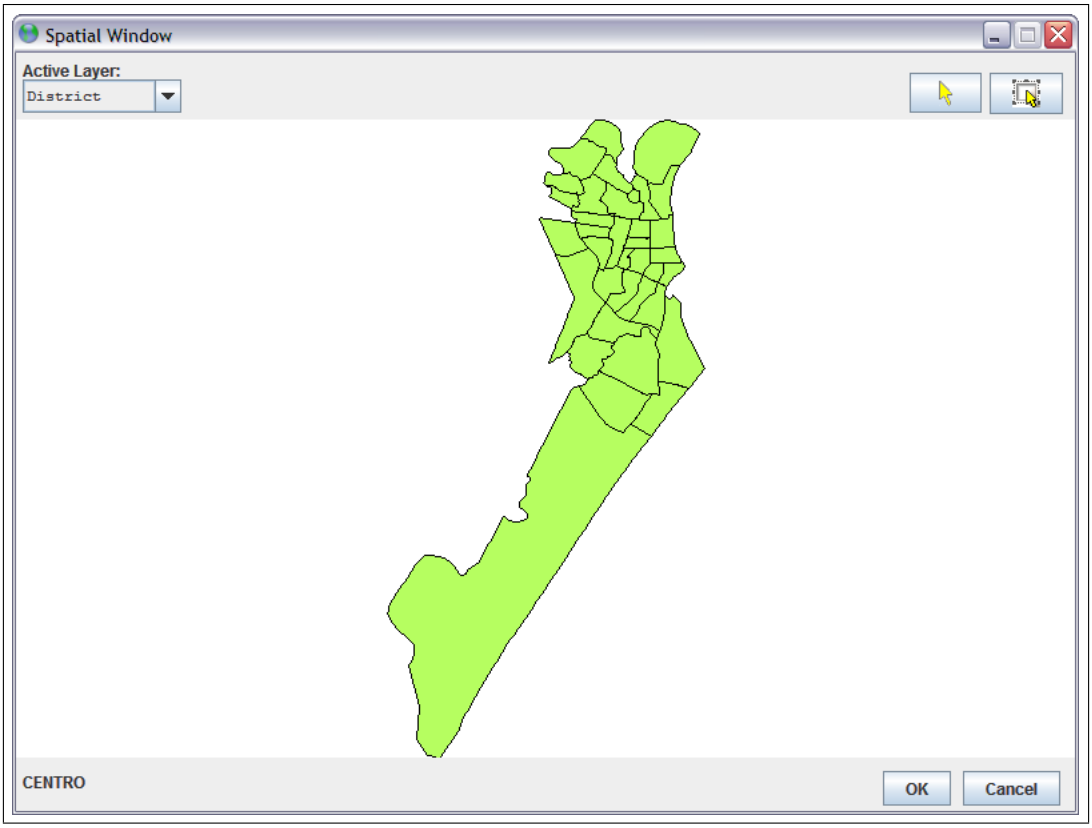

Figura 3.17: Interface de Janela Espacial

Definição de medidas de saída

A interface gráfica ilustrada na Figura 3.18 permite definir quais medidas serão exibidas no resultado de uma consulta. Além disso, dispõe de outras funcionalidades como: definir restrições aplicadas a fatos, critérios de ordenação de resultados e *ranking*. No que se refere à definição de restrições, esta interface tem as mesmas características da interface gráfica de definição de restrições sobre dimensões, ilustrada na Figura 3.15.

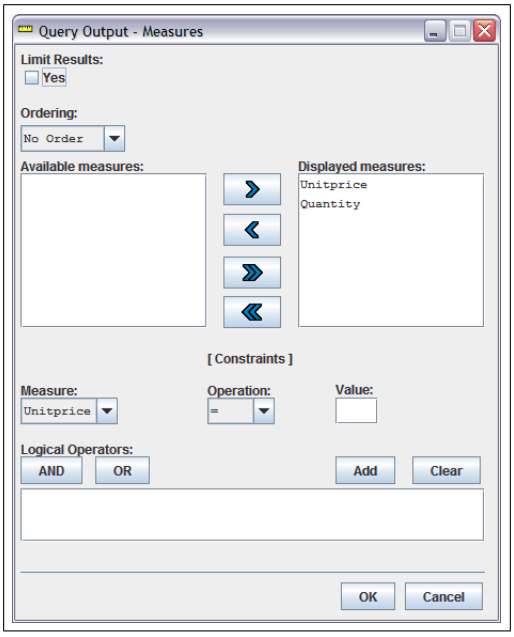

Figura 3.18: Definição de Saída de Medidas

Definição de Olap

A interface gráfica ilustrada na Figura 3.19 permite a definição as operações OLAP *Roll-up* e *Drill-Down*. Para cada dimensão do DW espacial é apresentado um diagrama representando suas hierarquias. A partir de cliques de mouse, pode-se escolher o tipo de operação OLAP e os níveis da hierarquia que serão utilizados no processo de agregação. Na Figura 3.19 é ilustrado uma operação de *Roll-up* do nível Localização (*Location*) até o nível Bairro *District*.

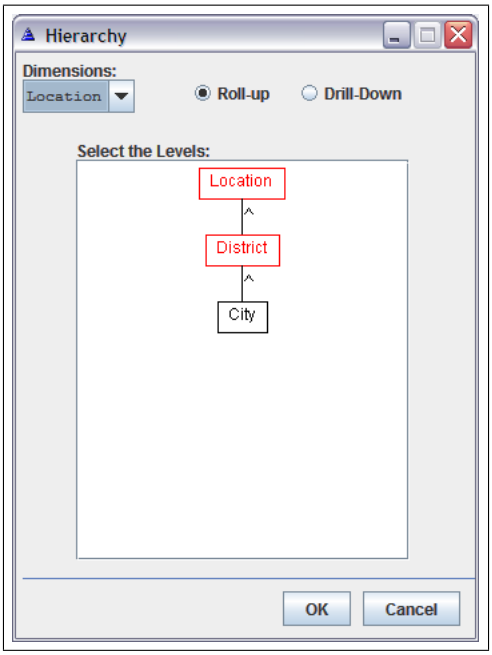

Figura 3.19: Definição de Olap

O Mapwarehouse permite também a realização de consultas com série temporal. Uma consulta com série temporal é uma consulta em que a dimensão temporal é utilizada como critério para a agregação de medidas. As séries temporais permitem que os fatos sejam agrupados e analisados comparativamente ao longo do tempo.

Estes tipos de consultas são comuns em ferramentas OLAP convencionais. Entretanto, do ponto de vista de ferramentas SOLAP que utilizam medidas espaciais, os resultados destas consultas devem ser apresentados aos usuários de maneira especial a partir de um conjunto de mapas, visto que a análise comparativa de medidas espaciais ao longo do tempo implica na geração de vários mapas, cada um representando um fato em determinado período de tempo.

Para auxiliar os usuários na visualização dos resultados de uma consulta com série temporal, o Mapwarehouse dispõe de um componente de interface gráfica como o ilustrado na Figura 3.20. Este componente permite que os usuários naveguem pela série temporal selecionando o critério de agregação temporal que foi utilizado para geração dos mapas. No exemplo da Figura 3.20, nota-se que o componente de navegação possui os itens "4/2009", "5/2009", "6/2009"e "7/2009", que representam respectivamente os meses Abril, Maio, Junho e Julho do ano de 2009. O clique do mouse em cada um desses itens separada-

mente, faz com que uma mapa seja exibido para o mês/ano selecionado.

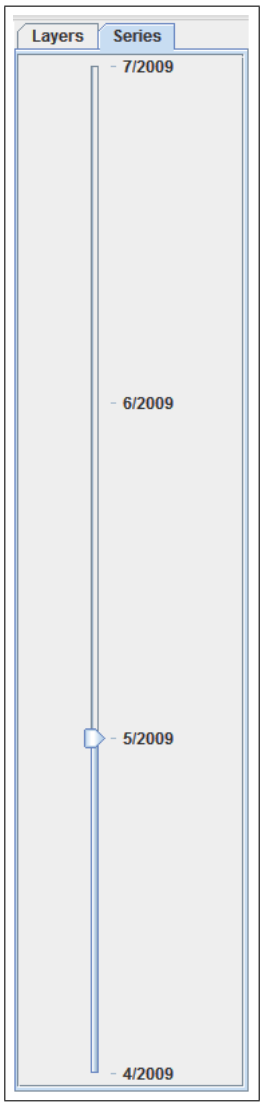

Figura 3.20: Componente de navegação na série temporal

Maiores detalhes sobre essa e outras funcionalidades e interfaces gráficas do *framework* Mapwarehouse serão apresentados no capítulo 4, que apresenta dois estudos de caso.

3.4 Implementação dos Pontos de Extensão

Esta seção apresenta detalhes sobre os pontos de extensão do Mapwarehouse e como eles devem ser implementados para que o *framework* possa ser utilizado com um SGBD específico. Além disso, serão apresentados detalhes de implementação de duas extensões do *framework* para SGBDs distintos: Oracle e PostgreSQL.

3.4.1 Processo de Extensão

Para utilizar o Mapwarehouse com um SGBD específico é necessário estender cada um dos pontos de extensão considerando as particularidades do SGBD. A seguir, será descrito de forma genérica como cada ponto de extensão deve ser estendido.

Extensão do Adaptador

Um adaptador é criado a partir da extensão da classe abstrata *AbstractAdapter* apresentada no diagrama UML da Figura 3.21. Cada método a ser implementado na classe concreta recebe como parâmetro um objeto que representa uma geometria de tipo geométrico específico (e.g. um ponto, um polígono, etc.) de algum SGBD e tem como retorno um objeto geométrico do modelo de dados do *framework* IGIS, utilizado pelo Mapwarehouse. Como por exemplo, o método **createPoint** da classe concreta *ConcreteAdapter*, que recebe como parâmetro um objeto representando um ponto segundo algum SGBD específico e tem como retorno um objeto Point que é instância da classe Point do modelo de dados do IGIS.

A implementação do método *creatPoint* pode ser observado na caixa de anotações do diagrama. Nota-se que um objeto Point da API de um SGBD qualquer (no exemplo: com.database.Point) foi convertido para um objeto Point do modelo IGIS.

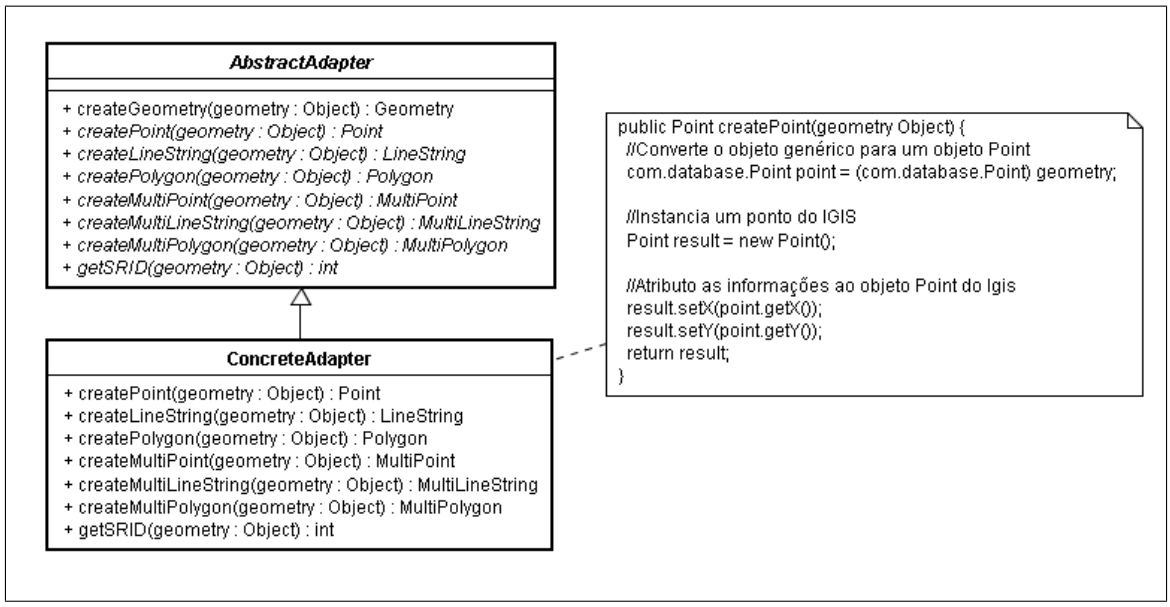

Figura 3.21: Adaptador de Geometrias

Extensão do Acesso a dados

Para que uma aplicação cliente se comunique com um servidor de banco de dados, é necessário o estabelecimento de uma conexão utilizando certos atributos, como por exemplo: *hostname* ou endereço IP, número da porta para comunicação, serviço, nome do esquema do banco de dados, etc. Alguns desses atributos são específicos em SGBDs de fabricantes diferentes. Por conta disso, foi necessário criar um ponto de extensão no Mapwarehouse para prover o acesso ao servidor de banco de dados em que se localiza o *Data Warehouse* Espacial.

O ponto de extensão de acesso a dados é criado a partir da extensão da classe *AbstractDataAccess* e implementação de seu método abstrato *getConnection*, observado na Figura 3.22. Nota-se que *AbstractDataAccess* possui dois atributos: *hostName* e *port*. Estes atributos são comuns a todos os fabricantes de SGBD e representam respectivamente o nome ou endereço IP do servidor de banco de dados e a porta utilizada para comunicação.

Os atributos que forem específicos do SGBD a ser utilizado com o Mapwarehouse devem ser adicionados como atributos da classe concreta que estende *AbstractDataAccess*. Isto é exemplificado na classe concreta *ConcreteDataAccess* ilustrada na Figura 3.22. Nota-se que *ConcreteDataAccess* contém dois atributos específicos (*specificAttribute1* e *specificAttribute2*).

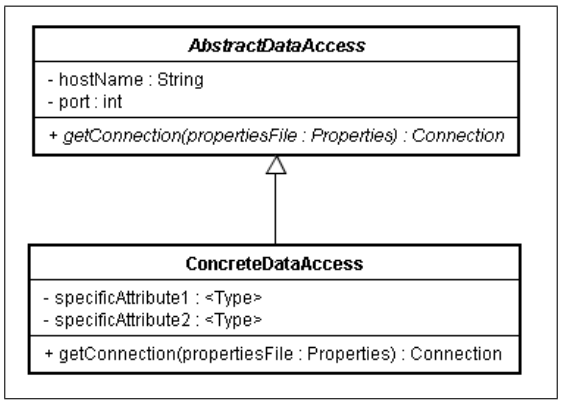

Figura 3.22: Acesso a Dados

O método *getConnection* que tem como objetivo retornar um objeto de conexão JDBC [44] com o banco de dados, recebe como parâmetro um objeto *Properties*. Um objeto *Properties* permite o acesso ao arquivo XML de configuração do Mapwarehouse (*database_config.xml*) que tem a forma como o arquivo apresentado no Código Fonte 3.1. Este arquivo deve possuir, além dos atributos *hostName* e *port*, todos os outros atributos específicos do SGBD que será utilizado com o Mapwarehouse. Quando o *framework* é inicializado, ele carrega as informações contidas neste arquivo para que o método *getConnection* as utilize para criar o objeto de conexão com o banco de dados.

Código Fonte 3.1: Arquivo XML de configuração

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
   <comment>Mapwarehouse Database Configuration File</comment>
   <entry key="hostName">value</entry>
   <entry key="port">value</entry>
   <entry key="specificAttribute1">value</entry>
   <entry key="specificAttribute2">value</entry>
</properties>
```

Extensão do Gerador de Esquema Lógico

O ponto de extensão gerador de esquema tem como objetivo mapear um esquema conceitual definido pelo usuário para um esquema lógico de um SGBD específico. Sua implementação é

realizada com a extensão da classe *AbstractLogicalSchemaGenerator* ilustrada no diagrama da Figura 3.23. Como observado no diagrama, *AbstractLogicalSchemaGenerator* possui apenas um método de acesso público (*generateSQL*) que recebe como parâmetro um objeto *SpatialCube*. Um *SpatialCube* representa um esquema de DW Espacial modelado segundo o metamodelo conceitual apresentado na seção 3.1.1, ou seja, um SpatialCube é composto por classes *Fact* e classes *Dimension*.

Pode-se observar na caixa de anotações do diagrama da Figura 3.23, a implementação do método *generateSQL*. Percebe-se que *generateSQL* invoca dois métodos abstratos: *generateDimensionsSQL* e *generateFactSQL*. Estes são os métodos que precisam ser implementados para gerar o esquema lógico.

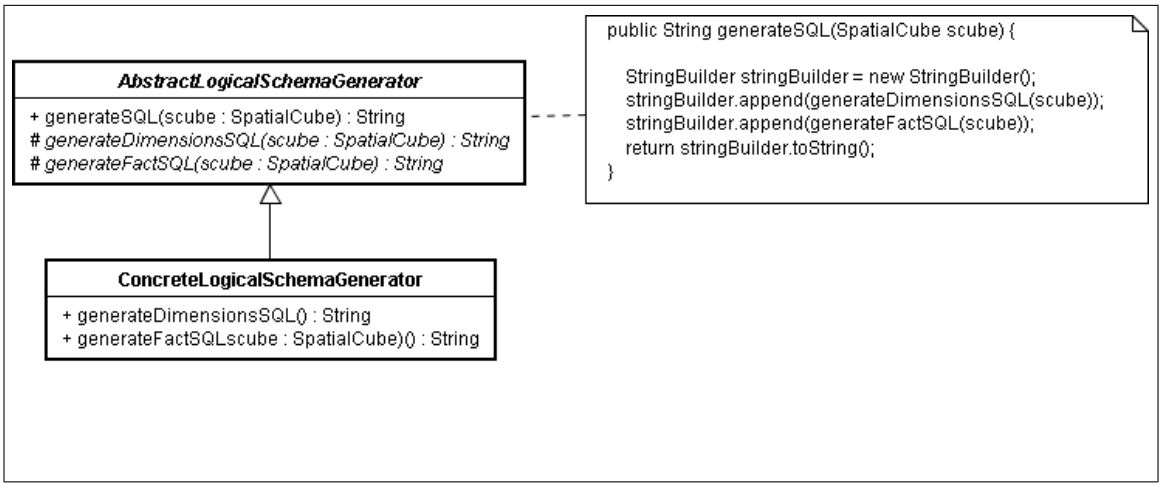

Figura 3.23: Gerador de Esquema Lógico

Como pode ser notado, este ponto de extensão foi definido de forma genérica. Isto ocorreu pelo seguinte motivo: um dos objetivos específicos desta dissertação é estender o *framework* para os SGBDs Oracle e PostgreSQL. Para a extensão Oracle, foi decidido utilizar o modelo de dados objeto-relacional. Dessa forma, a extensão Oracle foi implementada em termos de *Object Types* e *Object Tables*. Por outro lado, para a extensão PostgreSQL foi decidido utilizar o modelo de dados "quase relacional", sendo implementado em termos de tabelas relacionais convencionais. Chamamos de "quase relacional"haja vista a necessidade de esta extensão utilizar tipos e funções do PostGIS, que é objeto-relacional.

Extensão do Gerador de Consulta Lógica

O Gerador de Consulta Lógica é o ponto de extensão de maior complexidade de implementação, visto que é necessário conhecer funções específicas de cada SGBD e considerar a forma como o Gerador de Esquema Lógico foi implementado.

Para criar um Gerador de Consulta Lógica deve-se estender a classe *AbstractLogicalQueryGenerator* ilustrada na Figura 3.24. Pode-se observar que *AbstractLogicalQueryGenerator* possui um único método de acesso público *createSQL* que recebe como parâmetro uma consulta conceitual (*ConceptualQuery*) e retorna um objeto *SQLQuery*. Um objeto *SQLQuery* é composto por um atributo que representa o comando SQL referente à consulta conceitual, além de outros atributos utilizados para formatação do resultado da consulta.

O método *createSQL* foi implementado segundo o padrão de projeto *Template Method*. Um *template method* define o esqueleto de um algoritmo, postergando alguns passos para subclasses [36]. Estes passos são os métodos abstratos que devem ser implementados.

Figura 3.24: Gerador de Consulta Lógica

A seguir serão descritos os métodos abstratos que devem ser implementados em uma subclasse de *AbstractLogicalQueryGenerator*:

- *formatOutputDimension*: dada uma *OutputDimension* o método deve retornar uma cadeia de caracteres representando o atributo que será adicionado à cláusula *SELECT* do SQL a ser gerado.
- *formatOutputMeasure*: dada uma *Measure* o método deve retornar uma cadeia de caracteres que representa uma função de agregação sendo aplicada à medida. Se a medida for numérica utiliza-se a função de agregação *SUM*, caso seja espacial deve-se utilizar a função de união geométrica do SGBD específico.
- *getFrom*: retorna uma cadeia de caracteres que representam a cláusula *FROM* do SQL.
- *formatJoin*: retorna uma cadeia de caracteres que representam a junção natural das tabelas de dimensões com a tabela de fatos.
- *getGeometryByBoudingBox*: a partir de um objeto MBR composto por dois pontos, o método deve retornar uma cadeia de caracteres que representa um retângulo formado pelos dois pontos. Deve-se utilizar o construtor de um elemento geométrico específico do SGBD.

Os outros métodos de *AbstractLogicalQueryGenerator* representam as expressões definidas no metamodelo 3.12. Cada um desses métodos deve retornar uma cadeia de caracteres representando a expressão em termos de comandos SQL. Cada expressão deve considerar as operações e funções espaciais específicas do SGBD.

Extensão do Processador de Consultas

Como explicado anteriormente, é no Processador de Consulta que ocorre a execução propriamente dita de consultas solicitadas pelos usuários. Este submódulo do engine de banco de dados interage com Gerador de Consulta Lógica, que converte uma consulta conceitual em lógica, e a executa junto ao SGBD, gerando um conjunto de resultados *ResultSet*, que será formatado pelo Processador de Resultados para posterior apresentação ao usuário. Além

da execução da consulta e geração de resultados, o Processador de Consultas realiza outras tarefas auxiliares, como a geração de *buffer* (contorno) de geometria, conversão de uma geometria em uma cadeia de caracteres, entre outras.

Um Processador de Consultas é criado com a extensão da classe *AbstractQueryProcessor*, ilustrada na Figura 3.26. Uma classe concreta de *AbstractQueryProcessor* deve implementar cinco métodos abstratos. A descrição e a forma de implementação de cada um desses métodos são descritas a seguir:

- *getMapwarehouseGeometry*: este método visa converter um objeto não formatado (classe *Object*) em uma geometria do Mapwarehouse. Sua implementação deve ser feita da seguinte maneira: (i) deve-se converter o objeto não formatado para um objeto geométrico específico do SGBD; (ii) utilizar o Adaptador para converter o objeto geométrico do SGBD em um objeto do modelo de dados do IGIS. Esta dupla conversão é necessária pelo fato de os *Drivers* JDBC não terem sido concebidos para lidarem com dados espaciais, logo é necessário recuperá-los como objetos da classe *Object*, para em seguida convertê-los para objetos geométricos do SGBD, utilizando uma API específica que cada fabricante disponibiliza e logo após convertê-los novamente para o modelo de dados do IGIS.

- *getGeometries*: tem como objetivo retornar uma lista de geometrias do IGIS a partir de uma dimensão espacial do *Data Warehouse*. Deve-se executar uma consulta ao DW recuperando todos os atributos do nível e dimensão especificados e utilizar o método *getMapwarehouseGeometry* para convertê-los.

- *getGeoIdentifier*: retorna um objeto da classe *GeoIdentifier*, ilustrado na Figura 3.25, a partir de dois pontos selecionados em uma interface gráfica de janela espacial (exemplificada no capítulo 4). O objetivo é identificar qual atributo espacial de determinado nível de uma dimensão foi selecionado pela interface gráfica para que seja utilizado em consultas envolvendo operações espaciais.

- *getBufferGeometry*: tem como finalidade gerar uma geometria representando o *buffer* (contorno) de um elemento geométrico. Esta geometria gerada é utilizada para destacar o resultado de consultas que envolvam a função espacial de Distância (*Distance*).

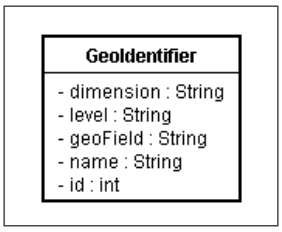

Figura 3.25: Classe GeoIdentifier

Algumas consultas com a função *Distance* são apresentados no Capítulo 4. A implementação deste método deve ser feita da seguinte forma: deve-se realizar uma consulta ao DW para selecionar a geometria identificada pelos atributos de GeoIdentifier e aplicar a função *Buffer* do SGBD, considerando o raio **dist** passado com parâmetro.

- *convertToString*: deve retornar uma cadeia de caracteres que representa o elemento geométrico identificado pelo *GeoIdentifier*. Esta cadeia de caracteres é utilizada em consultas envolvendo operações espaciais.

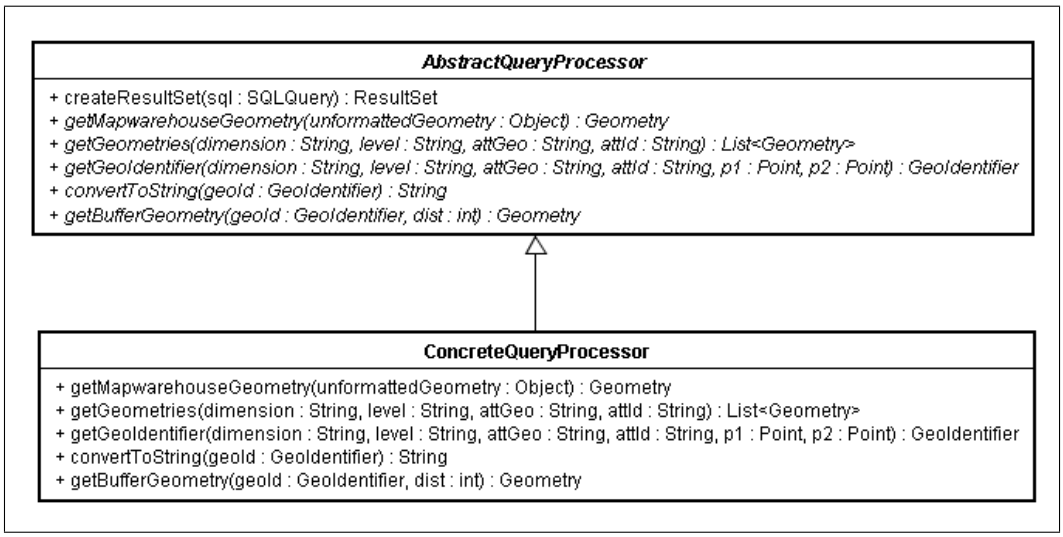

Figura 3.26: Processador de Consulta

O processo de extensão do Processador de Consultas é uma tarefa relativamente simples, visto que grande parte da lógica do processador já se encontra implementada na classe abstrata, o que torna necessário apenas que os cinco métodos auxiliares sejam implementados.

3.4.2 Conectando as extensões ao Mapwarehouse

Uma vez que foram implementados todos os pontos de extensão, é necessário informar ao Mapwarehouse que eles serão utilizados para o mapeamento de esquemas e consultas conceituais. Isto é feito com o auxílio da classe abstrata *AbstractDatabaseFactory* ilustrada na Figura 3.27. Esta classe instancia todos os pontos de extensão, sendo o meio utilizado pelos outros módulos do Mapwarehouse para acessar o Engine da Banco de Dados.

Cada novo SGBD que for "plugado"ao Mapwarehouse deve implementar uma subclasse concreta de *AbstractDatabaseFactory*, como a ilustrada na caixa de anotações da Figura 3.27. Além disso, deve ser adicionada uma nova entrada com a chave (*key*) *DatabaseFactory* no arquivo de configuração do *framework* (vide Código Fonte 3.2), informando qual classe o *framework* deverá instanciar para acessar o Engine de Banco de Dados.

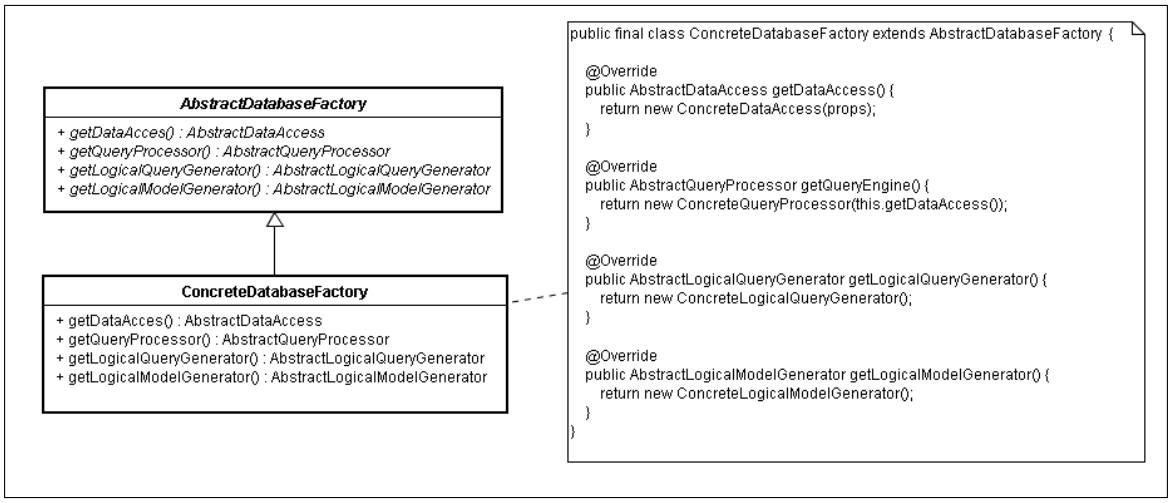

Figura 3.27: Fábrica

Código Fonte 3.2: Arquivo XML de configuração

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <comment>Mapwarehouse Database Configuration File</comment>
    <entry key="DatabaseFactory">mapwarehouse.database.concrete.
        ConcreteDatabaseFactory</entry>
    <entry key="hostName">value</entry>
    <entry key="port">value</entry>
    <entry key="specificAttribute1">value</entry>
```

```
    <entry key="specificAttribute2">value</entry>
</properties>
```

3.4.3 Oracle Spatial

Nesta seção e na seção 3.4.4 serão apresentados de forma resumida, aspectos importantes relacionados à implementação de alguns pontos de extensão do Mapwarehouse para os SGBDs Oracle com recurso Oracle Spatial e PostgreSQL com extensão PostGIS.

Adaptador

Na implementação do Adaptador de geometrias para o Oracle foi utilizada a API *Oracle Spatial Java Class Library* [25], que permite que aplicações Java manipulem e processem objetos espaciais armazenados no Oracle com recurso Oracle Spatial. A Figura 3.28 ilustra a implementação do método **createPoint** utilizando a API Oracle.

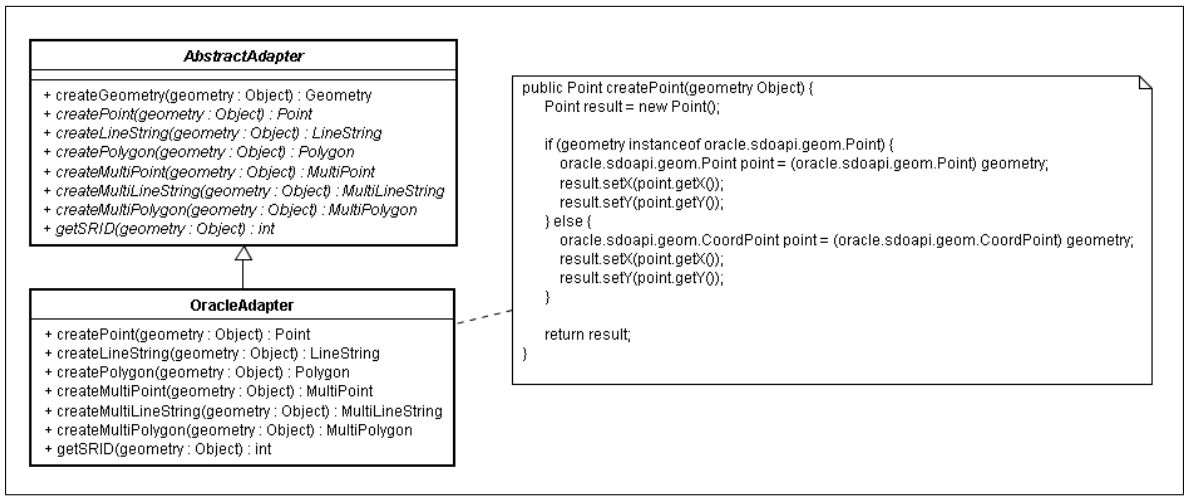

Figura 3.28: Adaptador - Oracle

Acesso a Dados

Como observado na Figura 3.29, além dos atributos *hostName* e *port*, que são comuns a todos os SGBD, foram adicionados à classe concreta *OracleDataAccess* os atributos: **sid** (*System Identifier*), **schema** (esquema a ser conectado) e **password** (senha para acesso).

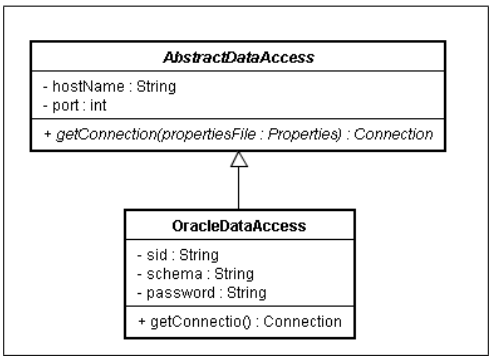

Figura 3.29: Acesso a dados - Oracle

Gerador de Esquema Lógico

A extensão do Oracle foi implementada segundo o modelo de dados Objeto-Relacional. Cada dimensão do esquema definido no Mapwarehouse é mapeada para uma *Object Table* de um *Object Type* específico. Cada *Object Type* é definido da seguinte forma:

CREATE TYPE <Fact_Name>.<TDimension_Name> **AS OBJECT** (<key_attribute>, <level_list>), onde:

- <TDimension_Name>: é o nome da dimensão.
- <key_attribute>: é um atributo que representará a chave primária da *Object Table*.
- <level_list>: é o conjunto de níveis das hierarquias da dimensão, cada nível na forma: **CREATE TYPE** <Fact_Name>.<TNivel_Name> **AS OBJECT** (<attribute_list>), onde:
  - <TNivel_Name>: é o nome do nível.
  - <attribute_list>: é o conjunto de atributos do nível. Cada atributo é mapeado para um tipo primitivo do Oracle.

Da mesma forma que as dimensões, o fato é mapeado para uma *Object Table* na forma:

CREATE TYPE <Fact_Name>.<TFact_Name> **AS OBJECT** (<lista_de_referências>, <lista_de_medidas>), onde:

- <TFact_Name>: é o fato.

- <lista_de_referências>: é o conjunto de referências aos objetos da classe dimensão. Cada referência é definida na forma:
  - <nome_dimensao> REF SCOPE IS <TDimension_Name>;
- <lista_de_medidas>: é o conjunto de medidas, sendo cada medida definida segundo o algoritmo do Pseudocódigo 3.3.

Código Fonte 3.3: Pseudocódigo para mapear medidas

```
if(medida for atômica) {
  //Realiza-se o mapeamento para um tipo primitivo do Oracle.
} else if(medida for complexa e uma coleção) {
  //A coleção é mapeada para uma tabela aninhada (Nested Table). A
      definição do tipo a ser armazenado na coleção segue este mesmo
      algoritmo.
} else {
  //Trata-se de uma medida complexa formada por medidas de tipo distintos
      . Neste caso, este algoritmo é repetido para cada medida que compõe
      a medida complexa.
}
```

Mapeamento de Operações e Funções Espaciais

A tabela 3.1 representa o mapeamento de funções e operações espaciais do Mapwarehouse para funções e operações do SGBD Oracle com recurso Oracle Spatial. Estas funções e operações foram utilizadas na implementação da classe OracleLogicalQueryGenerator, que estende *AbstractLogicalQueryGenerator*.

Função ou Operação do Mapwarehouse	Função ou Operação Oracle
Disjoint	SDO_OVERLAPBDYDISJOINT
Equals	SDO_EQUAL
Touches	SDO_TOUCH
Contains	SDO_CONTAINS
Covers	SDO_COVERS
Overlap	SDO_OVERLAPS
Inside	SDO_INSIDE
Covered_by	SDO_COVEREDBY
Length	SDO_LENGTH
Area	SDO_GEOM.SDO_AREA
Distance	SDO_WITHIN_DISTANCE

Tabela 3.1: Funções e Operações Espaciais - Oracle

3.4.4 PostgreSQL - PostGIS

Para a implementação dos pontos de extensão para o SGBD PostgreSQL foram considerados o modelo de dados relacional e a extensão espacial PostGIS.

Adaptador

Assim como na implementação Oracle, o Adaptador PostgreSQL utilizou uma API específica [30] para converter objetos do PostGIS em objetos do modelo de dados do IGIS. A Figura 3.30 ilustra o método creatPoint da classe PostgresAdapter.

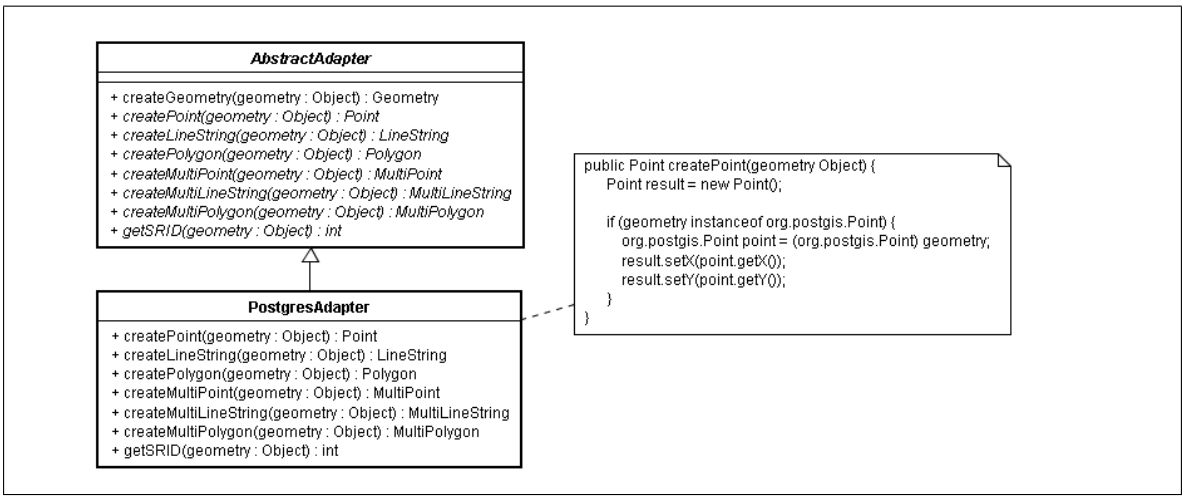

Figura 3.30: Adaptador - PostgreSQL

Acesso a Dados

Para estabelecer uma conexão com o SGDB PostgreSQL é necessário especificar, além do endereço do servidor e porta de comunicação, os atributos database, schema e a senha. Este atributos foram adicionado à classe PostgresDataAccess, ilustrada na Figura 3.31.

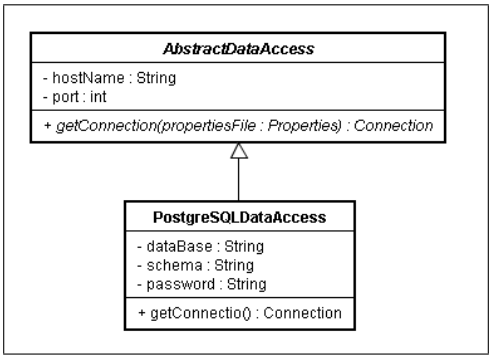

Figura 3.31: Acesso a dados - PostgreSQL

Gerador de Esquema Lógico

Como explicado anteriormente, a implementação do PostgreSQL foi "quase relacional". Dessa forma, esquemas conceituais são mapeados para esquemas lógicos definidos de termos de tabelas relacionais convencionais. Os dois métodos da classe *PostgresLogicalModelGenerator* foram implementados da seguinte forma:

- *generateDimensionsSQL*: cada dimensão de um esquema definido no Mapwarehouse

é mapeado para uma tabela relacional contendo os atributos da dimensão e sua chave-primária. Considerando que pode existir mais de um atributo com o mesmo nome em níveis diferentes, convencionou-se o formato <Nivel><Nome> para identificar os atributos de certo nível.

- *generateFactSQL*: para armazenar os fatos, foi gerada uma tabela relacional contendo cada medida. A chave-primária desta tabela é a composição de todas as chaves primárias das dimensões.

Para o mapeamento de medidas no SGBD PostgreSQL foi considerado o seguinte algoritmo ilustrado no pseudo-código 3.4:

Código Fonte 3.4: Pseudocódigo para mapear medidas

```
if(medida for atômica) {
  //Realiza-se o mapeamento para um tipo primitivo do PostgreSQL
} else if(medida for complexa e uma coleção) {
  //A coleção é mapeada para uma tabela relacional que tem como atributos
      um campo de tipo primitivo referente ao tipo do item da coleção e
      outro campo que representa a chave estrangeira para a tabela de
      fatos. Em outras palavras, é feito um relacionamento 1:N entre a
      tabela de fatos e outra tabela que representará a coleção.
} else {
  //Trata-se de uma medida complexa formada por medidas de tipo distintos
      .
  Para cada medida simples da medida complexa deve-se aplicar este
      algoritmo de forma recursiva.
}
```

Mapeamento de Operações e Funções Espaciais

Assim como na implementação do Oracle, foi necessário estender a classe *AbstractLogicalQueryGenerator*, implementar os métodos referentes às operações e funções espaciais, e realizar o mapeamento das operações e funções espaciais do Mapwarehouse para as funções específicas do PostGIS. A tabela 3.3 representa este mapeamento.

Função ou Operação do Mapwarehouse	Função ou Operação PostGIS
Disjoint	Disjoint
Equals	Equals
Touches	Touches
Contains	Contains
Covers	ST_Covers
Covered_by	ST_CoveredBy
Overlap	Overlaps
Inside	Within
Length	Length
Area	Area
Distance	Distance

Tabela 3.3: Funções e Operações Espaciais - PostGIS

3.5 Discussão

Esse capítulo apresentou a proposta do Mapwarehouse a partir da descrição de sua arquitetura, das interfaces gráficas de definição de esquemas e consultas, além de seu processo de extensão. O desenvolvimento do Mapwarehouse foi guiado por alguns requisitos, entre eles a independência de SGBD e o público-alvo (os gestores).

Para contemplar a independência de SGBD, foram criados em sua arquitetura, pontos de extensão que permitem que programadores o estendam a partir da implementação de algumas funcionalidades.

Para tornar viabilizar a utilização do Mapwarehouse por usuários não especialistas, foi necessário definir um novo metamodelo multidimensional conceitual que considera um DW Espacial uma extensão natural de um DW convencional. O metamodelo definido integra totalmente dados convencionais e espaciais, o que permite modelar aplicações de DW Espacial de forma abrangente. Além disso, foram implementadas interfaces gráficas conceituais com as quais usuários podem definir esquemas de DW Espacial baseados no metamodelo e

consultas analítico-espaciais em vários níveis de complexidade.

Este capítulo também apresentou o processo de extensão do *framework*, ou seja, como um programador deve estender o *framework* para utilizá-lo em diferentes SGBDs. Neste sentido, foram descritos cada ponto de extensão e a forma como eles devem ser implementados.

Além disso, foram apresentas de forma resumida algumas questões relacionadas à implementação dos pontos de extensão para os SGBDs Oracle e PostgreSQL. A implementação Oracle considerou o modelo de dados objeto-relacional, que permite criar tabelas de objetos a partir de tipos abstratos de dados. Por outro lado, a implementação PostgreSQL foi quase puramente relacional. Por conta da utilização de distintos modelos de dados, foi necessário definir alguns pontos de extensão de forma genérica.

Capítulo 4

Aplicação do *Framework*

Este capítulo apresenta dois estudos de caso de aplicações de DW Espacial desenvolvidas com auxílio do *framework* Mapwarehouse. Cada estudo de caso será descrito da seguinte maneira:

- Uma descrição sucinta do estudo de caso.
- Uma descrição do esquema da aplicação.
- A etapa de definição do esquema utilizando o módulo de definição de esquemas do *framework*.
- Exemplificação de um conjunto de consultas visando demonstrar as interfaces gráficas e funcionalidades do Mapwarehouse.

4.1 Estudo de Caso: Distribuição Agrícola

Esta seção apresenta o estudo de caso de uma aplicação de DW Espacial referente à distribuição agrícola do estado da Paraíba. Nesta aplicação exemplo, informações como o tipo de solo, o tipo de plantação, a precipitação e a localização são consideradas para auxiliar o processo de tomada de decisões relacionadas a políticas de distribuição de sementes.

4.1.1 Esquema Conceitual de Distribuição Agrícola

Para sua implementação, foi definido o esquema da Figura 4.1. Segundo o esquema, o fato a ser analisado é representado pelas medidas complexas Áreas Plantadas (*Crop_Areas*), que é uma sequência de polígonos representando áreas de plantação, e Quantidades (*Quantities*), que é uma sequência de valores numéricos representando o total de grãos plantados em cada área plantada.

As seguintes dimensões caracterizam os fatos: Plantação (*Plantation*), Solo (*Soil*), Precipitação (*Precipitation*), Mês (*Month*) e Município (*Municipality*). As dimensões Mês e Município possuem respectivamente, as hierarquias Mês < Ano (*Month* < *Year*) e Município < Microrregião < Região < Estado (*Municipality* < *Microregion* < *Region* < *State*).

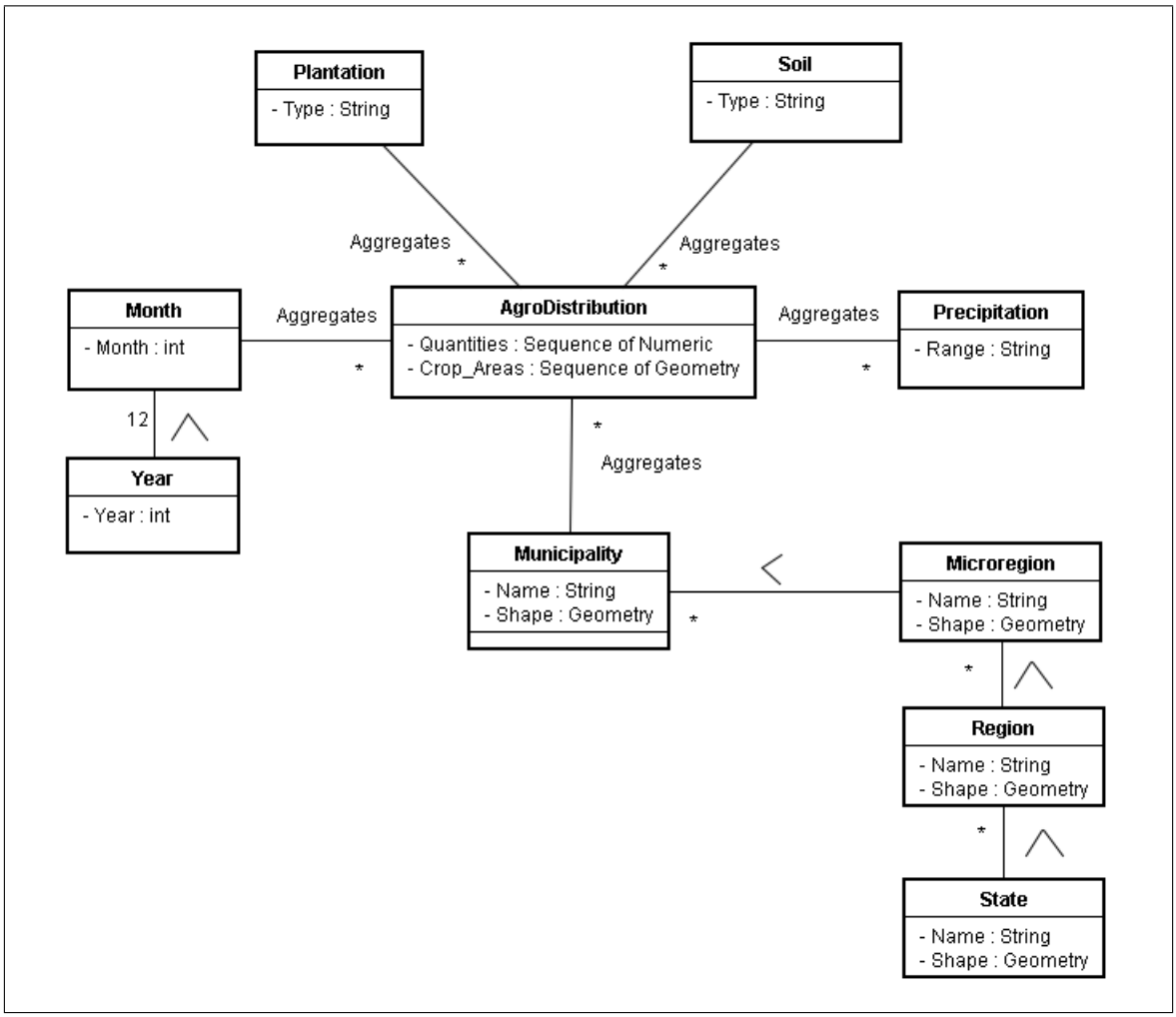

Figura 4.1: Diagrama UML - Distribuição Agrícola

4.1.2 Definição do Esquema

Como ilustrado na seção 3.3, esquemas de DW espaciais podem ser definidos a partir de um conjunto de interfaces gráficas que implementam basicamente duas funcionalidades: definição de medidas e definição de dimensões.

Medidas

As medidas da aplicação Distribuição Agrícola, isto é, Quantidades e Áreas Plantadas, foram definidas a partir das interfaces gráficas ilustradas na Figura 4.2. Para cada uma das medidas foram atribuídos seus respectivos nomes e tipos de dados.

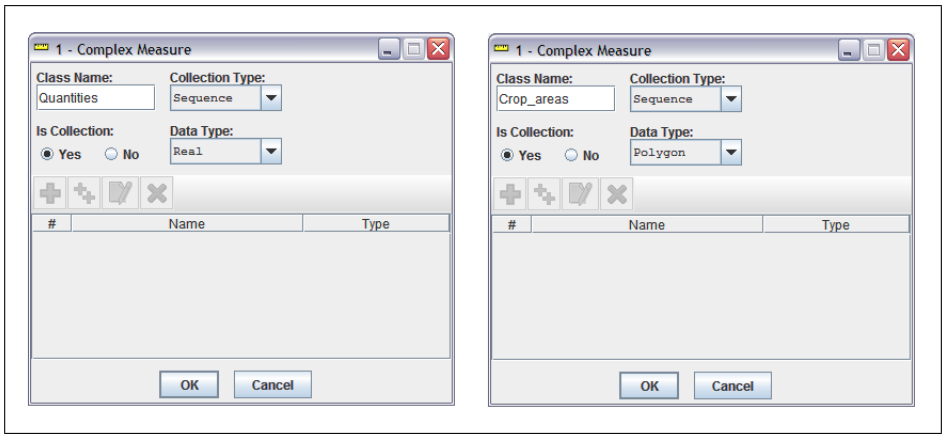

Figura 4.2: Definição das medidas

Dimensões e Hierarquias

Cada dimensão do modelo de dados foi definida utilizando o *Wizard* apresentado na subseção 3.3.1, o qual permite definir dimensões, seus níveis e suas hierarquias a partir uma única interface gráfica. A Figura 4.3 ilustra os três passos necessários para a definição da dimensão Município (*Municipality*). O passo 1 ilustra a definição dos atributos Nome (*Name*) e Geometria (*Shape*) que compõem o nível Município. Os passos 2 e 3 ilustram respectivamente, a definição dos outros níveis da dimensão e a configuração da hierarquia gerada automaticamente pelo Mapwarehouse. Caso fosse necessário, esta hierarquia poderia ser modificada a partir da interface gráfica da Figura 4.4.

Após a definição de todas as medidas e dimensões, o esquema da aplicação ficou como ilustrado na Figura 4.5.

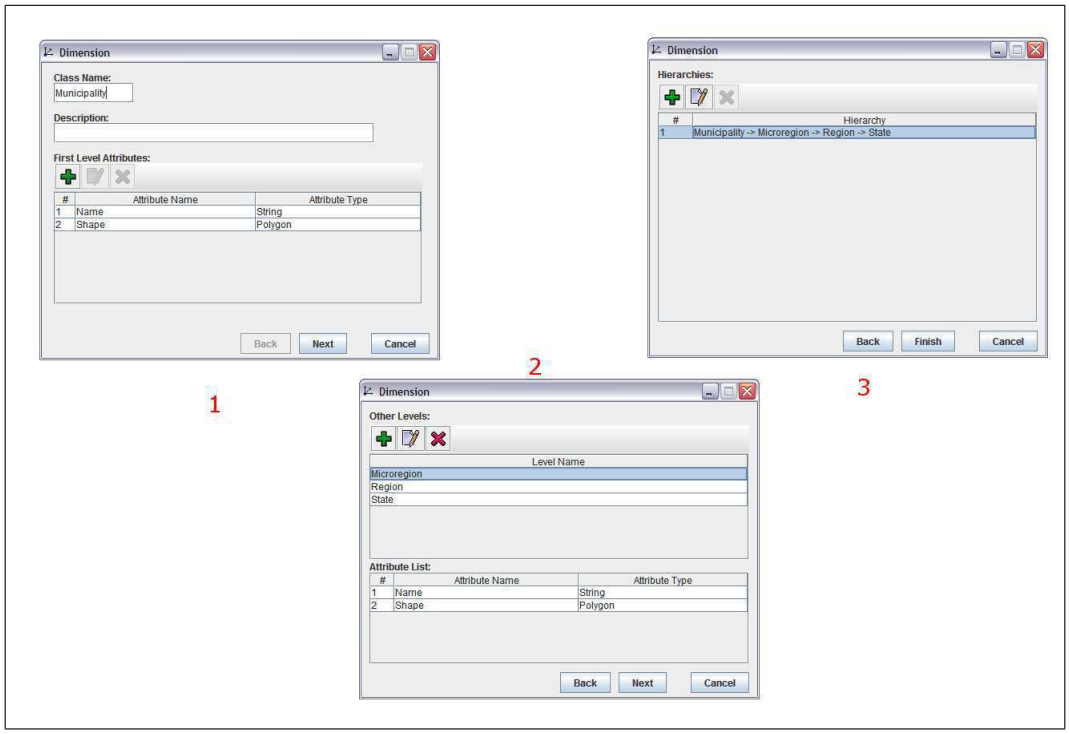

Figura 4.3: Definição da dimensão Município

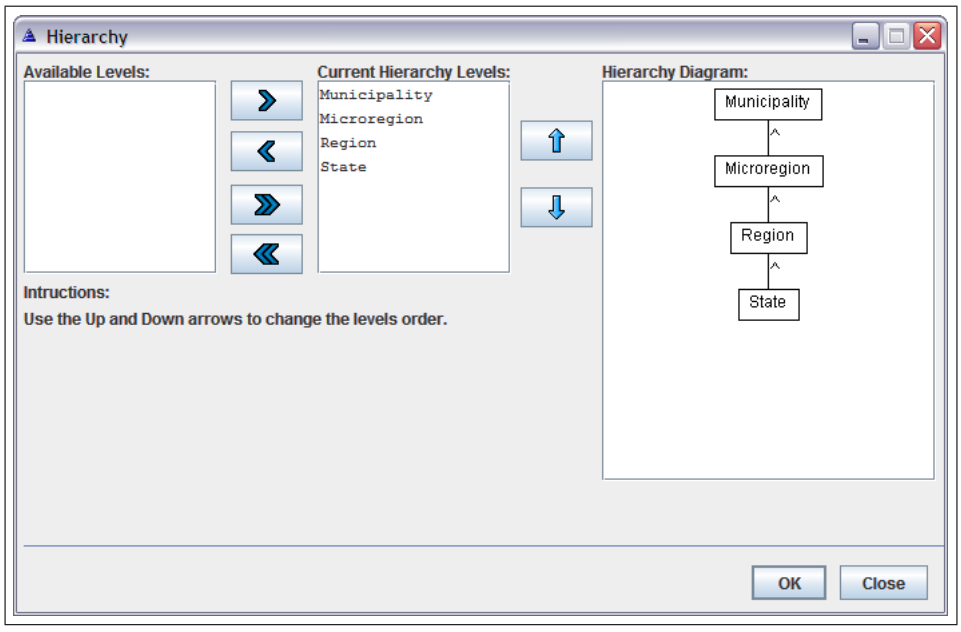

Figura 4.4: Hierarquia da dimensão Município

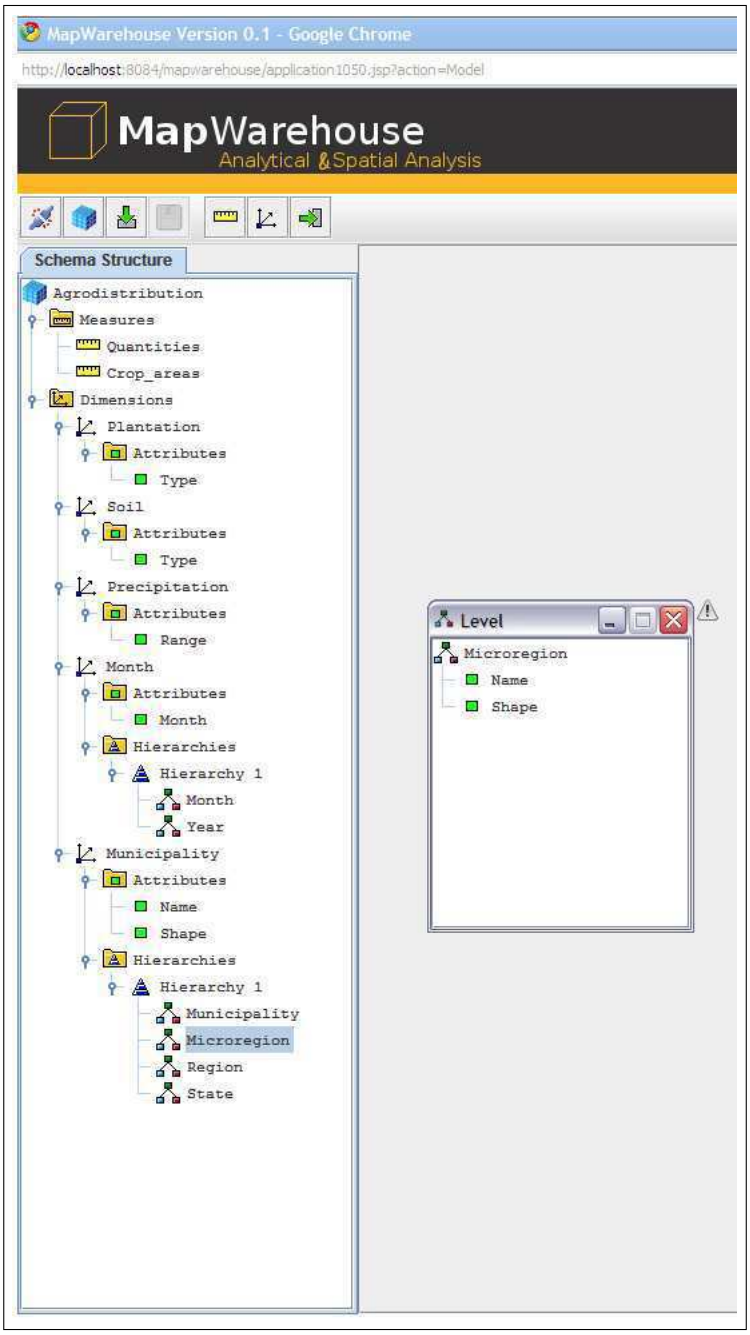

Figura 4.5: Estrutura do Esquema Distribuição Agrícola

Geração de Esquema Lógico

Uma vez que todo o esquema da aplicação foi definido, foi gerado, a partir de um botão da interface gráfica, o esquema lógico para o SGBD Objeto-Relacional Oracle 10g. O *script* representando o esquema pode ser parcialmente observado na Figura 4.6.

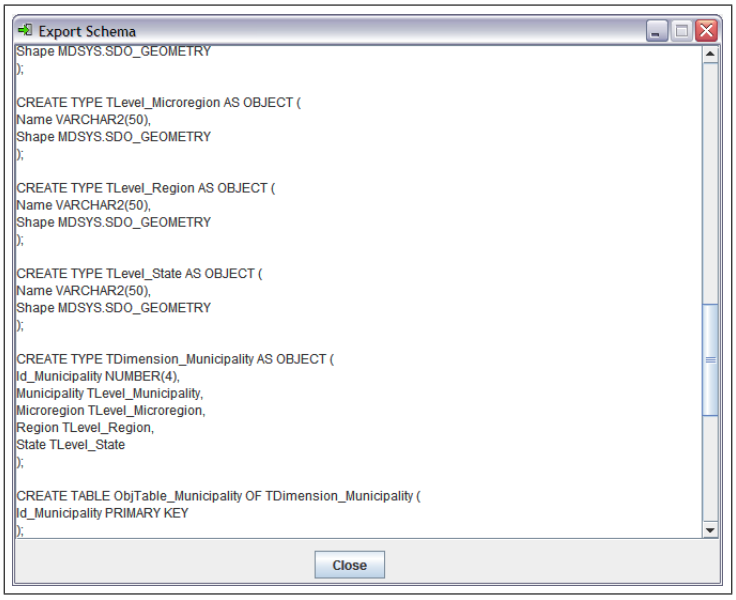

Figura 4.6: Geração de Esquema Lógico

4.1.3 Consultas Exemplo

As consultas a seguir têm como objetivo apresentar as funcionalidades relacionadas as tarefas de definição e execução de consultas no Mapwarehouse.

Consulta 1

Considerando a consulta: *"exiba as áreas plantadas de milho e feijão e a quantidade de milho e feijão para cada região e microrregião do estado da Paraíba, durante Maio de 2009"*, serão apresentados os passos para sua definição e execução.

Inicialmente, foram selecionadas as medidas Áreas Plantada e Quantidades a serem exibidas no resultado da consulta. Esta tarefa está ilustrada na Figura 4.7.

Logo após, foram selecionados os atributos escolhidos a serem visualizados em mapas e tabelas. Da dimensão plantação (*Plantation*) foi selecionado o atributo tipo (*type*) e da

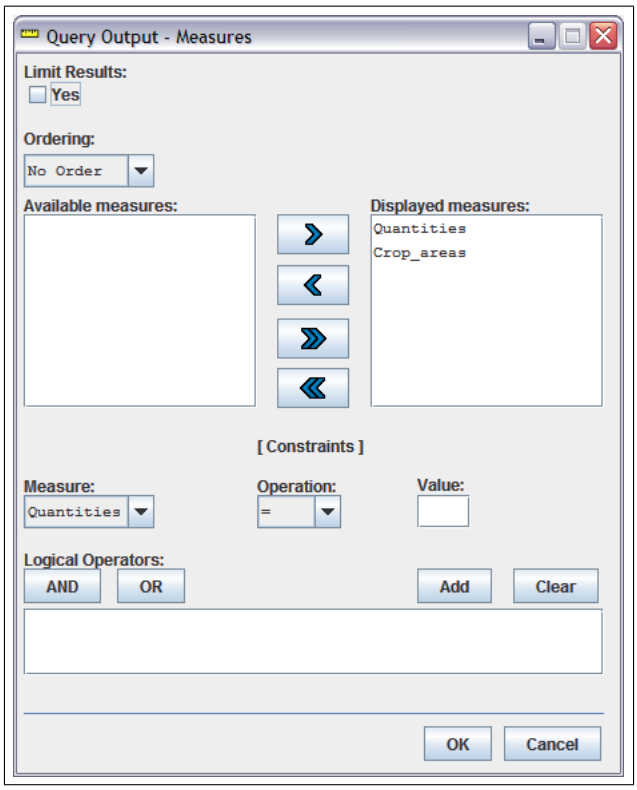

Figura 4.7: Escolha das medidas

dimensão Município (*Municipality*) foram selecionados os atributos descritivos e espaciais dos níveis Microrregião (*Microregion*) e Região (*Region*). Esta tarefa está ilustrada na Figura 4.8.

Além disso, foram definidas as restrições para o tipo de plantação (Milho ou Feijão), o mês e ano (Março de 2009) e o estado (Paraíba), como observado na Figura 4.9.

Por fim, foi definida a operação OLAP *drill-down*, visando exibir as informações desde o nível de detalhe Microrregião até o nível Região. Esta tarefa é ilustrada na Figura 4.10.

Após sua definição da consulta, a consulta ficou da forma como apresentada no *workspace* da Figura 3.13.

O resultado da consulta é ilustrado na Figura 4.12. A Figura 4.12 também ilustra uma pequena caixa com detalhes sobre determinada medida. Esta caixa de detalhes é exibida a partir do clique do mouse sobre a medida.

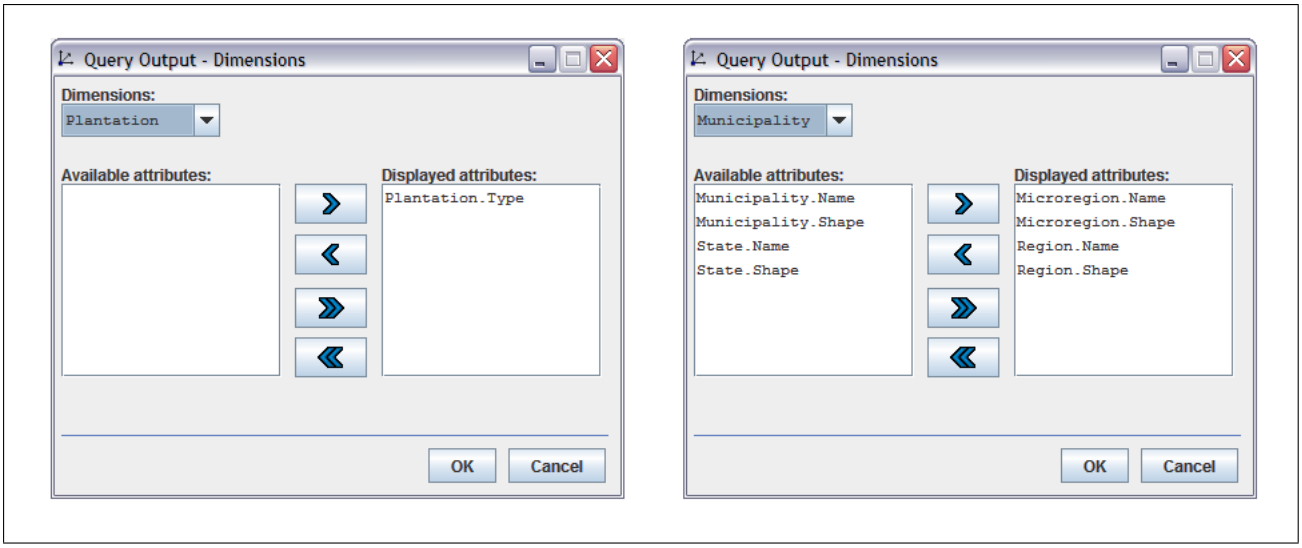

Figura 4.8: Escolha dos atributos das dimensões

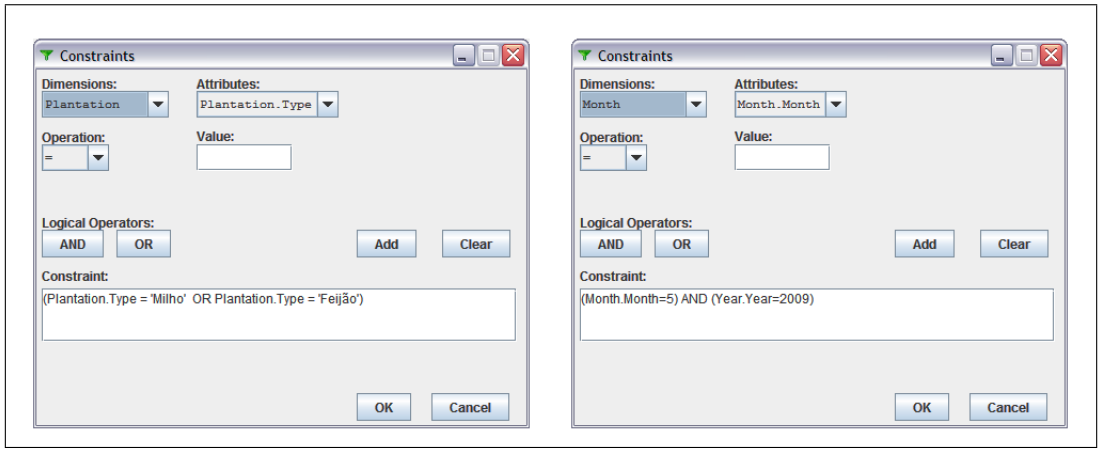

Figura 4.9: Definição das restrições

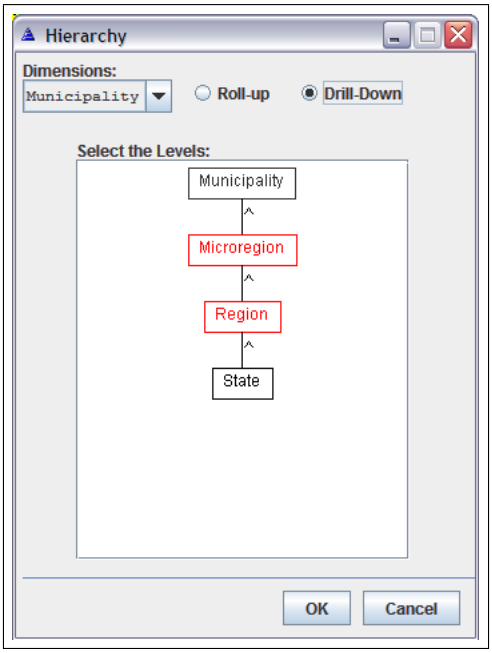

Figura 4.10: Seleção dos níveis para *Roll-up*

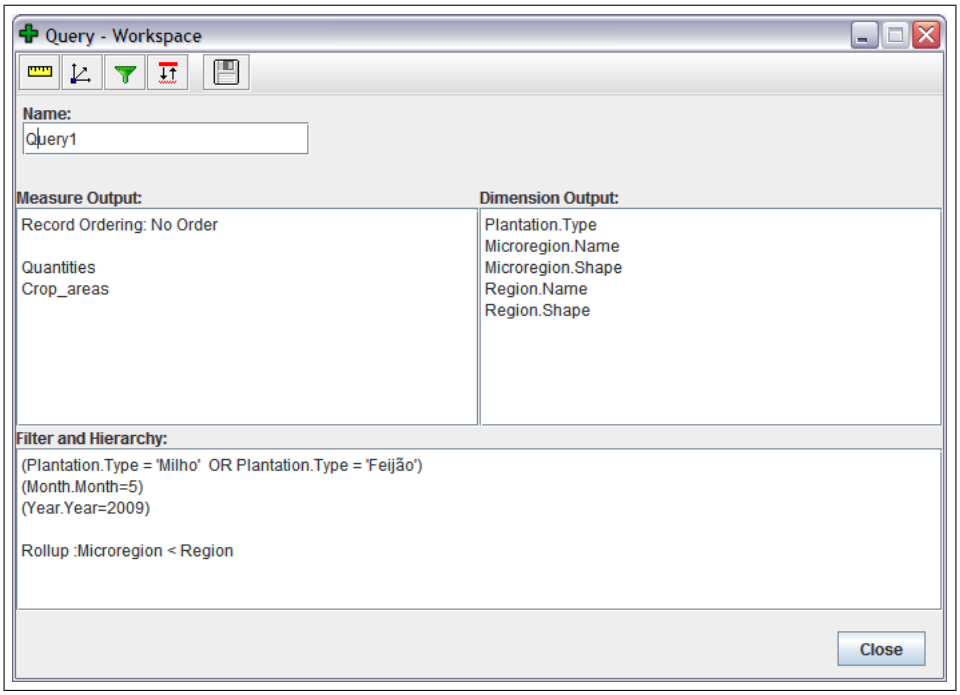

Figura 4.11: Configuração da Consulta 1

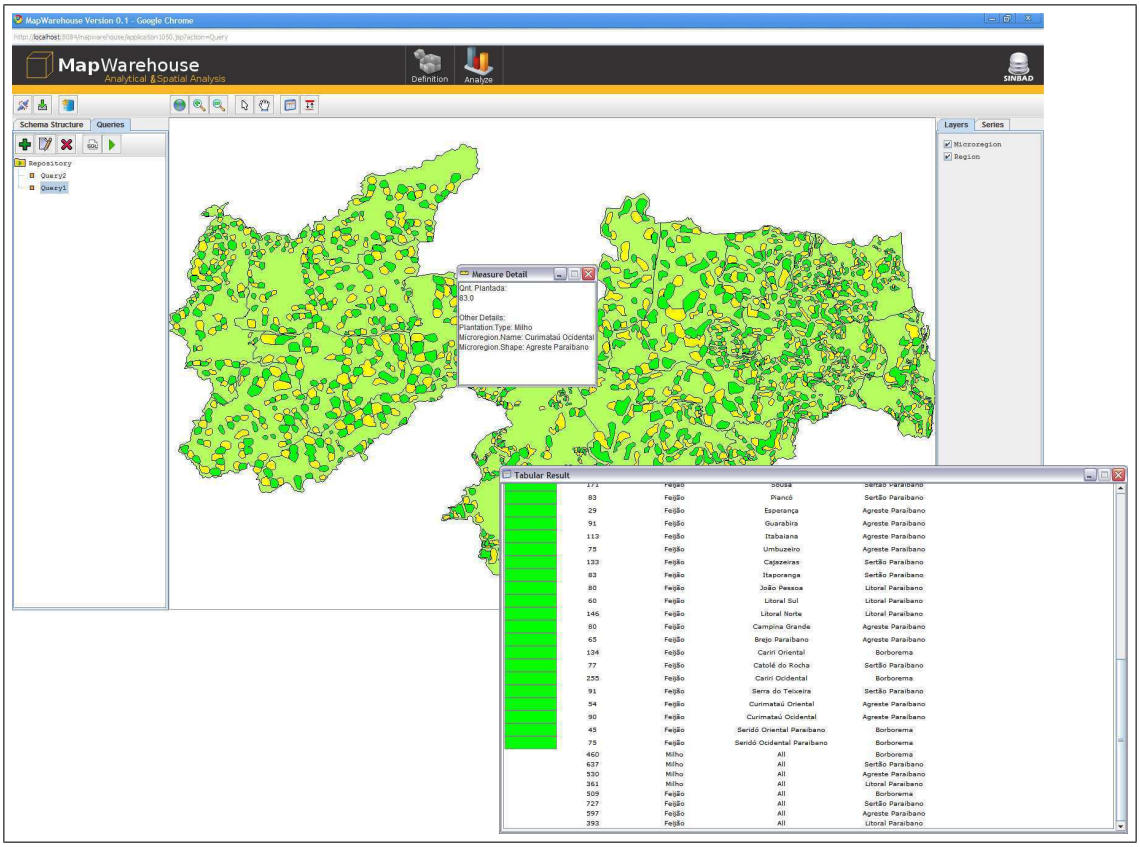

Figura 4.12: Resultado da consulta 1

Consulta 2

"Exiba em um mapa, a quantidade plantada de milho e feijão para cada microrregião do estado da Paraíba".

Inicialmente foi selecionada a medida numérica Quantidades (*Quantities*), como observado na Figura 4.13.

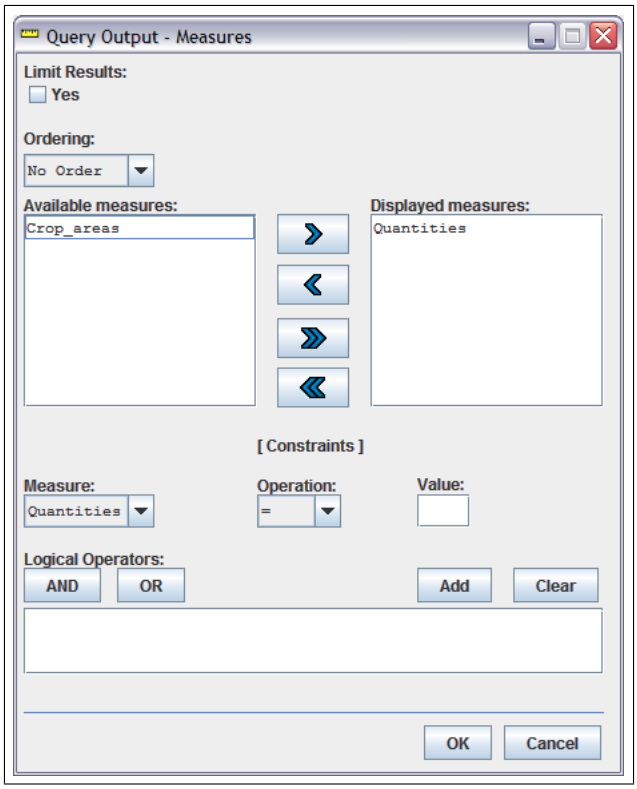

Figura 4.13: Escolha das medidas

Em seguida, foram selecionados os atributos do nível Microrregião da dimensão Município. Considerando que a consulta explicita que seu resultado deve ser exibido no mapa, foi selecionado o atributo espacial (*Shape*) de Microrregião. Esta tarefa é ilustrada na Figura 4.14.

Além disso, foram definidas as restrições para o tipo de plantação (Milho ou Feijão) e o Estado (Paraíba). Esta tarefa é ilustrada na Figura 4.15.

Ao final da definição da consulta, o *workspace* configurou-se como ilustrado na Figura 4.16.

Os resultado da consulta é apresentado na Figura 4.17.

Visando exemplificar a utilização das operações OLAP de maneira *ad-hoc*, foi realizado

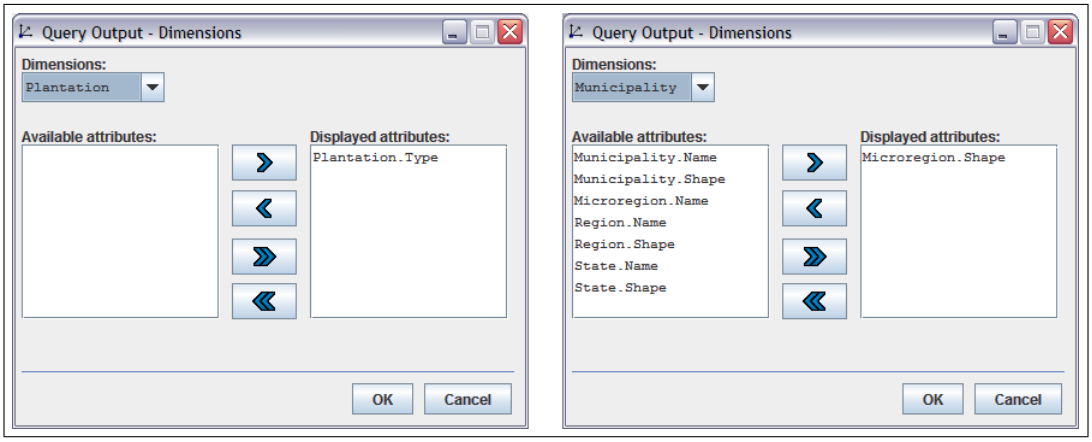

Figura 4.14: Escolha dos atributos das dimensões

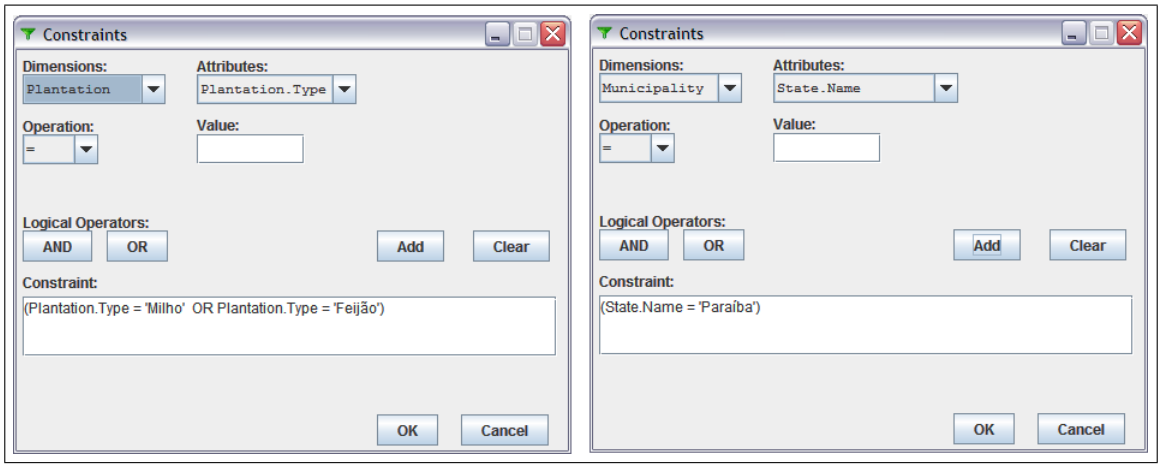

Figura 4.15: Definição das restrições

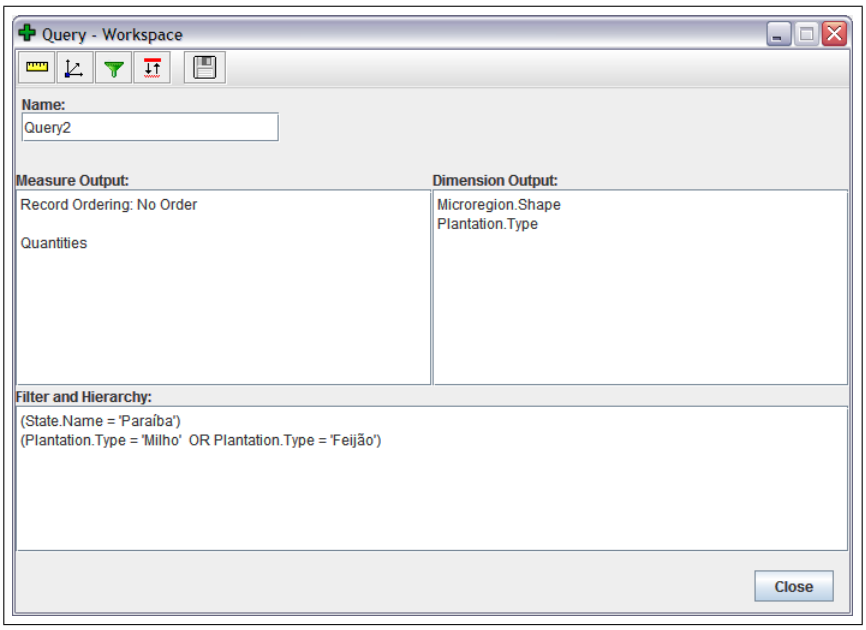

Figura 4.16: Configuração da Consulta 2

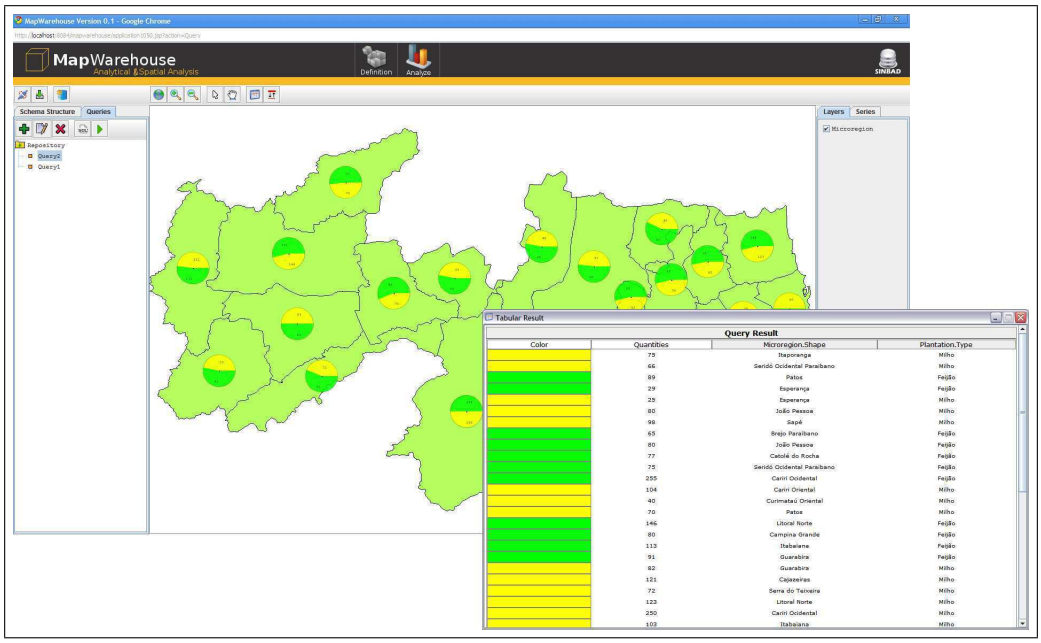

Figura 4.17: Resultado da consulta 2

um *Roll-up* do nível Microrregião para Região, como apresentado na Figura 4.18. O resultado desta operação *Roll-up* pode ser observado na Figura 4.19, na qual observa-se o novo mapa gerado e o componente tabular atualizado com os valores agregados pelo nível Região.

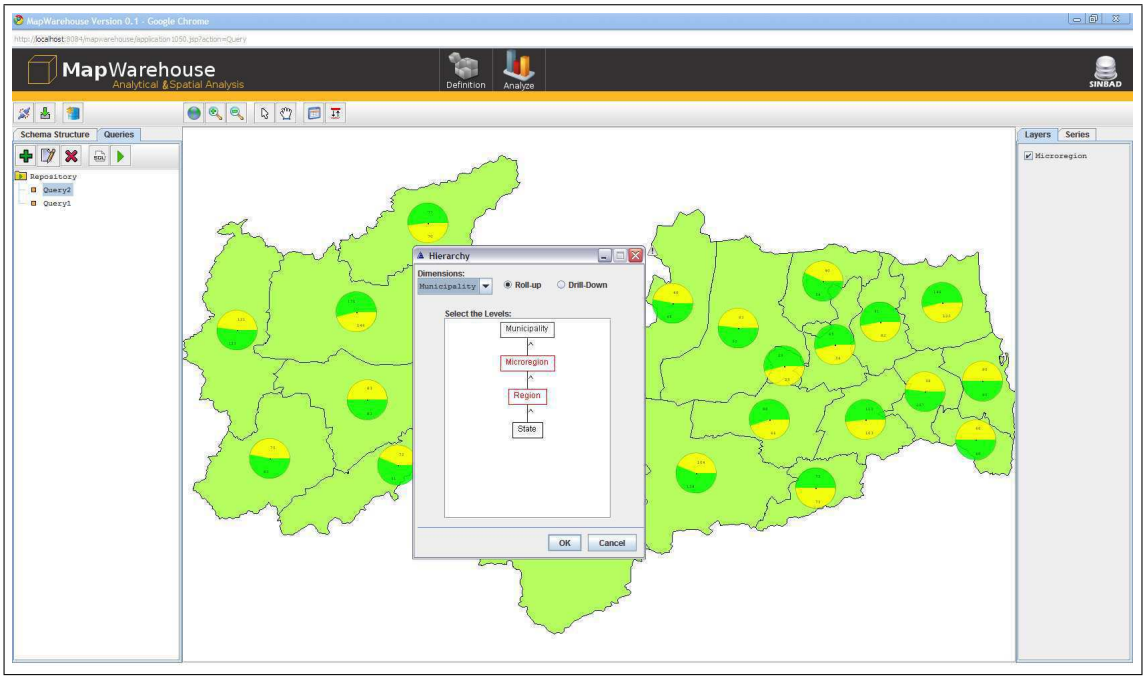

Figura 4.18: Roll-up de Microrregião para Região

Três condições devem ser satisfeitas para que o Mapwarehouse apresente gráficos sobreposto como resultado de uma consulta. São elas:

1. Apenas medidas numéricas devem ser escolhidas para serem exibidas como resultado de consultas.

2. Pelo menos um atributo espacial deve ser escolhido, para que seja gerado um mapa.

3. Pelo menos dois critérios para agregação de medidas devem ser escolhidos, ou seja, atributos de pelo menos duas dimensões.

A combinação das três condições justifica a utilização de gráficos sobrepostos a mapas, visto que pode-se ter mais de um valor agregado de medida numérica para um mesmo elemento geométrico. Logo, a forma mais intuitiva de representar esses valores é através de gráficos.

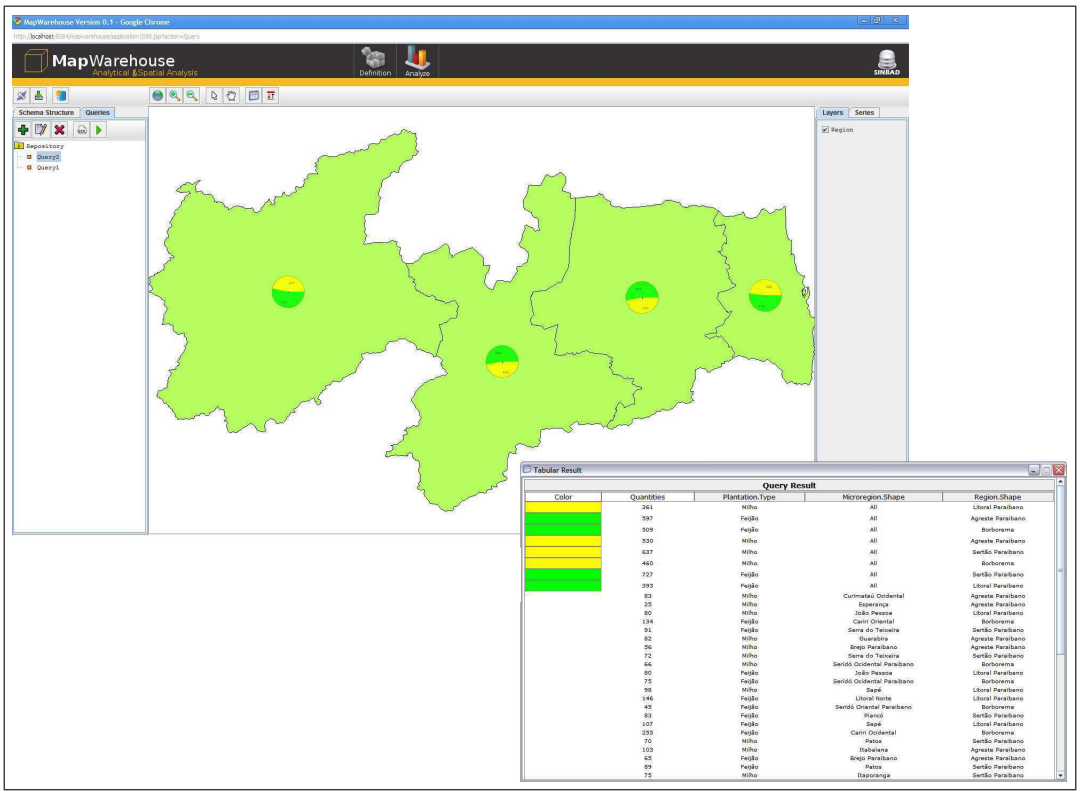

Figura 4.19: Resultado da operação *Roll-up*

Consulta 3

A terceira consulta: *"exiba as áreas plantadas e quantidade plantada de milho nos municípios adjacentes à cidade de Campina Grande"* tem o objetivo de ilustrar o uso de restrições com operações espaciais.

Inicialmente, foram selecionadas as medidas Áreas Plantadas e Quantidades a partir da interface gráfica da Figura 4.20.

Em seguida, foram selecionados os atributos Nome (*Name*) e Geometria (*Shape*) da dimensão Municípios, como observado na Figura 4.21.

Por fim, foi definida a restrição para o tipo de plantação (Milho) e a restrição com operação espacial *Touches*, visando selecionar apenas os municípios adjacentes ao município de Campina Grande. Ambas as restrições são ilustradas na Figura 4.22.

A geometria do município de Campina Grande foi escolhida utilizando a interface gráfica ilustrada na Figura 4.23 denominada interface de Janela Espacial. Esta janela é acessada a partir do botão destacado em vermelho na Figura 4.22. A interface de janela espacial provê basicamente duas funcionalidades que são: permitir a seleção de um elemento geométrico

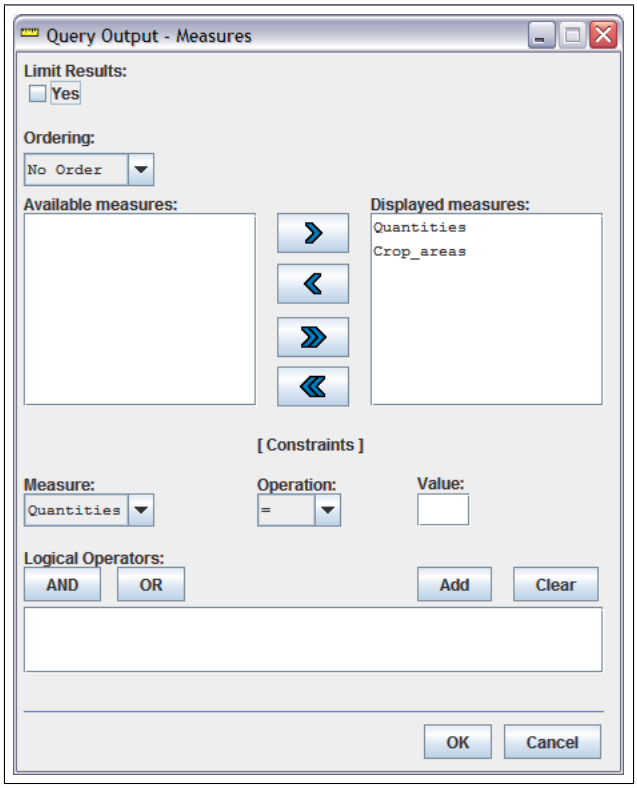

Figura 4.20: Escolha das medidas

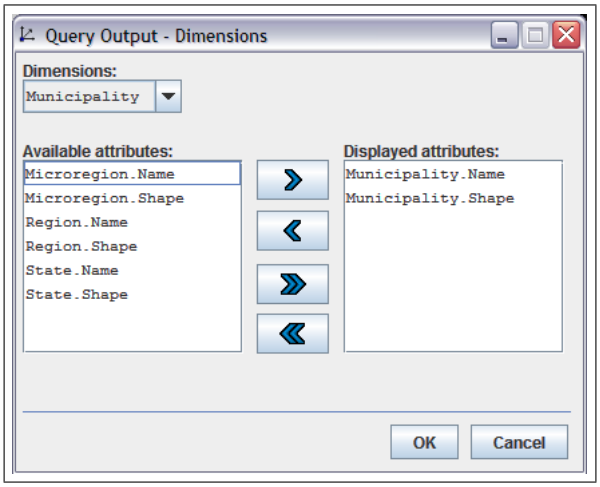

Figura 4.21: Escolha dos atributos Nome e Geometria

do mapa e permitir a definição de uma janela espacial (*Clipping Area*). Estas duas funcionalidades são acionadas pelos botões do canto superior direito destacados em vermelho.

Para selecionar a geometria do município de Campina Grande bastou clicar no botão cujo ícone é uma seta, e então clicar sobre o município de Campina Grande. Com isso, o Mapwarehouse mapeou o ponto clicado para um ponto do mapa e verificou qual o elemento geométrico da camada ativa continha o ponto clicado.

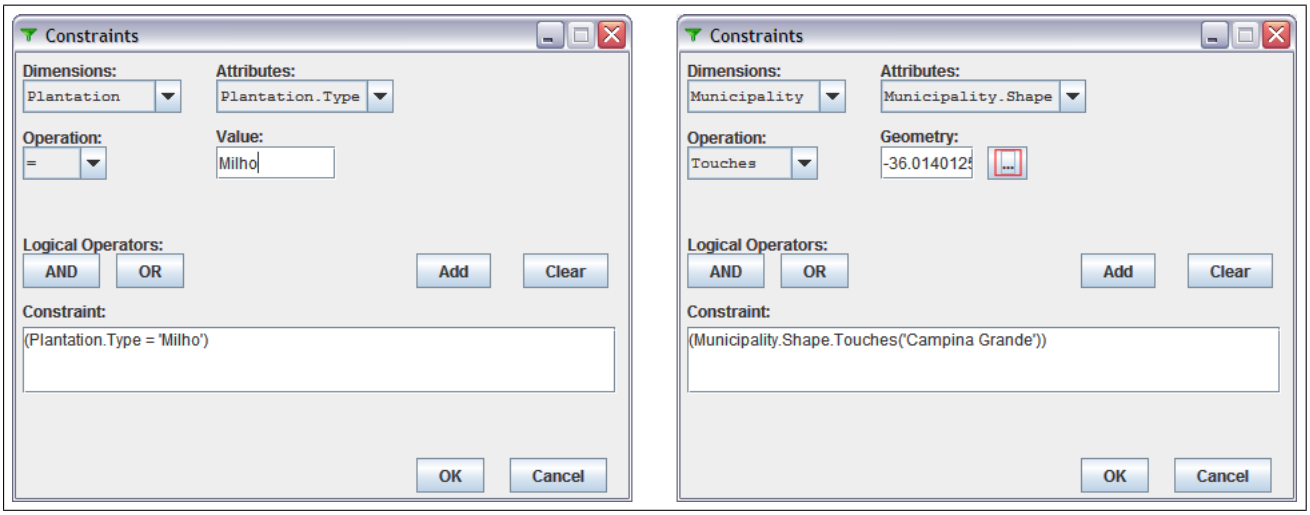

Figura 4.22: Definição da restrição espacial

O resultado da consulta 3 é ilustrado na Figura 4.24.

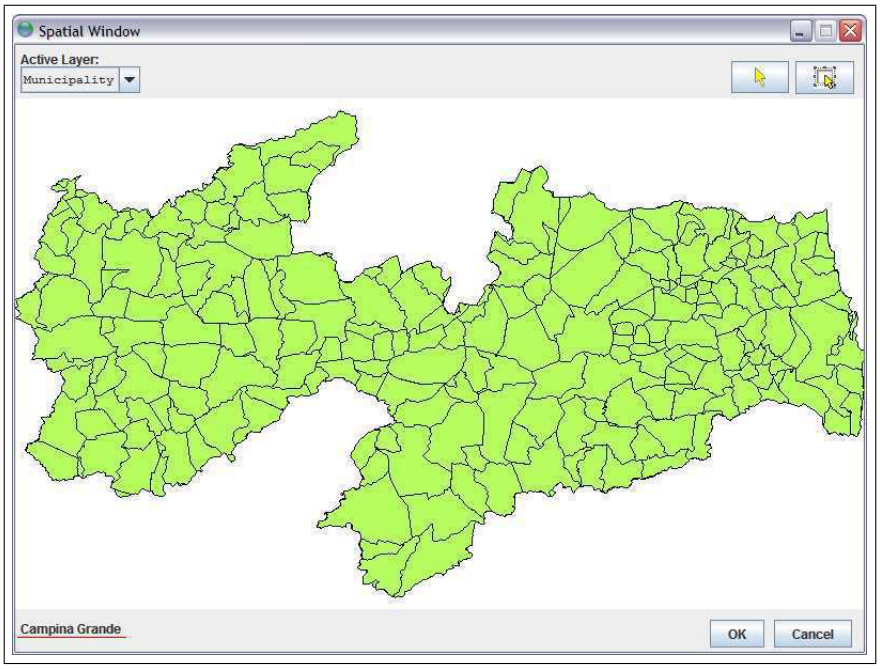

Figura 4.23: Seleção do município de Campina Grande

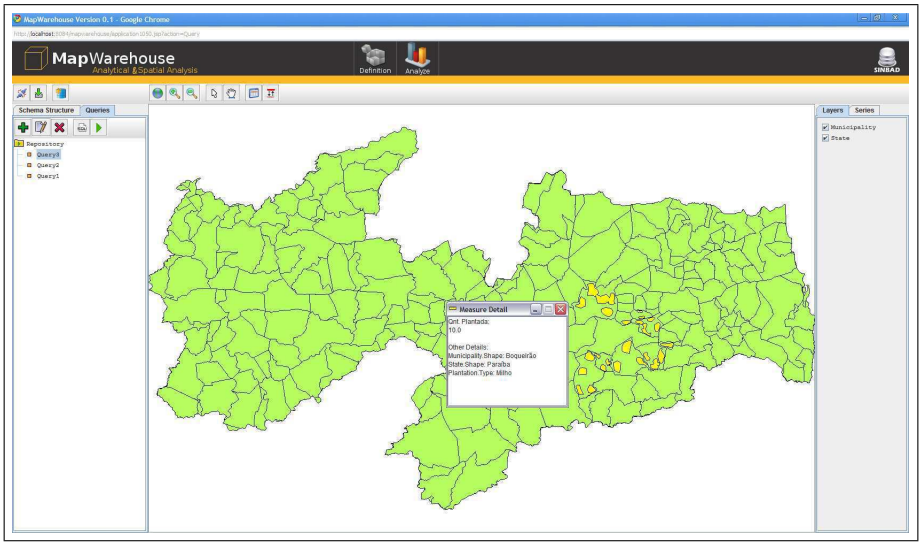

Figura 4.24: Resultado da consulta 3

Consulta 4

"Exiba as áreas geográficas de plantação de milho e feijão e quantidade de milho e feijão por plantação, dentro de uma janela retangular, por microrregião e região do estado da Paraíba".

A consulta 4 visa demonstrar a outra funcionalidade da interface gráfica de janela espacial, ou seja, a seleção de uma *Clipping Area*. Para definição desta consulta foram selecionados os atributos Tipo de Plantação da dimensão Plantação e as Geometrias dos níveis microrregião e região. Além disso, foram definidas as restrições para o tipo de plantação e estado, conforme observado na Figura 4.25.

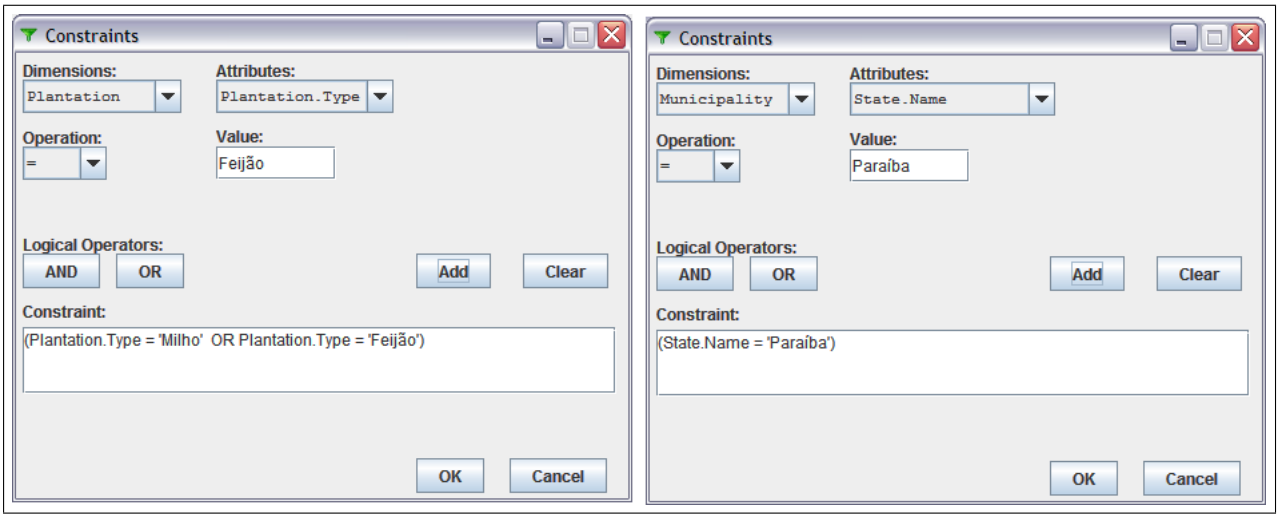

Figura 4.25: Definição das restrições

Em seguida foram selecionadas as medidas Áreas Plantadas e Quantidades. Para acessar a interface Janela Espacial, foi utilizado o botão destacado em vermelho na Figura 4.26.

Na interface Janela Espacial, ilustrada na Figura 4.27, foi utilizado a funcionalidade para selecionar a *Clipping Area*. A partir da seleção de dois pontos da interface gráfica, o Mapwarehouse automaticamente gera o MBR (*Minimum Bounding Rectangle*) que contém os dois pontos selecionados.

Após sua execução, o Mapwarehouse apresentou o resultado ilustrado na Figura 4.28.

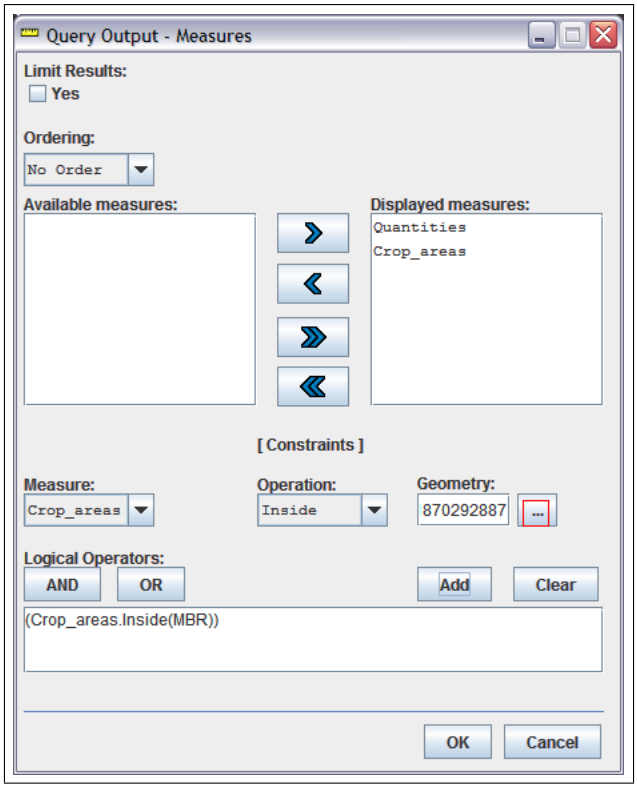

Figura 4.26: Seleção das medidas e restrição espacial

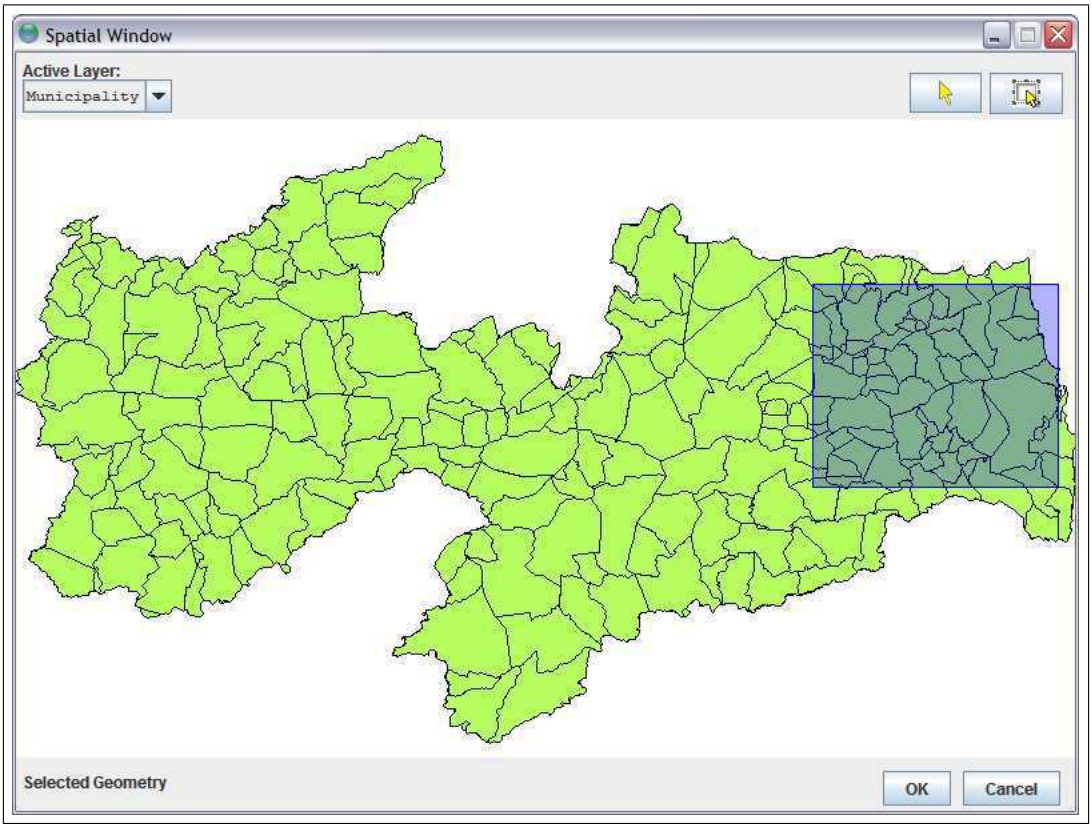

Figura 4.27: Seleção da Janela Espacial

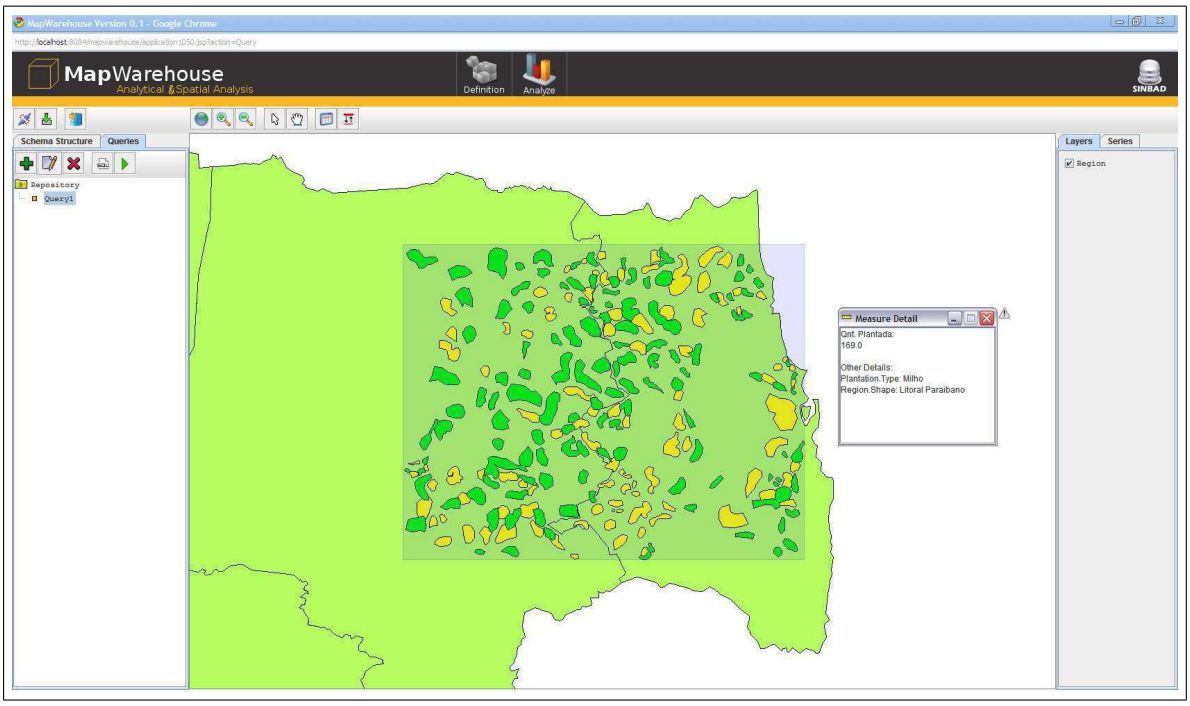

Figura 4.28: Resultado da Consulta 4

Consulta 5

"Exiba as áreas geográficas e quantidades plantadas de milho, por tipo de solo e por faixa de precipitação, no segundo trimestre de 2009, na região Sertão da Paraíba".

Para definição desta consulta foram selecionadas as medidas Áreas Plantadas e Quantidades Plantadas, além dos atributos das dimensões Plantação, Precipitação e Solo. Além disso, foram definidas as restrições ilustrada na Figura 4.29. A restrição para a região Sertão da Paraíba pode definida de duas formas: utilizando o atributo Nome do nível Região (i.e. Region.Name = 'Sertão da Paraíba') ou utilizando o operador espacial *Equals* aplicado às geometrias do nível Região. Foi utilizada a segunda opção visando demonstrar novamente a utilização da interface de Janela Espacial. Observa-se na Figura 4.30, a seleção (no canto inferior esquerdo) da região Sertão da Paraíba. Isto foi feito da seguinte forma: foi selecionada a camada *Region* como camada ativa (*Active Layer*) e foi utilizada a funcionalidade de seleção de geometrias para clicar sobre o Sertão da Paraíba exibido no mapa.

Após a definição da consulta, o *workspace* configurou-se como ilustrado na Figura 4.31.

Os resultados na forma de mapa e na forma tabular são ilustrados respectivamente nas Figuras 4.32 e 4.33.

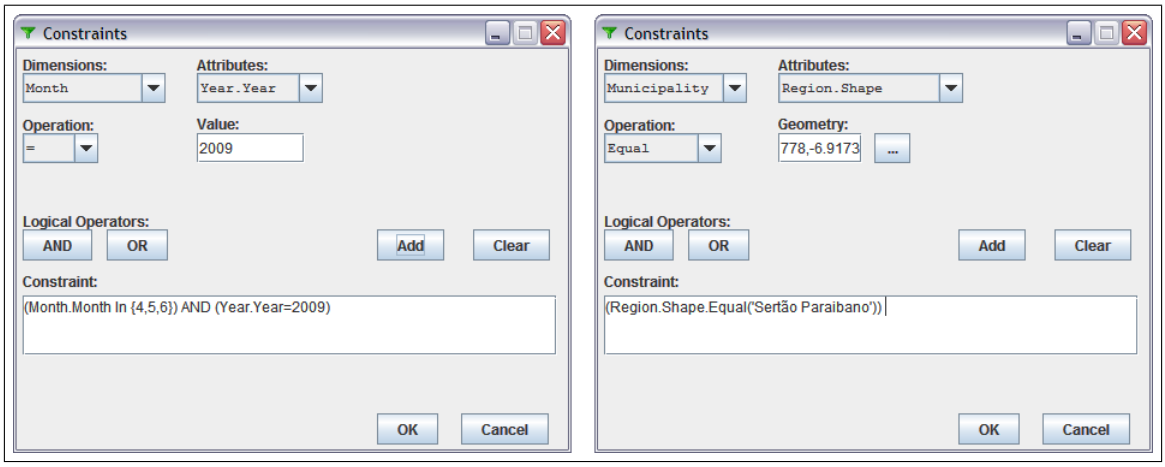

Figura 4.29: Resultado da Consulta 4

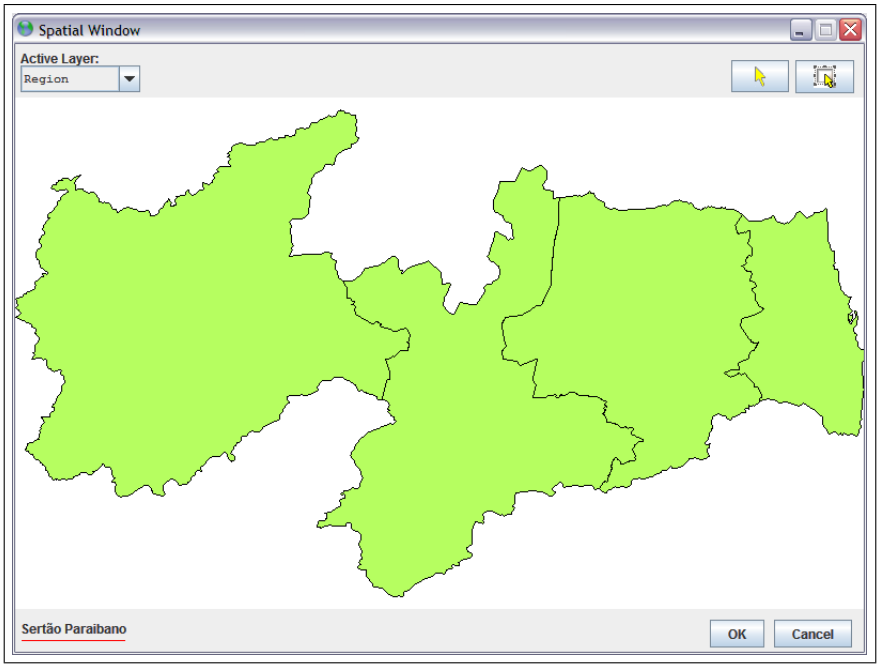

Figura 4.30: Seleção da geometria da Região Sertão da Paraíba

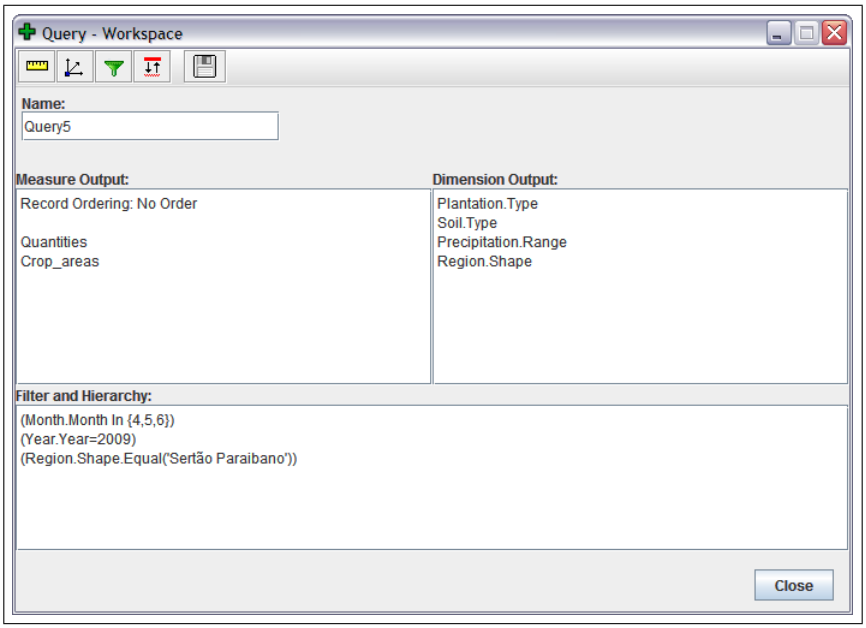

Figura 4.31: *Workspace*

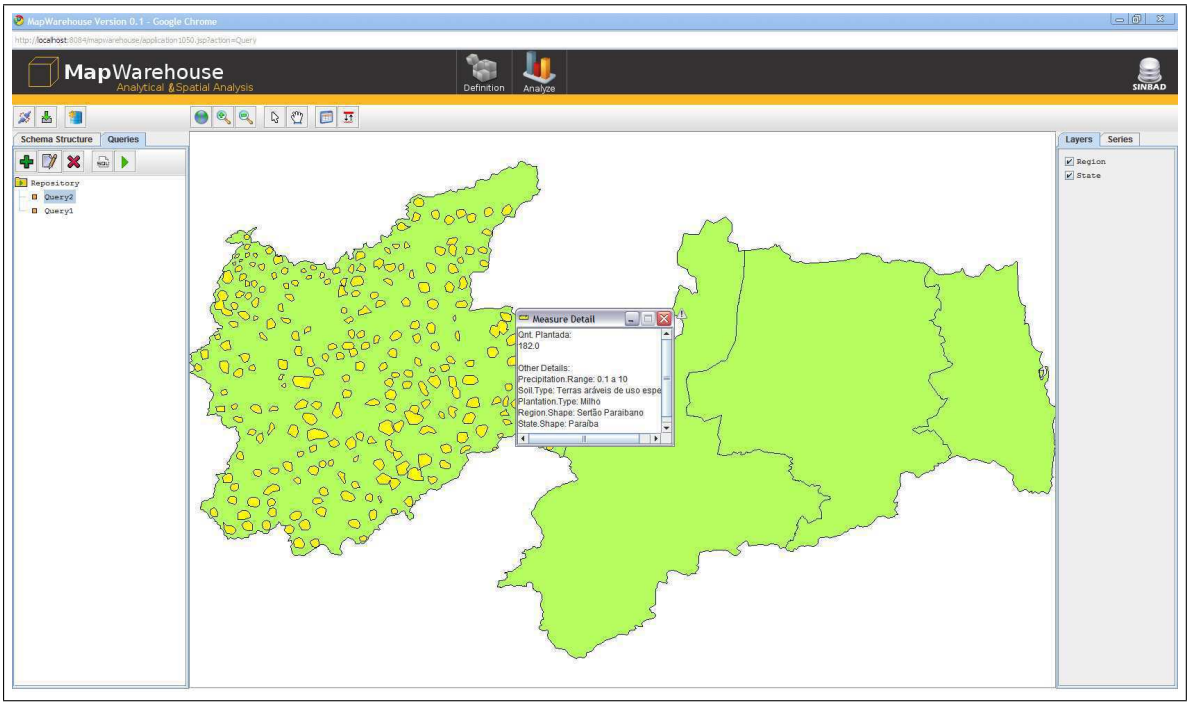

Figura 4.32: Resultado da Consulta 5

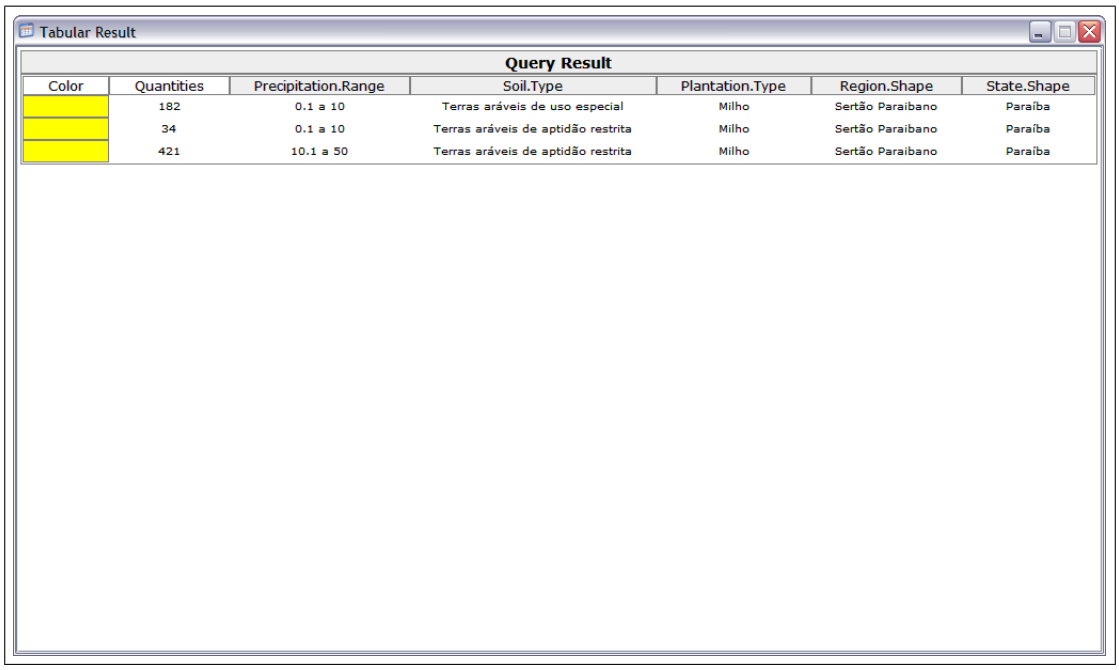

Figura 4.33: Resultado da Consulta 5

Consulta 6

"Exiba as áreas geográficas plantadas, mês a mês, ano a ano, por município do estado da Paraíba".

A consulta 6 representa uma consulta com série temporal. Ferramentas SOLAP devem resolver consultas com série temporal de maneira distinta das consultas convencionais, visto que o resultado de uma consulta com série temporal pode envolver a geração de vários mapas.

Para definição desta consulta foi selecionada a medida Áreas Plantadas, o atributo Geometria do nível Município da dimensão Município, além dos atributos Mês e Ano da dimensão temporal (Figura 4.34).

Quando o Mapwarehouse identifica uma consulta com série temporal, ele gera um *slider* como o ilustrado na Figura 4.35. Os itens do *slider* representam os diferentes grupos da dimensão temporal que serão considerados na realização da consulta. No exemplo da consulta 6, foi gerado o *slider* com os itens: 4/2009, 5/2009, 6/2009 e 7/2009, visto que o DW Espacial foi povoado apenas com fatos referentes a estes meses para o ano de 2009.

A partir do *slider* gerado, o usuário pode navegar pelos itens e visualizar os resultados da consulta para cada mês isoladamente. As Figuras 4.36, 4.37 e 4.38 representam, respectivamente, os resultados da consulta para os meses de Abril, Maio e Junho.

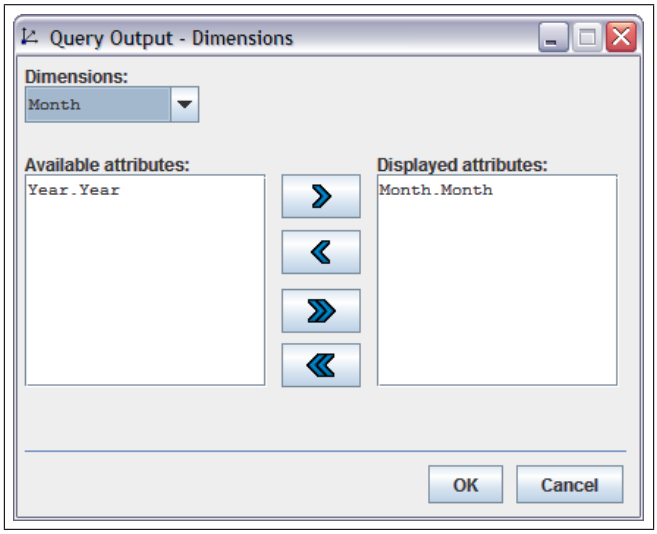

Figura 4.34: Seleção do atributo *Month*

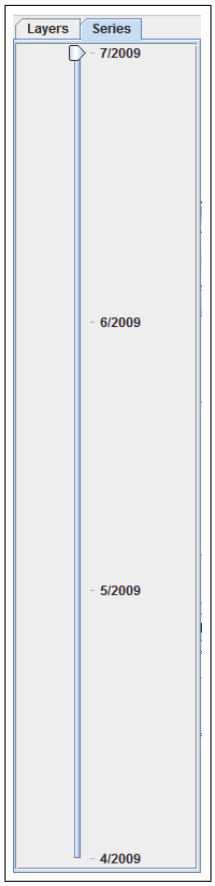

Figura 4.35: *Slider* representando a série temporal

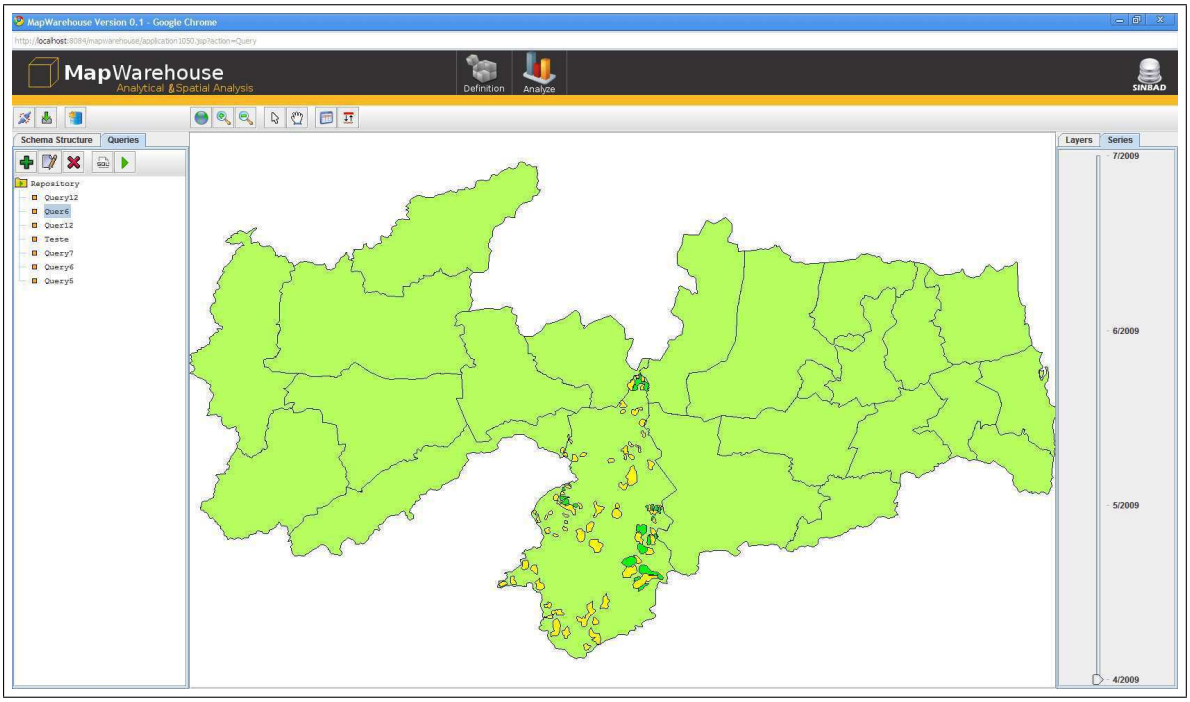

Figura 4.36: Resulta da Consulta 6 - Mês de Maio

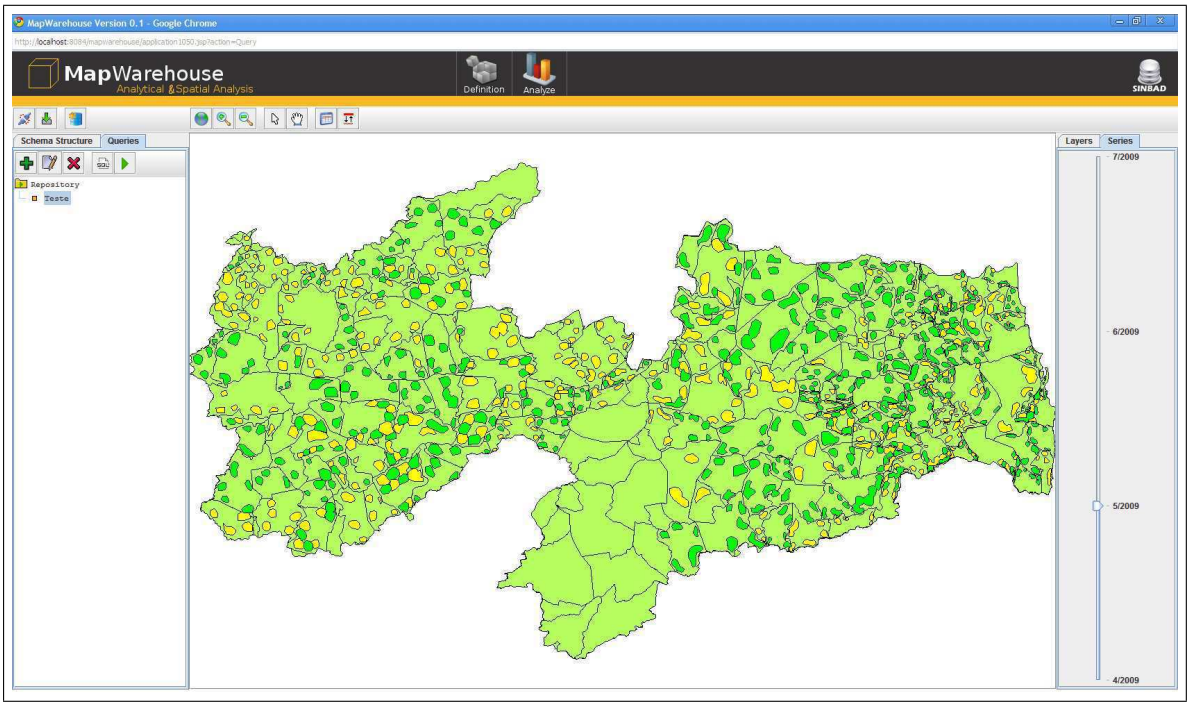

Figura 4.37: Resulta da Consulta 6 - Mês de Junho

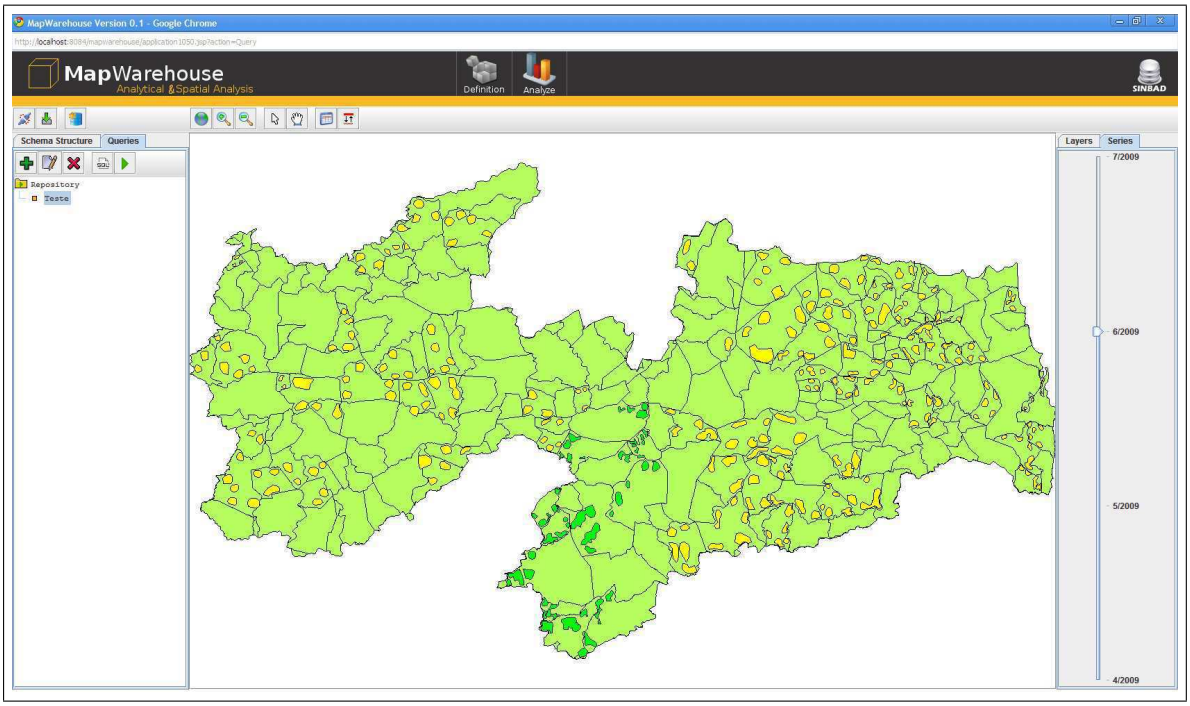

Figura 4.38: Resulta da Consulta 6 - Mês de Julho

Consulta 7

"Mostre as áreas de plantação e quantidade de milho em municípios com área maior ou igual a 250Km2, do estado da Paraíba, no ano de 2009".

A novidade desta consulta com relação às anteriores é a utilização da função espacial Área para restringir membros da dimensão espacial Município. Esta restrição é ilustrada na Figura 4.39.

O resultado da consulta 7 é apresentado na Figura 4.40, a qual ilustra as plantações de milho nos municípios com área maior ou igual a 250Km2.

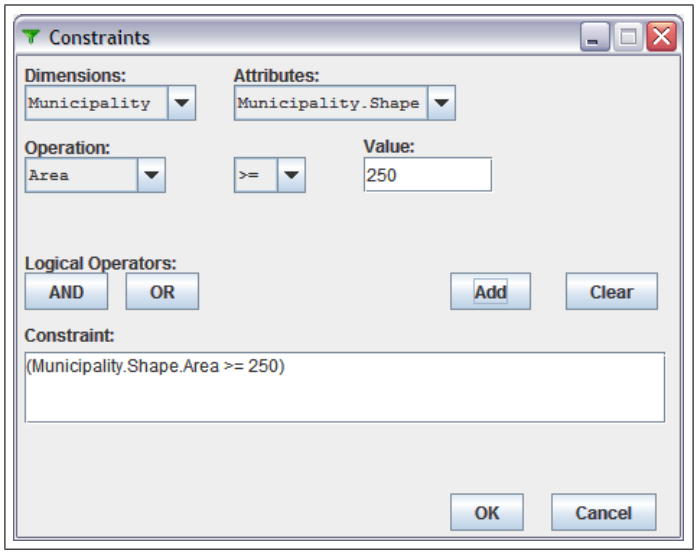

Figura 4.39: Restrição aplicada às áreas das geometrias dos municípios

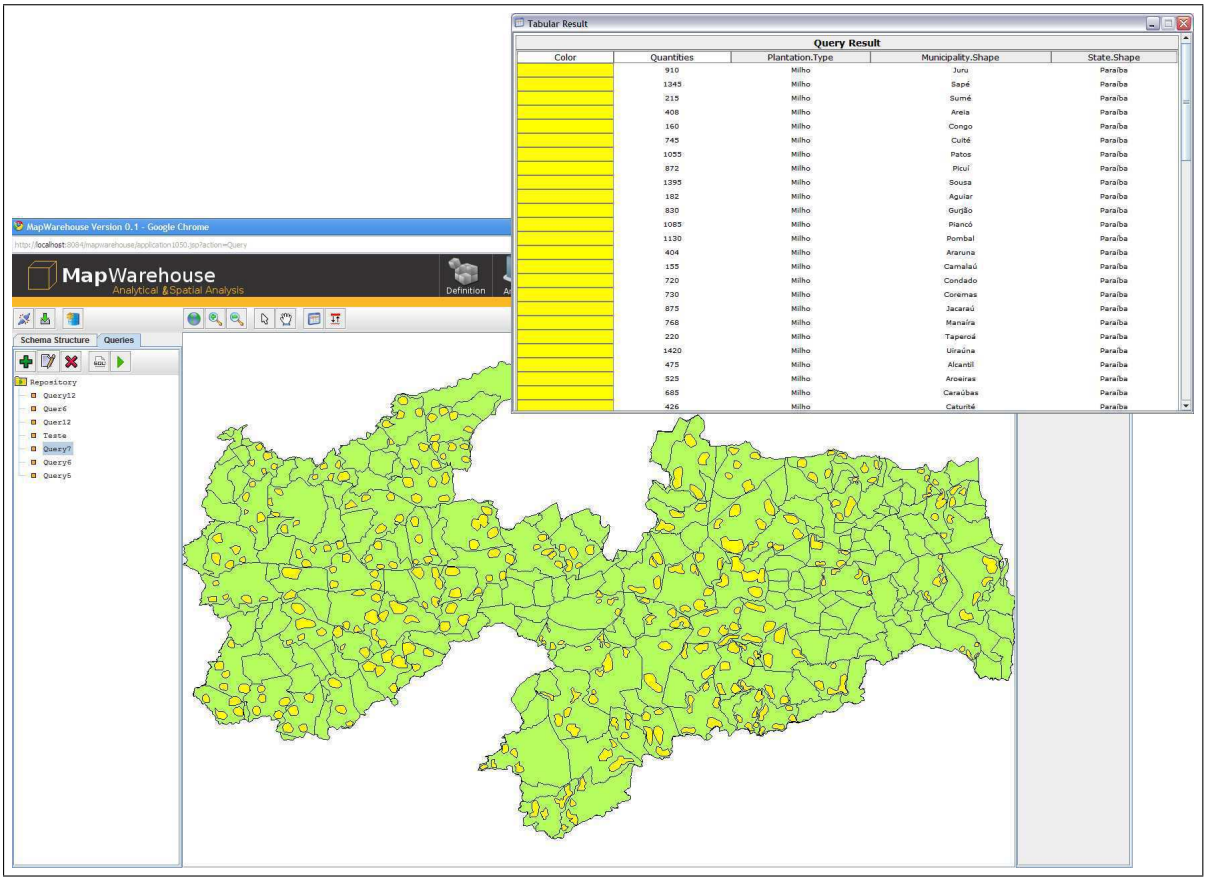

Figura 4.40: Resultado da consulta 7

Consulta 8

"Mostre as áreas de plantação e quantidades plantadas nos municípios que estão situados num raio de 50Km do município de Alagoa Seca, durante Março de 2009".

A inovação desta consulta com relação às anteriores é a restrição com a operação espacial *Distance*. Dessa forma, além da seleção das medidas e atributos das dimensões, foi definida a restrição ilustrada na Figura 4.41. Assim como em consulta anteriormente apresentadas, a seleção da geometria do município de Alagoa Seca foi realizada a partir da interface gráfica de Janela Espacial.

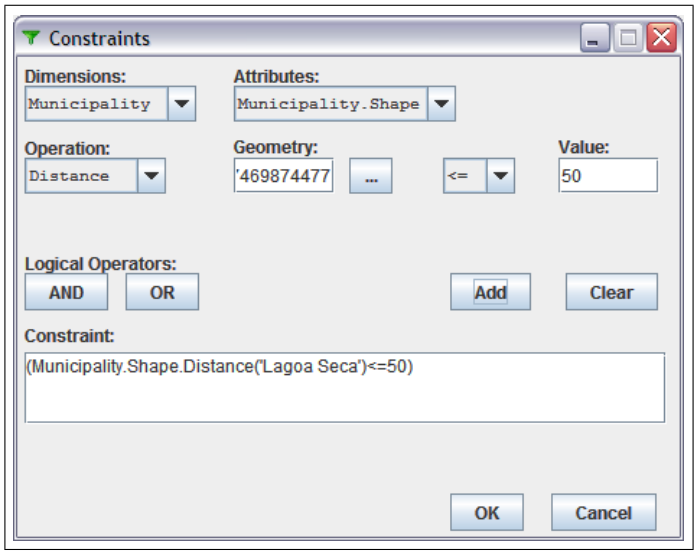

Figura 4.41: Restrição de distância aplicada aos membros da dimensão Município

O resultado pode ser observado na Figura 4.42. Nota-se que foi gerado um *buffer* de raio igual a 50 quilômetros ao redor da geometria do município de Alagoa Seca, dando maior destaque ao resultado da consulta.

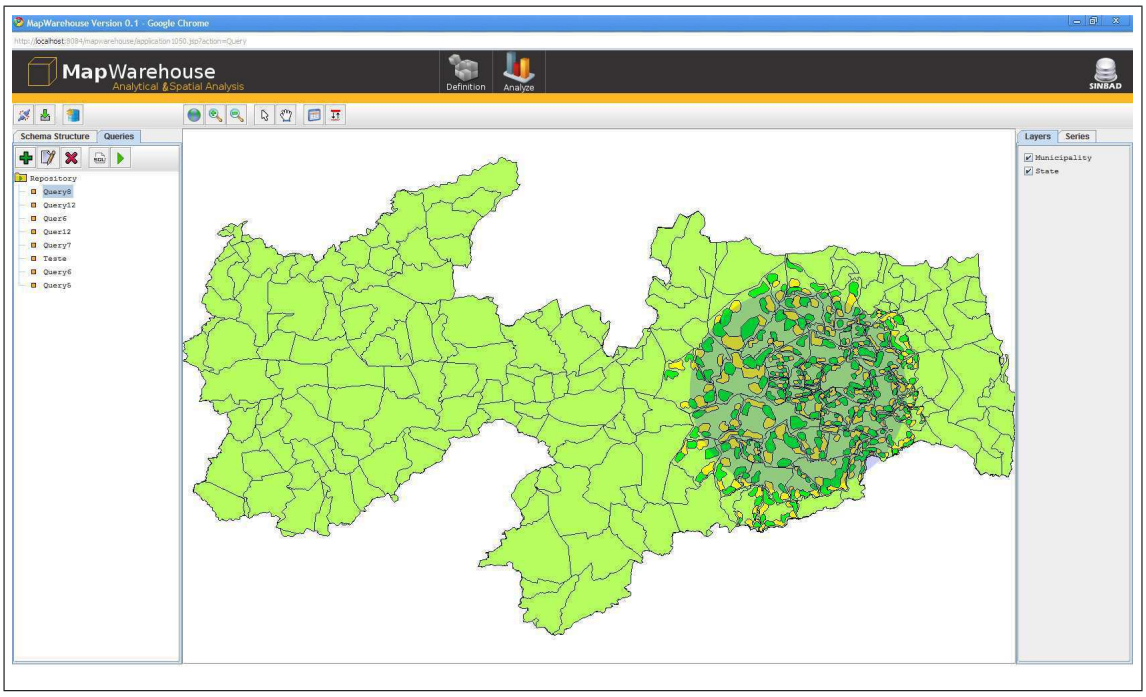

Figura 4.42: Resultado da consulta 8

Consulta 9

"Quais os 10 municípios que mais plantaram Milho ou Feijão no ano de 2009".

Esta consulta é um exemplo de consulta cujo resultado não apresenta dados espaciais, sendo utilizado a forma tabular para apresentação de resultados aos usuários. A inovação desta consulta é a utilização da funcionalidade *Ranking*. O *Ranking* permite ordenar e limitar o resultado de uma consulta. A utilização do *Ranking* é ilustrada na Figura 4.43.

A partir do resultado da consulta 9, ilustrado na Figura 4.44, foi realizado uma operação de *Roll-up* do nível Município para o nível Região. O resultado do *Roll-up* é ilustrado na Figura 4.46.

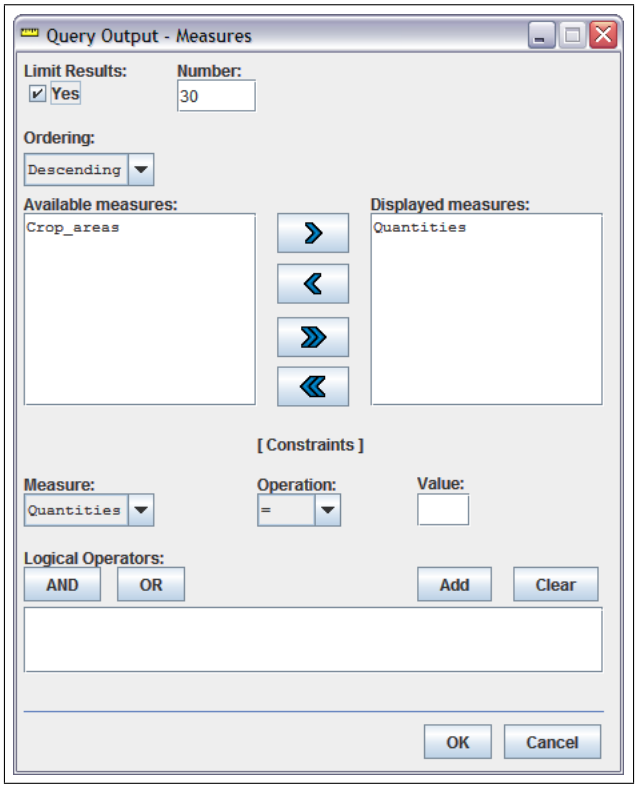

Figura 4.43: Ranking

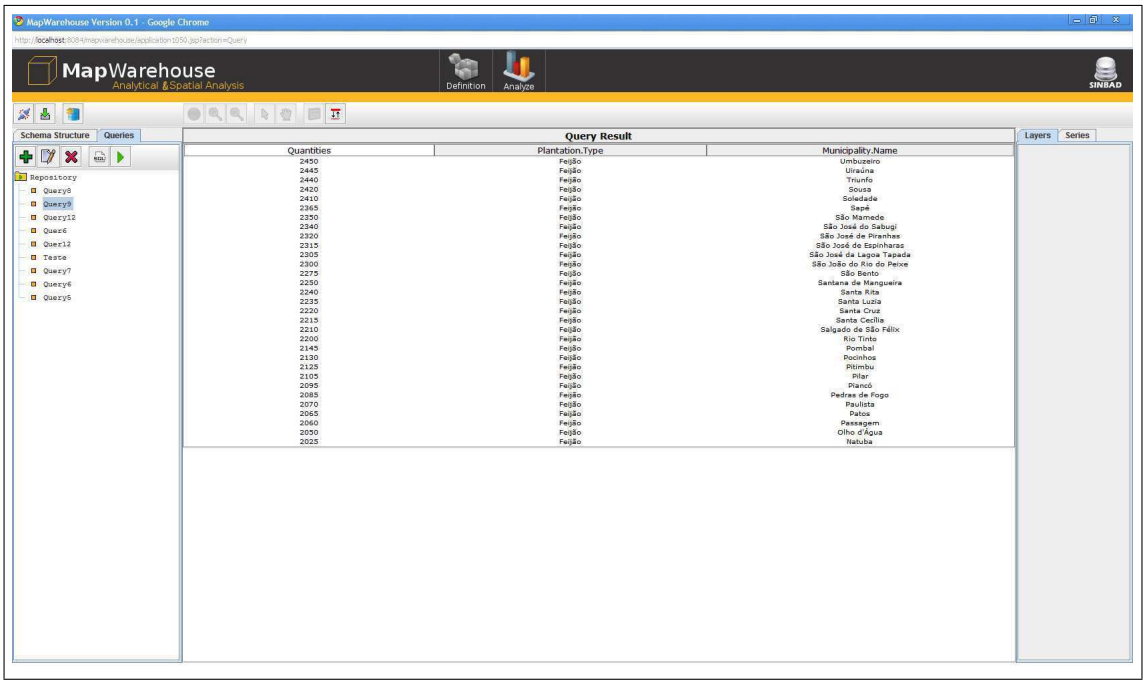

Figura 4.44: Resultado da consulta 9

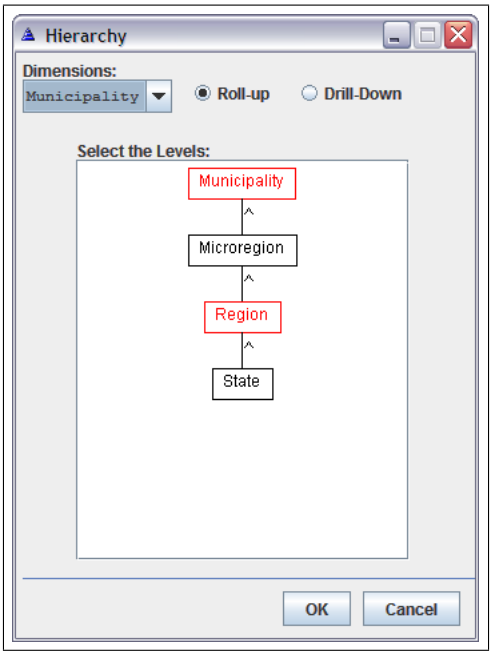

Figura 4.45: Roll-up do nível Município até Região

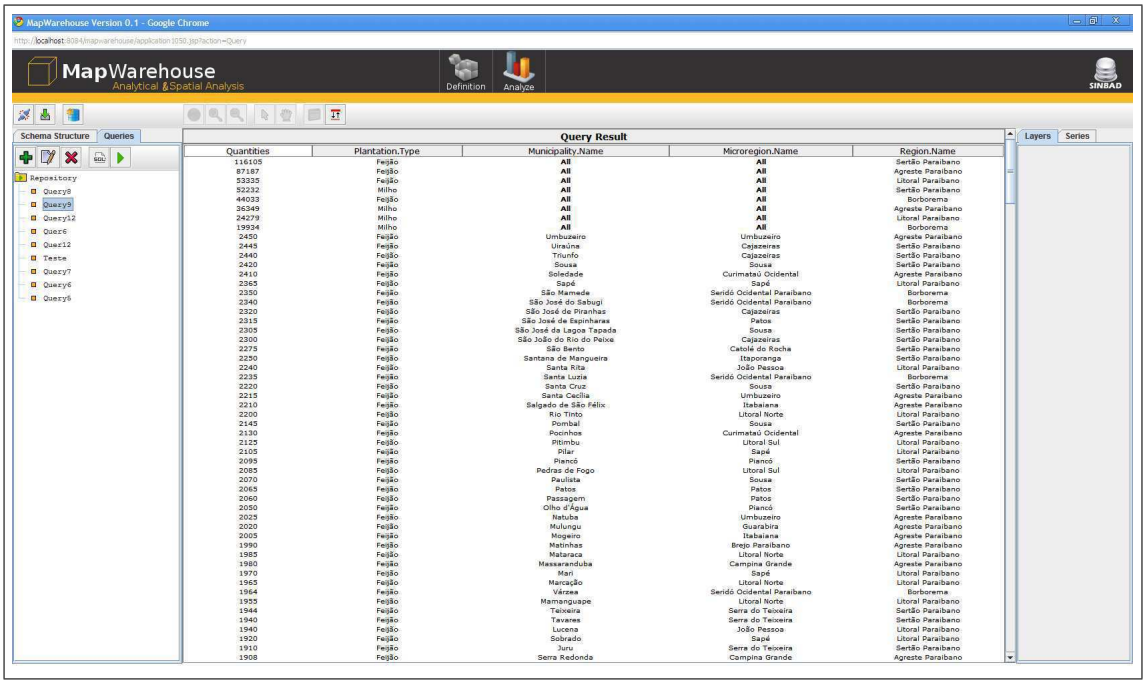

Figura 4.46: Resultado do Roll-up

4.2 Estudo de Caso: SAMU/Aracaju

O Serviço de Atendimento Móvel de Urgência (SAMU/192) é um programa que tem como finalidade prestar socorro à população em casos de emergência. Atendendo 24 horas por dia, o SAMU realiza atendimentos emergenciais de diversas natureza. Os pacientes são atendidos por ambulâncias de suporte básico ou avançado (UTI móvel) em residências, locais de trabalho e vias públicas. Uma aplicação de DW Espacial aplicado ao SAMU servirá como um termômetro da saúde do município, sendo possível analisar as solicitações de atendimentos considerando sua natureza, o tipo de ambulância utilizada, o horário de atendimento e sua localização.

4.2.1 Esquema Conceitual da aplicação Samu

A aplicação exemplo foi implementada segundo o modelo de dados da Figura 4.47, utilizando o SGBD PostgreSQL com sua extensão espacial: Postgis. Como observado no modelo, o fato a ser analisado é a ocorrência (*Ocurrence*), ou seja, uma solicitação para atendimento feita por algum usuário de saúde do município. As medidas que representam os fatos são: Vítimas (*Victims*), visto que uma ocorrência pode envolver mais de uma pessoa, e a Localização (*Location*), um ponto georreferenciado representando a localização exata do chamado.

Para caracterizar os fatos, o modelo prevê as dimensões Tipo de Chamado (*CallType*), que representa a natureza do chamado (e.g. Clínica, Traumática, etc.), Ambulância (*Ambulance*), podendo ser de dois tipos: USB (Unidade de Suporte Básico) ou USA (Unidade de Suporte Avançado), a dimensão temporal na granularidade minuto (*Minute* < *Hour* < *Day* < *Month* < *Year*), além da dimensão espacial Logradouro (*Street Address*) com a hierarquia Logradouro < Bairro < Cidade (*Street Address* < *District* < *City*).

Com relação à dimensão espacial Logradouro, entendemos que a granularidade ideal seria o nível Lote (ou propriedade), visto que o atendimento emergencial realizado pelo SAMU também acontece em residências, locais de trabalho, além de outras localidades. Além disso, entendemos que a dimensão espacial deveria possuir duas hierarquias como ilustrado na Figura 4.48, visto que um logradouro não está necessariamente contido em um bairro (e.g. Avenidas que cortam mais de um bairro). Entretanto, decidiu-se representar a

dimensão espacial como no modelo da Figura 4.47 por dois motivos: (i) Não foi possível conseguir em tempo hábil as geometrias referentes aos lotes da cidade de Aracaju; (ii) as geometrias dos logradouros estavam segmentadas por bairro. Por exemplo: a Av. Beira Mar foi dividida em várias geometrias, cada uma representando trechos dos bairros que cruza.

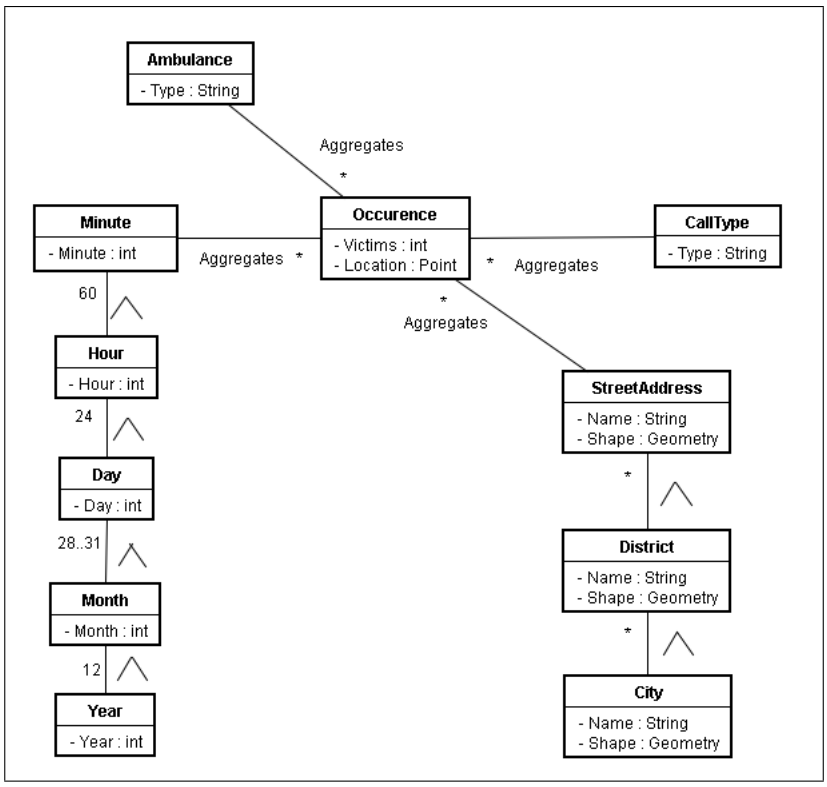

Figura 4.47: Diagrama UML - SAMU/Aracaju

4.2.2 Definição do Esquema

Assim como no estudo de caso anterior, o esquema da aplicação do SAMU foi definido a partir de interfaces gráficas do módulo de definição de esquemas do Mapwarehouse. A seguir, serão descritos resumidamente, os passos para definição deste esquema.

Medidas

As medidas da aplicação do SAMU, isto é, Vítimas (*Victims*) e Localização (*Location*), foram definidas a partir das interfaces gráficas ilustradas na Figura 4.49. Pode-se observar que ambas as medidas são atômicas.

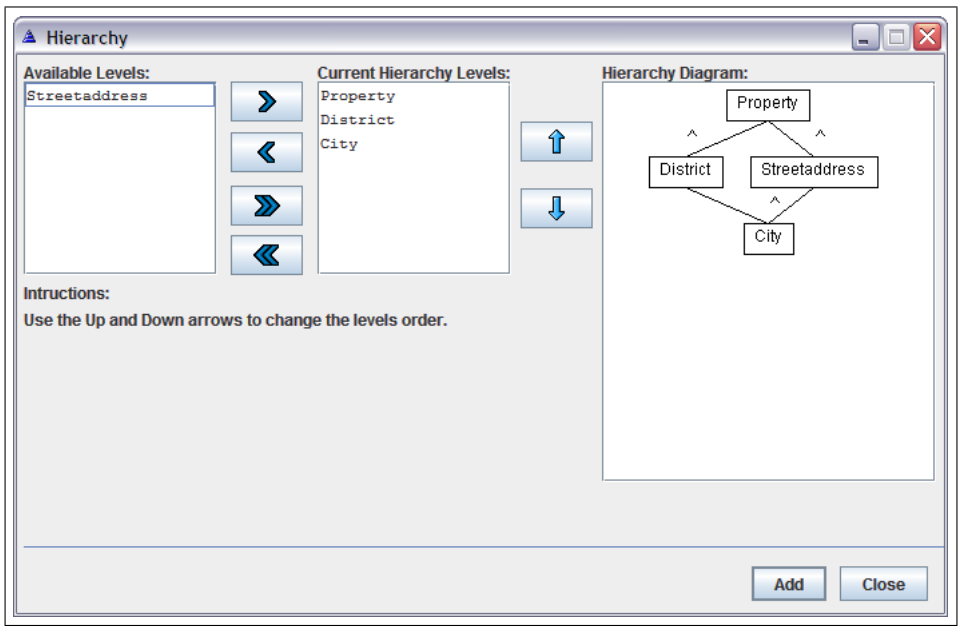

Figura 4.48: Hierarquias Ideal para a Dimensão Espacial

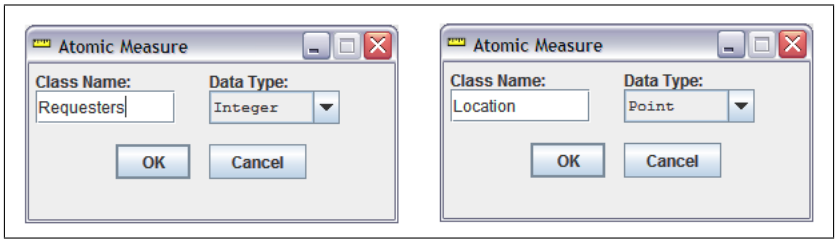

Figura 4.49: Definição das medidas

Dimensões e Hierarquias

Cada dimensão do esquema foi definida utilizando o *Wizard* de definição de dimensões apresentado na subseção 3.3.1. A Figura 4.50 ilustra a etapa de definição dos atributos do nível Cidade (*City*) na dimensão Logradouro (*StreetAddress*). Após a definição de cada nível da dimensão Logradouro, o Mapwarehouse gerou a hierarquia apresentada na Figura 4.51.

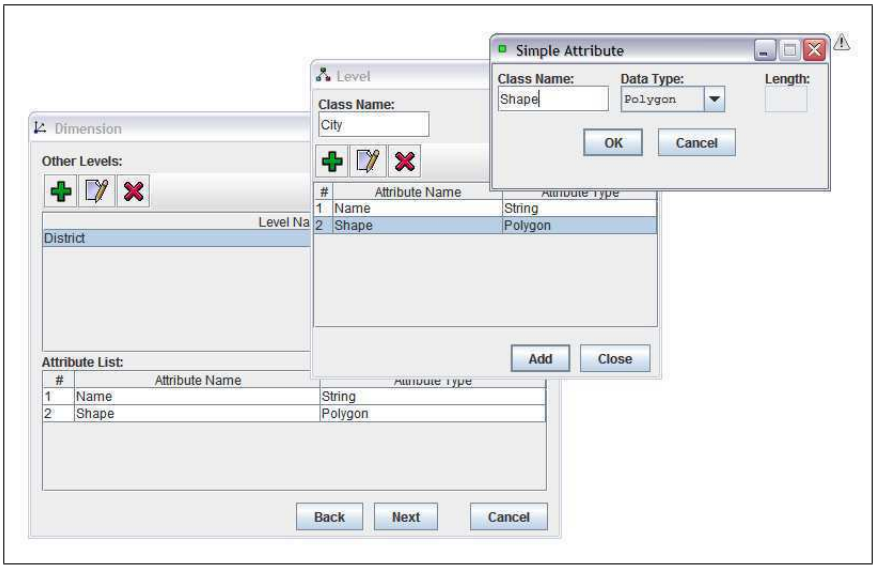

Figura 4.50: Definição da dimensão Logradouro (*StreetAddress*)

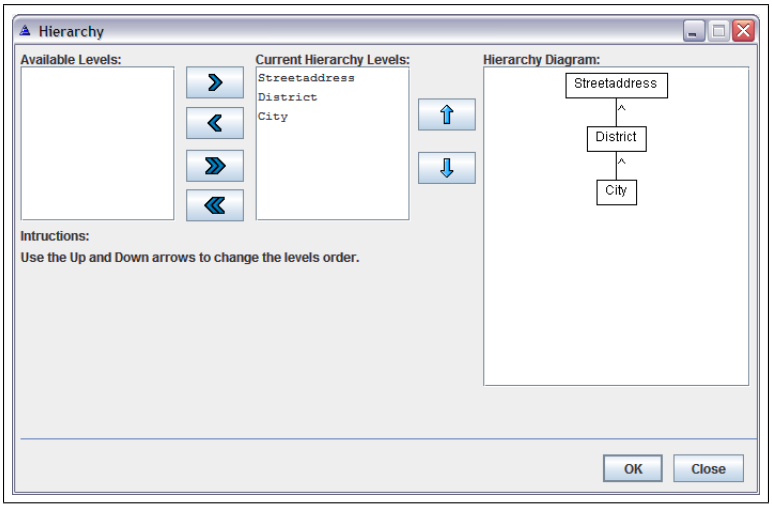

Figura 4.51: Diagrama da hierarquia Logradouro

Geração de Esquema Lógico

Uma vez que todas as dimensões e medidas foram definidas, o esquema da aplicação ficou como ilustrado na Figura 4.5. Foi gerado, então, o esquema lógico para o SGBD PostgreSQL (vide Figura 4.53).

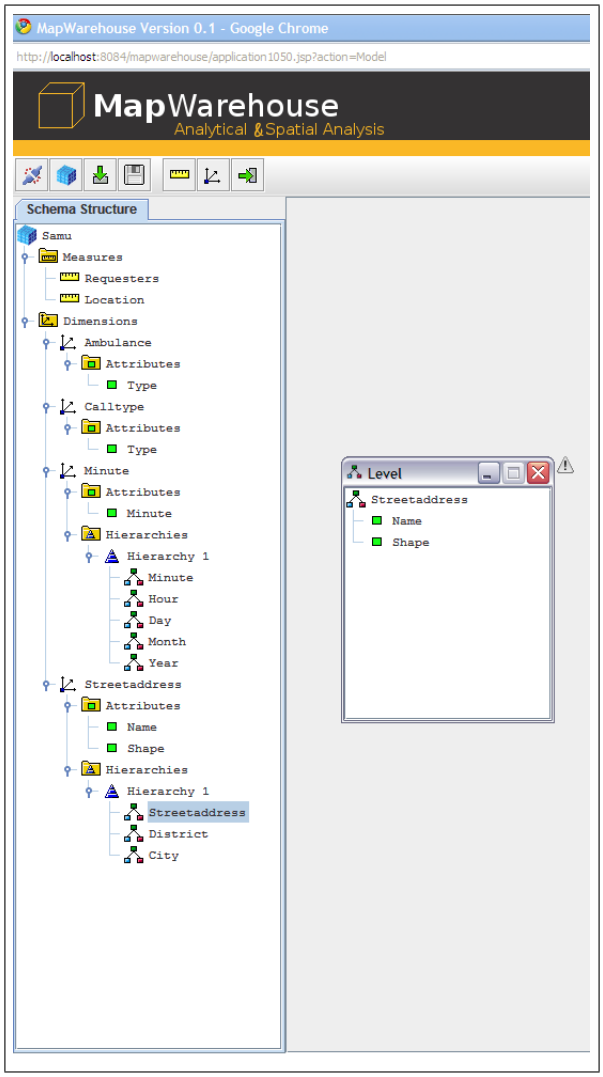

Figura 4.52: Estrutura do esquema

4.2.3 Consultas Exemplo

A seguir serão apresentadas duas consultas visando demonstrar a aplicação SOLAP do SAMU/Aracaju.

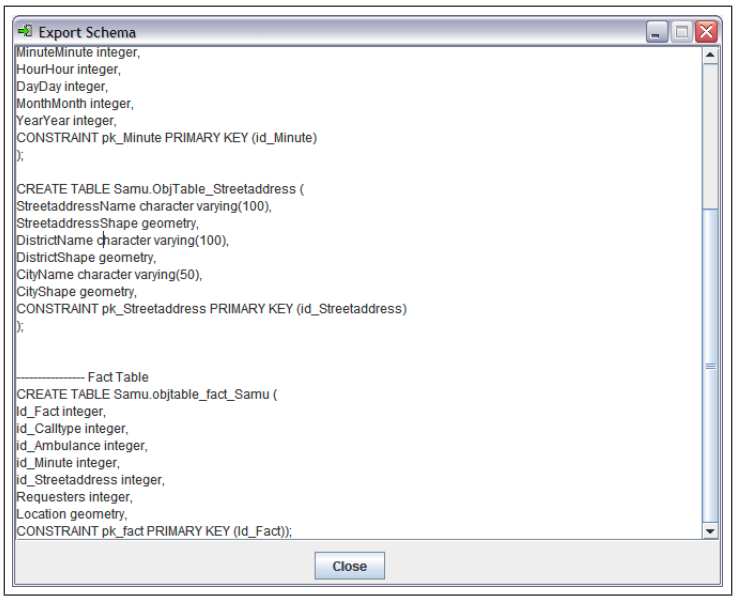

Figura 4.53: Esquema lógico para PostgreSQL

Consulta 1

"Exiba o total de vítimas e as localizações dos chamados de tipo traumático e clínico que ocorreram dentro de uma janela espacial selecionada no mapa". Esta consulta demonstra o uso da janela espacial, uma funcionalidade que permite ao usuário selecionar uma área do mapa e utilizá-la como operando em operações espaciais.

Primeiramente, foram selecionadas as medidas a serem exibidas, isto é, vítimas (*Victims*) e localização (*Location*), como ilustrado na Figura 4.54.

Em seguida foi definida a janela espacial a partir da interface gráfica ilustrada na Figura 4.55. O resultado da seleção da janela espacial é ilustrado na Figura 4.56, a qual apresenta a restrição *Location.Inside(MBR)* dentro do campo de restrições.

Após a definição da janela espacial, foram selecionadas os atributos das dimensões a serem exibidos, isto é, as geometrias dos Bairros (*District*) e Logradouros (*StreetAddress*) e o tipo de chamado (*CallType*). Além disso, foram definidas as restrições para a dimensão tipo de chamado, ou seja, apenas os chamados de natureza traumática e clínica. Estas tarefas podem ser observadas nas Figuras 4.57 e 4.58, respectivamente.

A Figura 4.59 apresenta uma interface gráfica com o comando SQL gerado pelo Mapwarehouse. Esta interface gráfica é acionada pelo botão "SQL", destacado em vermelho.

O resultado da consulta 1 é ilustrado na Figura 4.60.

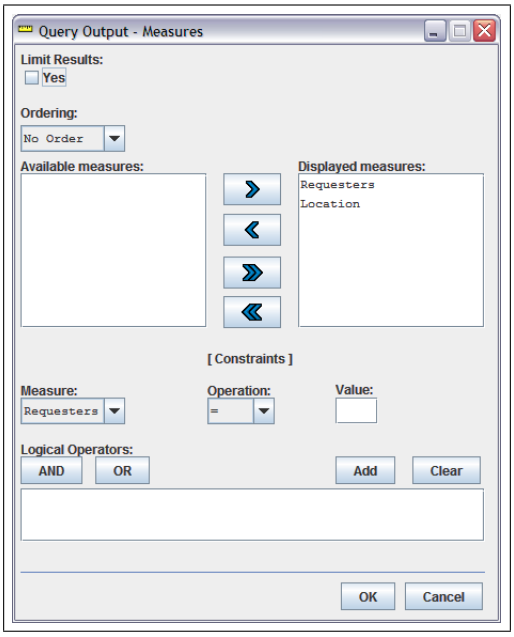

Figura 4.54: Seleção das medidas

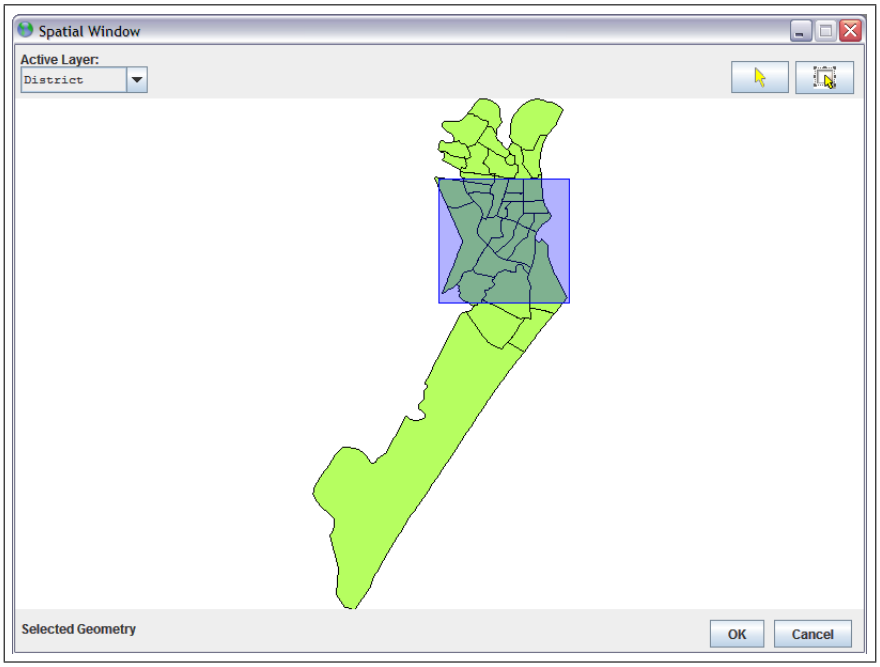

Figura 4.55: Definição da Janela Espacial

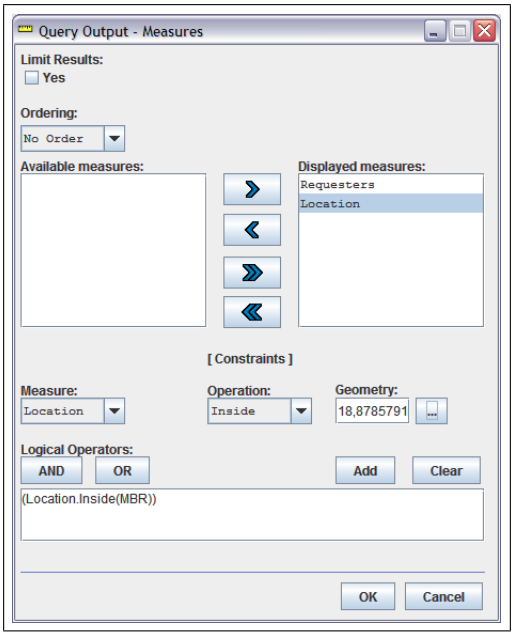

Figura 4.56: Definição da operação *Inside*

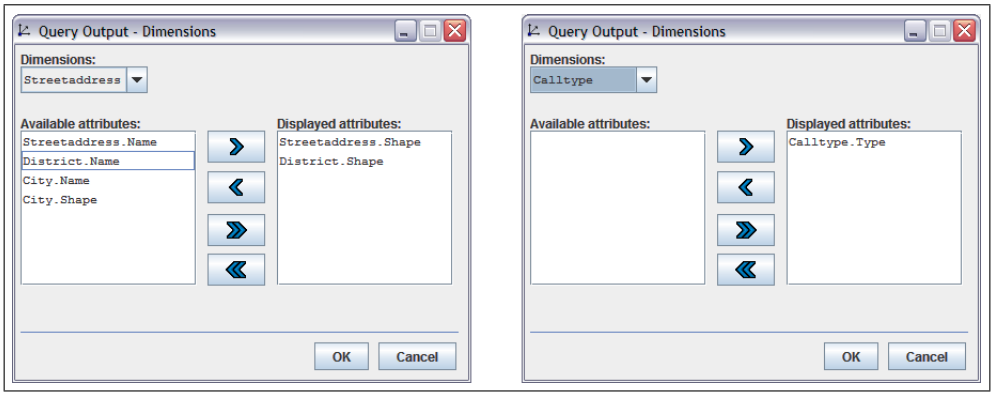

Figura 4.57: Seleção dos atributos das dimensões

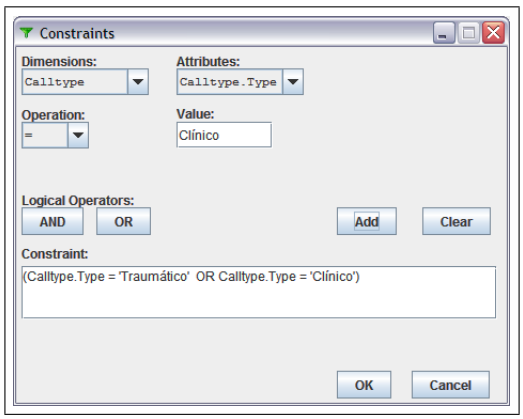

Figura 4.58: Definição das restrições

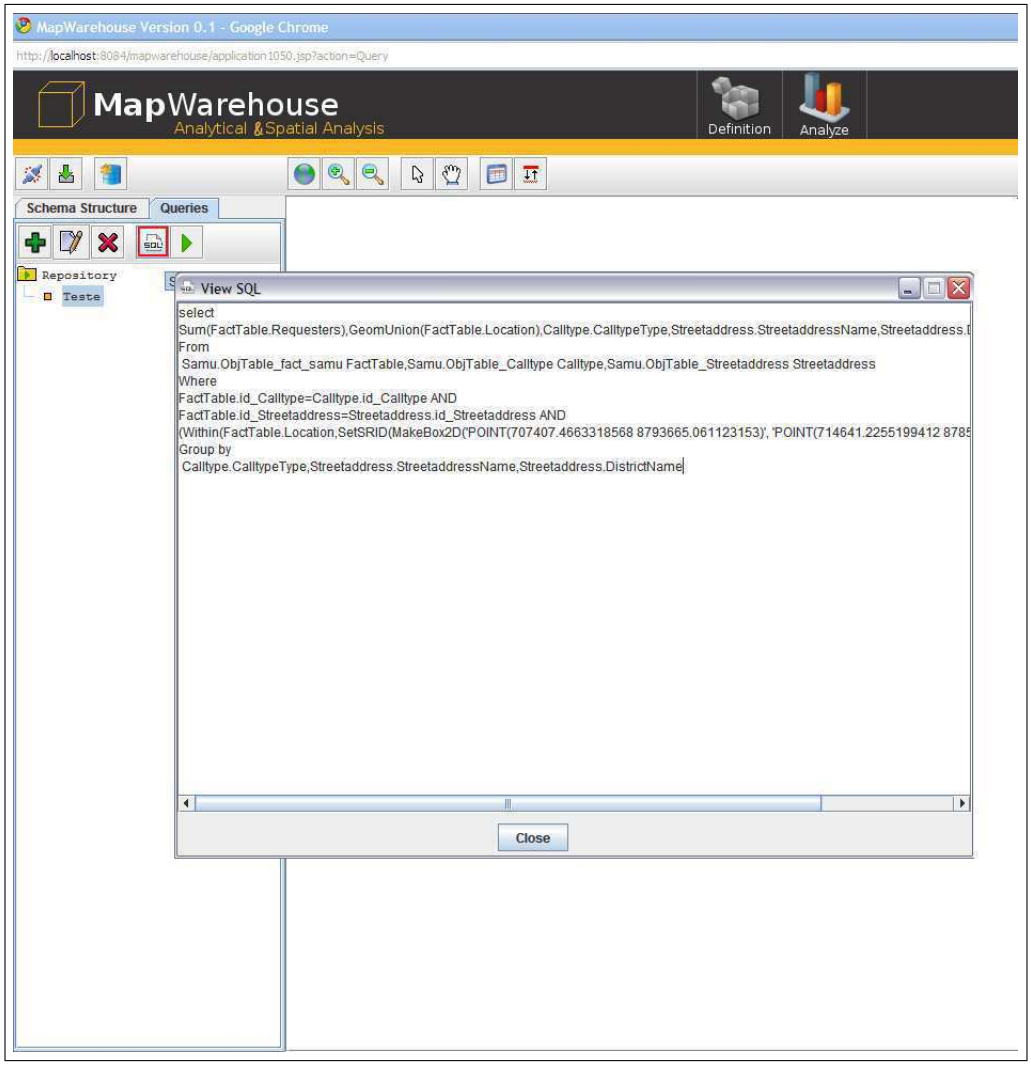

Figura 4.59: Consulta SQL do Postgis

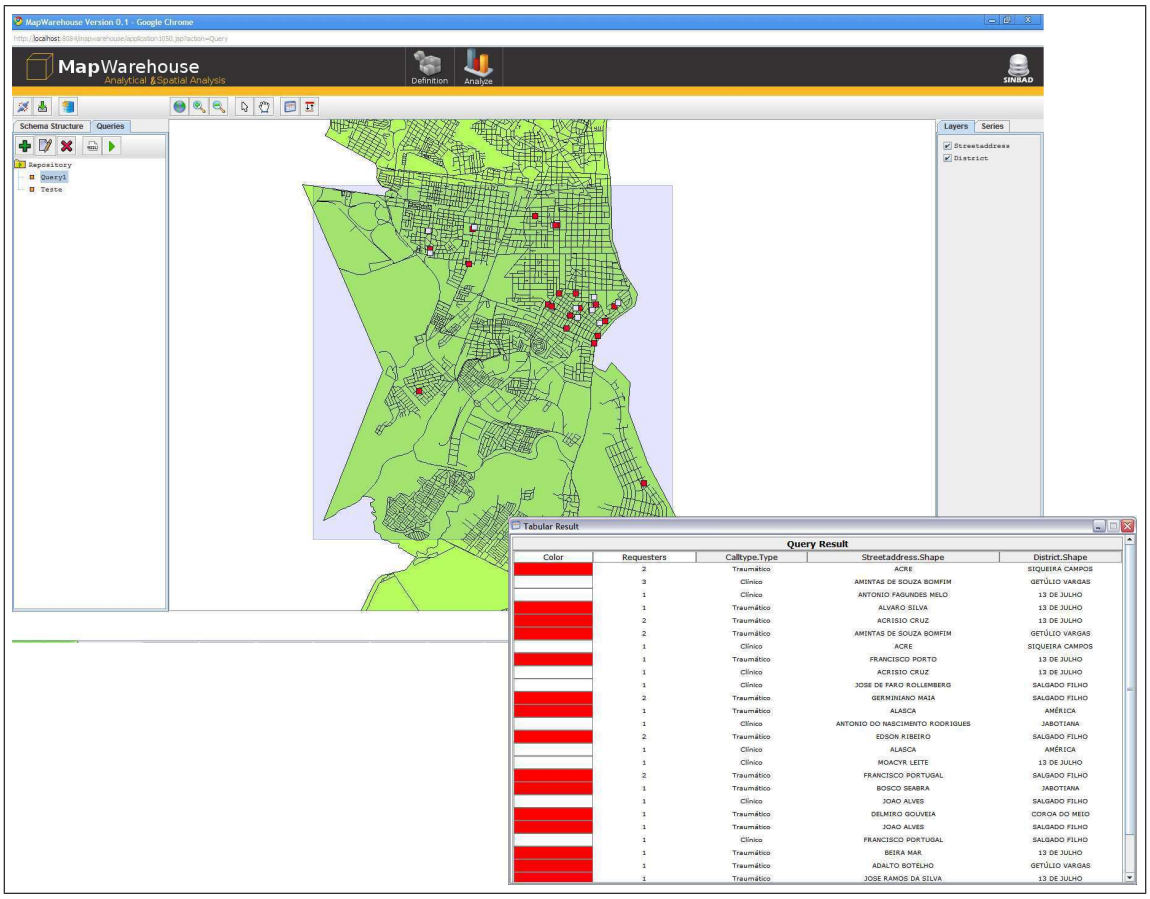

Figura 4.60: Resultado da consulta 1

Consulta 2

"Exiba o total de vítimas e as localizações dos chamados que ocorreram a uma distância de até 2 quilômetros do bairro Centro". Esta consulta tem como objetivo demonstrar a operação espacial que envolve distância, representada pela classe *SpatialBinaryScalar* do metamodelo da Figura 3.12.

Inicialmente, foram selecionadas para serem exibidas as medidas vítimas e localização. Logo após, foram selecionados os atributos das dimensões. Estas tarefas estão ilustradas nas Figuras 4.61 e 4.62, respectivamente.

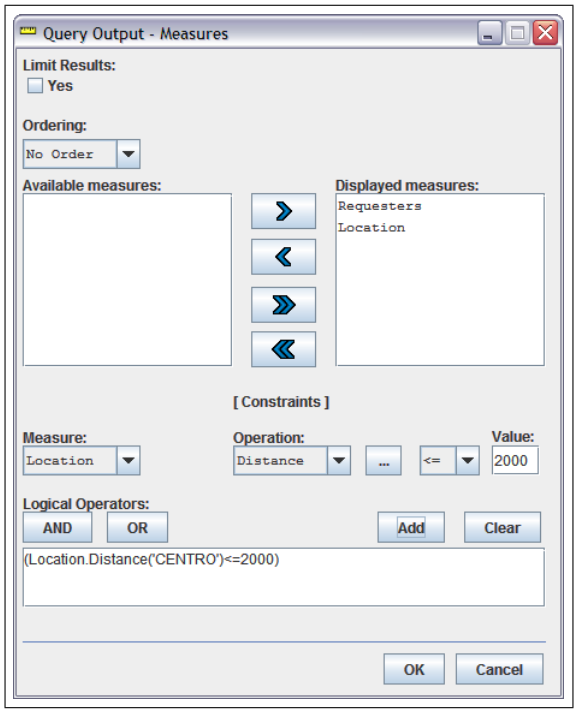

Figura 4.61: Escolha das medidas

A restrição para a localização dos chamados que ocorreram a uma distância de 2 quilômetros do bairro Centro é uma restrição aplicado aos fatos. Por isso, foi definida na interface de definição de medidas como pode ser observado na Figura 4.61. Para selecionar a geometria do bairro Centro, foi utilizada a interface gráfica janela espacial, ilustrada na Figura 4.63.

O resultado da consulta 3 pode ser observado na Figura 4.64. Nota-se que foi gerado um *buffer* de raio igual a 2 quilômetros ao redor da geometria do bairro Centro, dando maior destaque ao resultado da consulta.

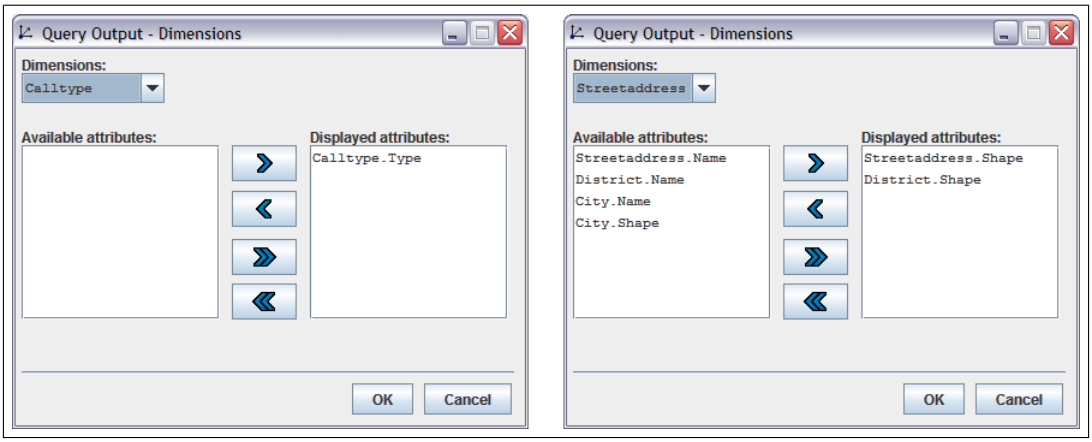

Figura 4.62: Escolha dos atributos das dimensões

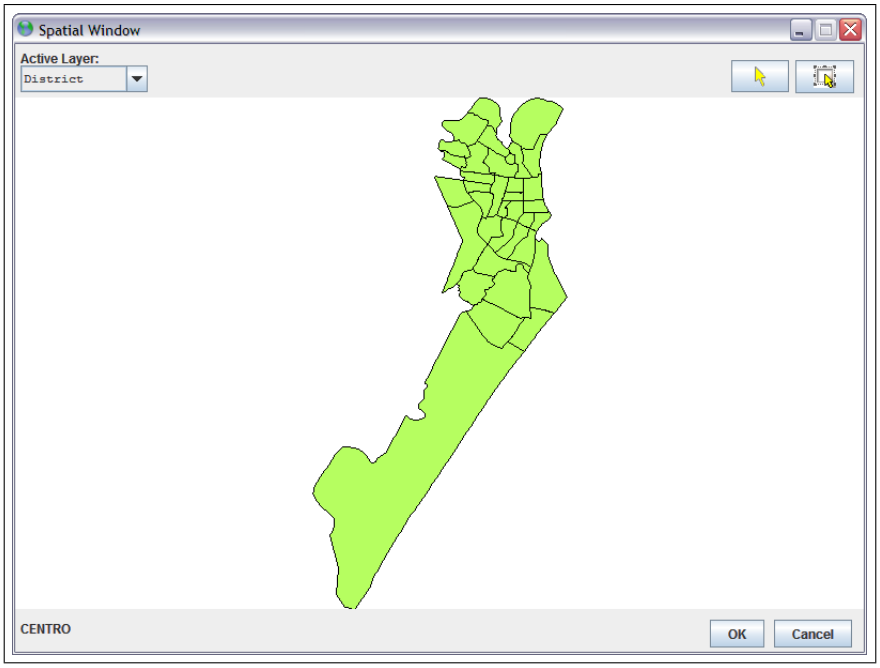

Figura 4.63: Seleção da geometria do bairro Centro

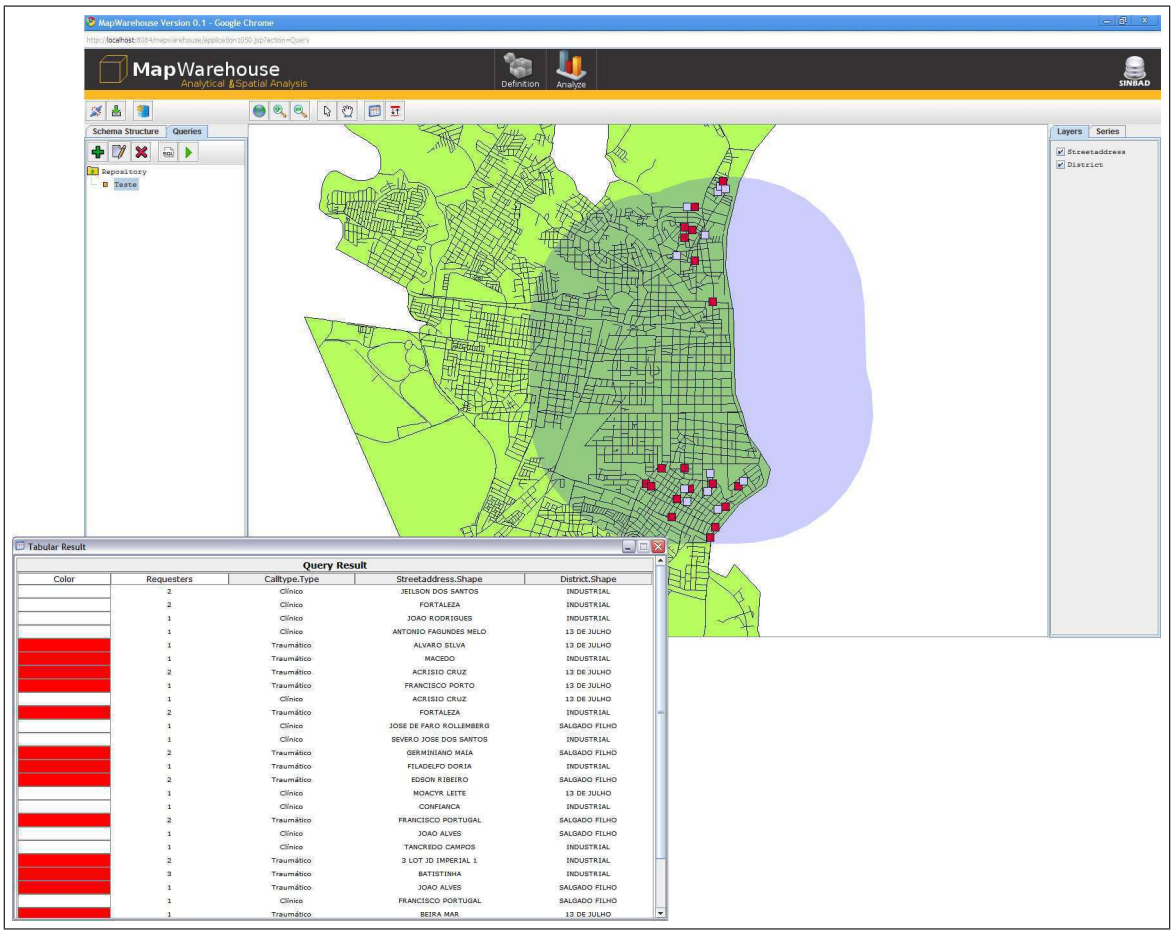

Figura 4.64: Resultado da Consulta 2

Capítulo 5

Conclusões

As pesquisas relacionadas a *Data Warehouse* Espacial e ferramentas SOLAP são bastante recentes. Em sua maioria, os trabalhos se concentram na definição de novos modelos de dados multidimensionais que lidam com dados espaciais, linguagens de consulta a DW espacial e ferramentas SOLAP. No que se refere a ferramentas SOLAP, os trabalhos têm como objetivo integrar características dos SIGs com técnicas OLAP, provendo um ambiente único para análise analítico-espacial. A partir da análise dos trabalhos apresentados no Capítulo 2, foram observadas algumas questões em aberto que motivaram esta dissertação. Entre elas:

- A ausência de uma ferramenta que permita a realização de consultas analítico-espaciais de forma intuitiva.
- Ferramenta SOLAP independente de SGBD específico: Em sua maioria, as propostas analisadas foram implementadas para SGBD específicos. Mesmo as que afirmam serem independentes de SGBD, demandariam um esforço considerável para serem utilizadas em um SGBD distinto ao que foi implementado.

Diante destas questões, este trabalho apresentou no Capítulo 3, uma nova proposta de ferramenta SOLAP chamada Mapwarehouse, cujo principal objetivo é permitir que usuários não especialistas, isto é, usuários gestores, definam seus esquemas para DW Espacial e realizem consultas analítico-espaciais de forma intuitiva. Para isto, o Mapwarehouse disponibiliza um conjunto de interfaces gráficas que permitem a definição esquemas e consultas baseados em metamodelos conceituais.

Uma característica que julgamos essencial em uma ferramenta SOLAP é a possibilidade de realização de consultas que envolvam tanto operações OLAP como também operações espaciais. Neste sentido, nosso trabalho tem um diferencial com relação aos trabalhos apresentados no Capítulo 2 que contemplam esta característica: a possibilidade de definir este tipo de consulta a partir de interfaces gráficas. Isto permite que gestores possam realizar consultas a DW espaciais sem a necessidade de conhecer detalhes da sintaxe da linguagem de consulta.

Para solucionar a questão da dependência de fabricante de SGBD, o Mapwarehouse foi implementado segundo a filosofia de *framework*. Em sua arquitetura foram criados alguns pontos de extensão que permitem estendê-lo para que seja utilizado com diferentes SGBDs, sendo esta uma grande contribuição do nosso trabalho. Outra contribuição do nosso trabalho são as implementações das extensões do Mapwarehouse para os SGBDs Oracle e PostgreSQL, também apresentadas no Capítulo 3. Estas extensões permitem a criação de aplicações SOLAP utilizando dois grandes bancos de dados dos segmentos de SGBDs.

No Capítulo 4, foram apresentados dois estudos de caso de aplicações SOLAP, objetivando demonstrar a utilização do Mapwarehouse nos SGBDs Oracle e PostgreSQL, além de demonstrar as tarefas de definição de esquemas e definição de consultas a partir de interfaces gráficas. Os estudos de caso permitiram concluir que o Mapwarehouse atendeu ao seus requisitos de forma satisfatória e permite a realização de consultas analítico-espaciais de vários níveis de complexidade. Entretanto, algumas sugestões apresentadas na seção seguinte, podem ser feitas para melhorar o Mapwarehouse e dotá-lo de maior poder de consulta e análise.

Visando um comparativo de nossa proposta com os trabalhos apresentados no Capítulo 2, apresentamos novamente a tabela 5.1 com as características dos trabalhos analisados e as características do *framework* Mapwarehouse.

Diante do que foi apresentado nesta dissertação, percebe-se que o Mapwarehouse contempla todas as características apresentadas na tabela. O Mapwarehouse foi implementado segundo a abordagem integrada de DW Espacial, na qual dados convencionais e espaciais são armazenados em um único ambiente. Seu metamodelo multidimensional permite a modelagem de medidas como objetos complexos, inclusive utilizando dados espaciais. Sua interface gráfica, desenvolvida com tecnologia aberta Java, é totalmente Web e permite os

usuários definam esquemas e consultas em alto nível de abstração. Além disso, observou-se que o Mapwarehouse é independente de domínio de aplicação e independente de SGBD.

	JMap	GeoWOlap	SOVAT	Piet	GolapWare	Mapwarehouse	*framework* Mapwarehouse
Abordagem Integrada		x			x	x	x
Medidas Espaciais	x	x		x	x	x	x
Medidas Complexas		x					x
Operadores Espaciais					x	x	x
Interface Web	x	x			x	x	x
Consulta Conceitual	x	x	x			x	x
Definição de Esquemas					x		x
Tecnologia Aberta	x				x		x
Extensível - Domínio	x	x			x		x
Extensível - SGBD	x						x

Tabela 5.1: Características dos trabalhos relacionados

5.1 Trabalhos Futuros

No decorrer desse trabalho foram observadas algumas questões que dão margem para a realização de novos trabalhos na área. A seguir, serão apresentadas essas questões na forma de sugestões para trabalhos futuros.

ETL Espacial

Com observado, o Mapwarehouse permite a geração de esquemas lógicos e a realização de consultas a *Data Warehouses* Espaciais definidos a partir destes esquemas. Entretanto, a ferramenta não considerou uma tarefa intermediária que é essencial: a carga dos dados no *Data Warehouse* Espacial. Para que o ciclo Definição-Carga-Consulta fique completo é necessário que a ferramenta disponha de mecanismos para realização do processo ETL. Neste sentido, sugerimos como trabalho futuro o desenvolvimento de um Sistema de ETL Espacial que possa ser integrado ao Mapwarehouse.

Módulo de *Dashboard*

Uma funcionalidade interessante seria o desenvolvimento de um módulo de geração de *Dashboards*. Um *Dashboard* é um conjunto de indicadores de desempenho que servem como termômetro para o negócio de uma empresa. *Dashboards* poderiam ser associados a consultas do Mapwarehouse, o que permitiria avaliar o desempenho do negócio com relação a indicadores pré-estabelecidos.

Otimização de Consultas

A agregação de dados espaciais é uma tarefa bastante custosa tanto para o SGBD quanto para a aplicação SOLAP - que deve renderizar os dados espaciais agregados. Neste sentido, trabalhos futuros podem ser conduzidos buscando a otimização de consultas SOLAP.

Outras Melhorias

Observamos ainda, as seguintes melhorias que poderiam ser feitas no Mapwarehouse:

- Melhoraria na interface de representação de resultados de consultas na forma tabular, visando torná-la mais intuitiva.

- Adicionar novos tipos de gráficos sobrepostos a mapas.
- Modificar a tela de definição de Saída de Medidas (*Measure Output*) para que seja possível escolher as funções de agregação que serão aplicadas às medidas agregadas.

Bibliografia

[1] Ramez Elmasri , Shamkant B. Navathe. *Fundamentals of Database Systems, 2nd Edition*. Benjamin/Cummings, 1994.

[2] Vidette Poe, Stephen Brobst, Patricia Klauer. *Building a Data Warehouse for Decision Support*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[3] Ralph Kimball, Margy Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[4] Ralph Kimball , Joe Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleanin*. John Wiley & Sons, 2004.

[5] E. F. Codd, S. B. Codd, C. T Salley. *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate*. 1993.

[6] W. H. Inmon. *Building the Data Warehouse,3rd Edition*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[7] Carl Franklin. An introduction to geographic information systems: linking maps to databases. *Database*, 15(2):12–21, 1992.

[8] S. Shekhar and S. Chawla. *Spatial Databases - A Tour*. Prentice Hall, 2003.

[9] Y. Bédard, T. Merrent, J. Han. Fundamentals of spatial data warehousing for geographic knowledge discovery., 2001.

[10] Jiawei Han, Nebojsa Stefanovic, Krzysztof Koperski. Selective materialization: An efficient method for spatial data cube construction. In *PAKDD'98: Proceedings of the*

Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 144–158, 1998.

[11] Gilberto Câmara, Marco A. Casanova, Andrea S. Hemerly, Geovane C. Magalhães, Claudia M. B. Medeiros. *Anatomia de Sistemas de Informação Geográfica*. Editora Unicamp, X Escola de Computação, Campinas, São Paulo, 1996.

[12] S. Rivest, Y. Bédard, P. Marchand. Toward better suppport for spatial decision making: Defining the characteristics of spatial on-line analytical processing (SOLAP). 2001.

[13] S. Bimonte, A. Tchounikine, M. Miquel. Spatial OLAP: open issues and a web based prototype. In *10th AGILE International Conference on Geographic Information Science (short paper)*, 2006.

[14] Y. Bédard. Spatial OLAP. In *2e Forum annuel sur la R-D, Géomatique VI: Un monde accessible*, Canadian Institute of Geomatics, Montreal, Canada, 1997.

[15] Sandro Bimonte, Pascal Wehrle, Anne Tchounikine, Maryvonne Miquel. GeWOlap: A web based Spatial OLAP proposal. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops (2)*, volume 4278 of *Lecture Notes in Computer Science*, pages 1596–1605. Springer, 2006.

[16] Matthew Scotch and Bambang Parmanto. Sovat: Spatial olap visualization analysis tool. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 6*, page 142.2, Washington, DC, USA, 2005. IEEE Computer Society.

[17] Marcus Costa Sampaio, André Gomes de Sousa, Cláudio de Souza Baptista. 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, 2006. ACM.

[18] Ariel Escribano, Leticia Gomez, Bart Kuijpers, Alejandro A. Vaisman. Piet: a GIS-OLAP implementation. In *DOLAP '07: Proceedings of the ACM Tenth International Workshop on Data Warehousing and OLAP*, pages 73–80, New York, NY, USA, 2007. ACM.

[19] Joel da Silva, Ausberto S. Castro Vera, Anjolina Grisi de Oliveira, Robson do Nascimento Fidalgo, Ana Carolina Salgado, Valéria Cesário Times. Querying geographical data warehouses with GeoMDQL. In *SBBD*, pages 223–237, João Pessoa, Brasil, 2007.

[20] S. Rivest, Y. Bédard, M. J. Proulx, M. Nadeau, F. Hubert, J. Pastor. SOLAP: Mergin Business Intelligence with Geospatial Technology for Interactive Spatio-TemporalExploration and Analysis of Data. *Journal of International Society for Photogrammetry and Remote Sensing (ISPRS) - Advances in Spatio-temporal Analysis and Representation, Vol. 60, No. 1, pp.17-33. 2005.*

[21] Kheops tecnology. In: http://www.kheops-tech.com/en/jmap/solap.jsp, acessado em 16 de maio de 2009.

[22] Spatial OLAP Portal. In: http://spatialolap.scg.ulaval.ca, acessado em 16 de maio de 2009.

[23] S. Tchounikine A. Bimonte and M. Miquel. Towards a spatial multidimensional model. In DOLAP '05: Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP*, pages 39–46, New York, NY, USA, 2005. ACM.*

[24] Sandro Bimonte, Anne Tchounikine, Maryvonne Miquel. GeoCube, a Multidimensional Model and Navigation Operators Handling Complex Measures: Application in Spatial OLAP. In Tatyana Yakhno Erich Neuhold, editor, Fourth Biennial International Conference on Advances in Information Systems*, pages 100–109. Springer-Verlag, Oct 2006.*

[25] Oracle Spatial. In: http://www.oracle.com/technology/products/spatial/index.html, acessado em 26 de maio de 2009.

[26] JPivot Home. In: pivot.sourceforge.net, acessado em 26 de maio de 2009.

[27] Mapxtreme. In: http://www.mapxtreme.com, acessado em 26 de maio de 2009.

[28] Matthew Laurence Scotch. An OLAP-GIS System for Numerical-Spatial Problem Solving in Community Health Assessment Analysis. *PhD thesis, University of Pittsburgh, Pittsburgh, PA, USA, 2006.*

[29] Matthew Scotch, Bambang Parmanto, Valeria Monaco. Usability evaluation of the spatial OLAP visualization and analysis tool (SOVAT). Jornal of usability studies*, 2:76–95, 2007.*

[30] Postgis. In: postgis.refractions.net, acessado em 26 de maio de 2009.

[31] Joel da Silva. GeoMDQL: Uma linguagem de consulta geográfica e multidimensional. *PhD thesis, Centro de Informática, Universidade Federal de Pernambuco, Recife, PE, Brasil, 2008.*

[32] Rafael da Fonseca, Joel da Silva, Robson do Nascimento Fidalgo, and Valéria Cesário Times. GeoDWCASE: Uma ferramenta para projeto de data warehouses geográficos. In SBBD*, pages 223–237, João Pessoa, Brasil, 2007.*

[33] Mondrian OLAP Server. In: http://mondrian.pentaho.org/, acessado em 20 de maio de 2009.

[34] Open Jump - java-based and open source geographic information system. In: http://www.openjump.org/, acessado em 20 de maio de 2009.

[35] JRubik - Java OLAP client. In: http://rubik.sourceforge.net/, acessado em 20 de maio de 2009.

[36] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns. *Addison-Wesley, Boston, MA, January 1995.*

[37] Don Roberts, Ralph Johnson. Evolving frameworks: A pattern language for developing object-oriented frameworks. In Proceedings of the Third Conference on Pattern Languages and Programming. *Addison-Wesley, 1996.*

[38] Mohamed Fayad, Douglas C. Schmidt. Object-oriented application frameworks. Commun. ACM*, 40(10):32–38, 1997.*

[39] Open Geospatial Consortium. In: http://www.opengeospatial.org, acessado em 26 de maio de 2009.

[40] Rodrigo V. Miranda, Cláudio de S. Baptista, Rodrigo R. Almeida, Bruno Catão, Eder Pazinatto. IGIS: um framework para sistemas de informações geográficas em n-camadas usando um SGBD objeto-relacional. In SBC GeoInfo*, Caxambu, Brasil, 2002.*

[41] Martin Fowler. Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.*

[42] Anthony C. Bloesch, Terry A. Halpin. Conquer: A conceptual query language. In Bernhard Thalheim, editor, Conceptual Modeling - ER'96, 15th International Conference on Conceptual Modeling, Cottbus, Germany, October 7-10, 1996, Proceedings*, volume 1157 of* Lecture Notes in Computer Science*, pages 121–133. Springer, 1996.*

[43] A. J. Morris, A. I. Abdelmoty, B. A. El-Geresy, C. B. Jones. A filter flow visual querying language and interface for spatial databases. Geoinformatica*, 8(2):107–141, 2004.*

[44] George Reese. Database Programming with JDBC and Java, Second Edition*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2000.*