

UNIVERSIDADE CANDIDO MENDES
CAMPOS DOS GOYTACAZES
MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

RODRIGO SOARES MANHÃES

UMA ARQUITETURA PARA VISUALIZAÇÃO E APLICAÇÃO DE OPERAÇÕES
SOBRE DADOS MULTIDIMENSIONAIS

CAMPOS DOS GOYTACAZES
2007

RODRIGO SOARES MANHÃES

UMA ARQUITETURA PARA VISUALIZAÇÃO E APLICAÇÃO DE OPERAÇÕES
SOBRE DADOS MULTIDIMENSIONAIS

Dissertação apresentada ao Curso de Mestrado em
Pesquisa Operacional e Inteligência Computacional
da Universidade Candido Mendes, como requisito
parcial para a obtenção do grau de Mestre.

Orientadora: Sahudy Montenegro González, D.Sc.
Co-Orientador: Asterio Kiyoshi Tanaka, Ph.D.

Campos dos Goytacazes
2007

RODRIGO SOARES MANHÃES

UMA ARQUITETURA PARA VISUALIZAÇÃO E APLICAÇÃO DE OPERAÇÕES
SOBRE DADOS MULTIDIMENSIONAIS

Dissertação apresentada ao Curso de Mestrado em
Pesquisa Operacional e Inteligência Computacional da
Universidade Candido Mendes, como requisito parcial
para a obtenção do grau de Mestre.

Aprovada em _____

BANCA EXAMINADORA

Prof^a. Sahudy Montenegro González, D.Sc. – Orientadora
Universidade Candido Mendes (UCAM-Campos)

Prof. Asterio Kiyoshi Tanaka, Ph.D. – Co-Orientador
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

Prof. Dalessandro Soares Vianna, D.Sc.
Universidade Candido Mendes (UCAM-Campos)

Prof. Rogério Atem de Carvalho, D.Sc.
Centro Federal de Educação Tecnológica de Campos (CEFET-Campos)

Campos dos Goytacazes
2007

AGRADECIMENTOS

A Ana Carla e João Carlos por todo o amor, em todos os momentos.

A meus pais Edilson e Salvadora pelo exemplo de vida e pelo auxílio durante todo o mestrado e ao meu irmão Rafael.

A minha sogra Vanda e minha cunhada Clara pela acolhida e pelo escritório.

Aos meus amigos, que não deixaram de ser amigos mesmo após muito tempo de completo abandono.

Aos meus orientadores Sahudy e Tanaka, pelos imensos auxílio e paciência durante toda a caminhada.

Ao ex-orientador Rogério, que me apresentou ao projeto.

Ao pessoal do Desenvolvimento/UENF por ouvir lamentações diárias sobre OLAP por anos a fio.

E, *last but not least*, a Deus, por cabeça, coração e tudo mais.

“Os filósofos não têm feito senão interpretar o mundo;
o que importa, porém, é transformá-lo”
Karl Marx, XI Tese sobre Feuerbach

SUMÁRIO

SUMÁRIO	IV
LISTA DE SIGLAS	X
LISTA DE FIGURAS	XII
LISTA DE TABELAS	XV
1 INTRODUÇÃO.....	1
1.1 MOTIVAÇÃO.....	2
1.2 OBJETIVOS	3
1.3 TRABALHOS RELACIONADOS.....	4
1.3.1 OLAP e Dados Geográficos	4
1.3.1.1 Processamento Analítico de Dados Geográficos	5
1.3.1.2 Integração de Aplicações DW/OLAP e SIG	5
1.3.1.3 Abordagem unificada para OLAP e SIG	6
1.3.2 Visualização Multidimensional	7
1.4 ORGANIZAÇÃO DA DISSERTAÇÃO	8
2 FUNDAMENTAÇÃO TEÓRICA	9
2.1 SISTEMAS DE SUPORTE À DECISÃO	9
2.2 DATA WAREHOUSE	12
2.3 SISTEMAS DE INFORMAÇÃO GEOGRÁFICA (SIG)	17
2.3.1 JUMP Unified Mapping Platform	18
2.4 OLAP	19
2.4.1 Hierarquia.....	21
2.4.2 Operações OLAP	21
2.4.2.1 Slice-and-Dice	22
2.4.2.2 Pivoting	23
2.4.2.3 Drill-down	23
2.4.2.4 Roll-up.....	24
2.4.2.5 Drill-across	24
2.4.2.6 Drill-through	25
2.4.3 OLAP Espacial.....	25

2.5 POSTGRESQL	27
2.5.1 Desempenho e Viabilidade	28
2.5.2 PostGIS	29
2.6 REPRESENTAÇÃO VISUAL MULTIDIMENSIONAL.....	29
2.7 CUBE PRESENTATION MODEL.....	32
2.7.1 AttributeGroup	34
2.7.2 DimensionGroup	34
2.7.3 Ancestor Function	35
2.7.4 SelectionCondition	35
2.7.5 EquallySelectionCondition	35
2.7.6 AxisSchema	36
2.7.7 Axis.....	36
2.7.8 Point.....	38
2.7.9 CubeSchema.....	38
2.7.10 Cube	39
2.7.11 2D-Slice.....	39
2.7.12 Tape.....	39
2.7.13 Cross-join.....	40
3 O PROJETO GEOOLAP.....	41
3.1 ESTEREÓTIPOS GEOGRÁFICOS.....	42
3.2 A FERRAMENTA POSTGEOOLAP.....	44
3.2.1 Princípios de Projeto.....	44
3.2.2 Modelo	45
3.2.3 Funcionalidades.....	47
3.2.4 Processamento do Cubo e Geração das Agregações.....	49
3.3 EXTENSÕES AO MODELO DO POSTGEOOLAP	50
3.3.1 Tratamento de Dados Multidimensionais	50
3.3.2 Dimensões e Atributos.....	54
3.3.3 Mapas.....	56
3.3.4 Atributo Principal por Nível Hierárquico	57
3.3.5 Resultado das extensões	58
3.4 MIGRAÇÃO PARA A PLATAFORMA JAVA	60
3.4.1 Java e Desempenho.....	60

	viii
3.4.2 Java e Software Livre	62
3.4.3 Visualização de Mapas.....	63
3.4.4 O Fator Geográfico.....	65
3.4.5 Interface Gráfica.....	65
3.4.6 Organização do Projeto	66
3.4.7 Arquitetura e Dependências	67
4 .PROPOSTA DE VISUALIZAÇÃO MULTIDIMENSIONAL	69
4.1 EXTENSÕES AO CUBE PRESENTATION MODEL	69
4.1.1 Definições.....	69
4.1.2 Interação OLAP flexível.....	71
4.1.3 Limitações de Flexibilidade no CPM	73
4.1.3.1 Examinando a relação AxisSchema/Axis no modelo CPM	76
4.1.4 Extensões à camada de apresentação do CPM	79
4.1.5 Exemplificando a Extensão Proposta.....	85
4.2 ARQUITETURA DO SISTEMA OLAP PROPOSTO.....	88
4.2.1 Comunicação entre o Modelo de Visualização e o Toolkit Gráfico.....	89
4.2.2 Comunicação entre o modelo OLAP e o modelo de visualização.....	94
4.3 SUPORTE A INTERFACES OLAP NO JAVA/SWING	97
4.3.1 Representação de cabeçalhos hierárquicos.....	98
4.3.2 JTableHeader e suporte a cabeçalhos hierárquicos.....	100
4.3.3 Cabeçalhos de linha	101
5 ESTUDO DE CASO.....	104
5.1 DESCRIÇÃO	104
5.2 MODELAGEM DO DATA WAREHOUSE	105
5.2.1 Projeto do Banco de Dados.....	106
5.3 CRIAÇÃO E PROCESSAMENTO DO CUBO	109
5.3.1 Conexão com o SGBD.....	109
5.3.2 Criação do cubo.....	110
5.4 ANÁLISE DOS DADOS	115
6 CONCLUSÃO	120
6.1 CONTRIBUIÇÕES	122

6.2 SUGESTÕES PARA TRABALHOS FUTUROS	123
REFERÊNCIAS BIBLIOGRÁFICAS	125

LISTA DE SIGLAS

AJAX – Asynchronous JavaScript and XML
ANSI – American National Standard Institute
API – Application Programming Interface
AWT – Abstract Windowing Toolkit
BSD – Berkeley Software Distribution
CPM – Cube Presentation Model
DOLAP – Desktop OLAP, Database OLAP
DW – Data Warehouse
E-R – Entidade-Relacionamento
ETL – Extract-Transformation-Load
GDSS – Group Decision Support System
GiST – Generalized Search Tree
GNU – GNU is Not Unix
GPL – General Public License
GUI – Graphical User Interface
HOLAP – Hybrid OLAP
HTML – Hyper Text Markup Language
IEC – International Engineering Consortium
IPTU – Imposto Predial e Territorial Urbano
IR – Information Retrieval
ISO – International Organization for Standardization
ISS – Imposto Sobre Serviços
JCP – Java Community Process
JDK – Java Development Kit
JIT – Just In Time
JTS – JTS Topology Suite
JUMP – JUMP Unified Mapping Platform
JVM – Java Virtual Machine
KDD – Knowledge Discovery in Databases
KS – Knowledge System
LS – Language System
MOLAP – Multidimensional OLAP

MTS – Multidimensional Type Structure
MVC – Model-View-Controller
OCL – Object Constraint Language
OGC – Open Geospatial Consortium
OLAP – On-Line Analytical Processing
OLTP – On-Line Transaction Processing
OpenGIS – Open Geodata Interoperability Specification
PPS – Problem Processing System
PS – Presentation System
ROLAP – Relational OLAP
SGBD – Sistema Gerenciador de Banco de Dados
SGBD-R – Sistema Gerenciador de Banco de Dados Relacional
SIG – Sistema de Informação Geográfica
SOLAP – Spatial OLAP
SQL – Structured Query Language
SSD – Sistema de Suporte à Decisão
SWT – Standard Widget Toolkit
TPC – Transaction Processing Performance Council
UML – Unified Modeling Language
W3C – World Wide Web Consortium
WKT – Well-Known Text
WOLAP – Web OLAP
XML – eXtended Markup Language

LISTA DE FIGURAS

Figura 1. Exemplo de modelo lógico multidimensional em estrela	15
Figura 2. Exemplo de <i>snowflake</i>	16
Figura 3. Cubo de dados para uma aplicação de vendas. Fonte: Datta e Thomas (1999)....	22
Figura 4. Exemplo de hierarquia da dimensão Tempo. Fonte: Galante, 2004.....	24
Figura 5. Paginação como representação visual n-dimensional. Fonte: Thomsen (2003)..	31
Figura 6. Grade multidimensional contendo as dimensões Vendedor, Local e Tempo ...	32
Figura 7. Representação UML da camada de apresentação do CPM.....	33
Figura 8. Visualização OLAP com fato como dimensão implícita	37
Figura 9. Visualização OLAP com fato como dimensão visual explícita	37
Figura 10. Associação com uma hierarquia extensa para a dimensão Rio	43
Figura 11. Estereótipo representando a associação de Rio com LineString	44
Figura 12. Modelo conceitual do PostGeoOlap. Fonte: Colonese (2004).....	46
Figura 13. Diagrama original de projeto do PostGeoOlap. Fonte: Colonese (2004).....	47
Figura 14. Consulta no PostGeoOlap original. Fonte: Colonese (2004)	50
Figura 15. Consulta no PostGeoOlap original. Fonte: Galante (2004).....	51
Figura 16. Extensão ao modelo do PostGeoOlap	53
Figura 17. Relação entre tabelas, atributos e dimensões. Fonte: Colonese (2004).....	54
Figura 18. Modelo proposto para a relação entre dimensões e atributos.....	55
Figura 19. Associação entre mapas e esquemas	56
Figura 20. Associação um-para-muitos entre ItemHierarquia e Atributo	57
Figura 21. ItemHierarquia associado a um Atributo principal e a vários auxiliares .	58
Figura 22. Resultado das extensões ao PostGeoOlap	59
Figura 23. Visualizador do OpenJUMP incorporado ao PostGeoOlap	64
Figura 24. Arquitetura do PostGeoOlap	67
Figura 25. <i>Data warehouse</i> de exemplo.	70
Figura 26. Dimensão Tempo sendo visualizada por ano	71
Figura 27. Dimensão Tempo sendo visualizada por semestre	71
Figura 28. A instância "2003" da dimensão Tempo sendo visualizada por semestre.....	72
Figura 29. Interface OLAP acrescida da dimensão Local	72
Figura 30. Dimensão Local após <i>drill-down</i> em uma das instâncias “Japão”	72
Figura 31. Interface OLAP com interações complexas.....	73

Figura 32. Interface OLAP após remoção da dimensão <i>Vendedor</i>	73
Figura 33. Exemplo utilizado para demonstrar o CPM. Fonte: Maniatis <i>et al.</i> (2003).....	74
Figura 34. Réplica do exemplo padrão do CPM.....	75
Figura 35. Tela OLAP após operação de <i>roll-up</i>	78
Figura 36. Pontos em destaque em uma interface OLAP	80
Figura 37. Visualização de todas as instâncias individuais dos níveis definidos.....	82
Figura 38. Detalhe de interface OLAP (a) antes e (b) depois de uma operação <i>roll-up</i>	83
Figura 39. Modelo conceitual do CPM acrescido de extensões.....	84
Figura 40. Tela OLAP após múltiplas operações de <i>drilling</i>	85
Figura 41. Interface OLAP dada pelos <i>axes schemata</i>	85
Figura 42. <i>Roll-up</i> no vendedor “Pitz”	86
Figura 43. <i>Roll-up</i> em país região.....	86
Figura 44. <i>Drill-down</i> em “Qtr1” e “Qtr4”	87
Figura 45. Mais aplicações da operação drill-down (Equação 16).....	87
Figura 46. Diagrama de componentes do sistema OLAP	88
Figura 47. Classes que transportam o conteúdo de uma interface gráfica OLAP	91
Figura 48. Exemplo de objetos <i>JTable</i> , <i>JTableHeader</i> e <i>JScrollPane</i> em ação.....	98
Figura 49. Aninhamento de células no cabeçalho de tabelas	99
Figura 50. Em vermelho, objetos <i>TableColumn</i> ; em cinza, objetos <i>HeaderGroup</i>	99
Figura 51. Extensões aos componentes Swing	100
Figura 52. Exemplo de tabela com cabeçalhos hierárquicos de coluna	101
Figura 53. Exemplo de transposição.....	101
Figura 54. Exemplo de cabeçalhos hierárquicos para linha e coluna	103
Figura 55. Diagrama de classes conceitual do modelo estrela para o estudo de caso.....	105
Figura 56. Configuração do esquema	110
Figura 57. Efetuando conexão com o banco de dados	110
Figura 58. Criação de um novo cubo.....	111
Figura 59. Seleção da tabela-fato e dos atributos agregáveis.....	111
Figura 60. Seleção de uma dimensão	112
Figura 61. Seleção da hierarquia para a dimensão <i>OrigemDebito</i>	112
Figura 62. Seleção da hierarquia para a dimensão <i>Debito</i>	113
Figura 63. Seleção da hierarquia para a dimensão <i>Natureza</i>	114
Figura 64. Seleção da hierarquia para a dimensão <i>Tempo</i>	114

Figura 65. Grade multidimensional apresentando Natureza x Tempo	115
Figura 66. Eixo vertical após operação de <i>drill-down</i>	116
Figura 67. Aplicação de <i>drill-down</i> apenas na instância “1994”	117
Figura 68. Aplicação de <i>drill-down</i> apenas na instância “1991”	117
Figura 69. Aplicação da operação de rotação	118
Figura 70. Incluindo a dimensão <i>Debito</i> na visualização	118
Figura 71. Dimensões <i>Debito</i> , <i>Tempo</i> e <i>Natureza</i> sendo visualizadas.....	118
Figura 72. Visualização geográfica no PostGeoOlap	119

LISTA DE TABELAS

Tabela 1. Comparação entre representações para mapas temáticos	18
Tabela 2. Estereótipos para os tipos de dados da OpenGIS	43
Tabela 3. Lista de casos de uso do PostGeoOlap.....	48
Tabela 4. Atributos da dimensão Debito	107
Tabela 5. Atributos da dimensão OrigemDebito	107
Tabela 6. Atributos da dimensão Natureza.....	108
Tabela 7. Atributos da dimensão Tempo	108
Tabela 8. Atributos da tabela fato ItemDebito	109

RESUMO

A integração entre sistemas de informações geográficas e sistemas de processamento analítico tem sido tema de muitos trabalhos nos últimos anos, com o objetivo de unificar, sob o ponto de vista do processamento analítico, dados convencionais e geo-referenciados, unindo o componente espacial aos dados analíticos. O presente trabalho visa estender a ferramenta PostGeoOlap, que permite o desenvolvimento de aplicações de suporte à decisão fazendo uso desta abordagem. PostGeoOlap trabalha junto a um sistema de gerenciamento de banco de dados objeto-relacional, o PostGreSQL, acrescido de sua extensão espacial, PostGIS, e executa suas funcionalidades sobre um *data warehouse*. O presente trabalho tem como objetivos portar a ferramenta para uma linguagem aceita no ambiente do Software Livre e propor extensões ao seu núcleo para o suporte à multidimensionalidade. O resultado de maior relevância deste trabalho é uma *engine* de visualização multidimensional independente de *toolkits* gráficos, que estende a proposta Cube Presentation Model. Um estudo de caso é apresentado para validar e testar a *engine* de visualização.

ABSTRACT

The integration of geographic information systems and analytical processing systems has been gained focus in the last years, in order to unify, by an analytical view, conventional e georeferenced data, joining spatial component to analytical data. This work aims to extend the PostGeoOlap tool

A integração entre sistemas de informações geográficas e sistemas de processamento analítico tem sido tema de muitos trabalhos nos últimos anos, com o objetivo de unificar, sob o ponto de vista do processamento analítico, dados convencionais e geo-referenciados, unindo o componente espacial aos dados analíticos. O presente trabalho visa estender a ferramenta PostGeoOlap, que permite o desenvolvimento de aplicações de suporte à decisão fazendo uso desta abordagem. PostGeoOlap trabalha junto a um sistema de gerenciamento de banco de dados objeto-relacional, o PostGreSQL, acrescido de sua extensão espacial, PostGIS, e executa suas funcionalidades sobre um *data warehouse*. O presente trabalho tem como objetivos portar a ferramenta para uma linguagem aceita no ambiente do Software Livre e propor extensões ao seu núcleo para o suporte à multidimensionalidade. O resultado de maior relevância deste trabalho é uma *engine* de visualização multidimensional independente de *toolkits* gráficos, que estende a proposta Cube Presentation Model. Um estudo de caso é apresentado para validar e testar a *engine* de visualização.

1 INTRODUÇÃO

As tecnologias da chamada terceira revolução industrial, a revolução da microeletrônica detonada com a invenção do transistor, têm transformado a face do planeta. Embora diferentes autores discordem frontalmente das suas conseqüências, dividindo-se entre entusiastas (GATES, 1995), críticos moderados (SASSEN, 1998; CASTELLS, 1999), críticos ferrenhos (CHOMSKY, 2002) e aqueles que compreendem a fase global-financeira do capitalismo como aquela que determinará seu colapso (KURZ, 1992; KURZ, 1997), praticamente todos convergem no entendimento de que a generalização do microcomputador e o avanço das telecomunicações impulsionam o mundo a um novo patamar, ainda que divirjam radicalmente na qualidade dos frutos deste processo para a humanidade.

Neste contexto, a informação adquire um imenso valor, seja para, em se tratando de empresas, estar em melhores condições para a concorrência com seus pares, seja para, falando-se do setor público ou organizações sem fins lucrativos, melhor desempenhar suas tarefas e atender as demandas de seu público-alvo. Assim, percebe-se que a informação está, cada vez mais, moldando a maneira como a sociedade trabalha, produz e pensa (CASTELLS, 1999).

Especificamente no que tange ao setor de tecnologia da informação, as organizações têm acumulado grandes quantidades de dados em seus sistemas. Estes dados, porém, para significar algo mais do que meros cadastros e registros operacionais, precisam tomar a forma, mais interessante e adequada, de informação. Para isto, surgiram os Sistemas de Suporte à Decisão (SSD), que integram tecnologias como *data warehousing*, *data mining*, OLAP (*On-Line*

Analytical Processing), SIG (Sistemas de Informações Geográficas), KDD (*Knowledge Discovery in Databases*) e IR (*Information Retrieval*), entre outras.

O objetivo final destas tecnologias é permitir ao usuário, normalmente alguém que atua nos níveis estratégico ou tático de uma instituição, visualizar, sob várias perspectivas, diversos tópicos pertinentes à sua área de atuação. A capacidade de análise de dados agregados e integrados a partir de diversas fontes faz do *Data Warehouse* (DW) e de uma aplicação OLAP que lhe forneça processamento analítico ferramentas indispensáveis aos usuários, permitindo observações e análises do domínio de atuação sob diversas perspectivas. A estas tecnologias somam-se os SIGs, que utilizam mapas para prover análises de dados que possuam atributos referentes ao posicionamento espacial, algo praticamente indispensável para quem possui alguma dimensão espacial em seu domínio de atuação (CÂMARA e DAVIS, 2001).

Numa primeira tentativa, muito se trabalhou sobre pontos de vista de integração de sistemas geográficos e analíticos pré-existent (KOUBA, MATOUŠEK e MIKŠOVSKÝ, 2000; FERREIRA, CAMPOS e TANAKA, 2002; FIDALGO, TIMES e SOUZA, 2001). A presente dissertação, porém, seguindo a perspectiva de um ambiente onde se tratem, de modo unificado, dados geográficos e convencionais num ambiente analítico, constitui uma proposta conceitualmente mais próxima às de Han, Stefanovic e Koperski (1998), Papadias *et al.* (2001), Rivest, Bédard e Marchand (2001) e Bimonte, Tchounikine e Miquel (2005). O presente trabalho inicia-se como uma continuação natural do trabalho de Colonese (2004), adicionando funcionalidades OLAP ao modelo inicialmente proposto.

1.1 MOTIVAÇÃO

Como é fato que o custo de aquisição ou licenciamento de sistemas de suporte à decisão como aplicações OLAP ou SIG é proibitivo a pequenas e médias empresas, a presente iniciativa torna acessíveis a um público muito maior estas tecnologias (COLONESE *et al.*, 2006), normalmente restritas à esfera das grandes organizações. No setor público, cujos custos com tecnologia da informação em boa parcela são creditados a licenciamentos de *software*, também possui grande importância o uso do Software Livre, no sentido da economia de recursos, da independência de fornecedores e, principalmente, da melhoria das condições sociais e da

qualidade de vida das populações, seguramente resultantes da honesta utilização das citadas tecnologias de suporte à decisão.

É importante frisar que, principalmente em países pobres ou ditos “em desenvolvimento” como o Brasil, é ainda mais importante a adoção de *software* sem restrições de uso ou distribuição e de código aberto, como instrumento para a democratização da tecnologia e da informação (SILVEIRA, 2003; BRANCO, 2005).

Neste contexto, o presente trabalho se insere com o objetivo de incrementar à ferramenta capacidades avançadas de análise, provendo a instituições uma ferramenta OLAP com extensões geográficas de baixo custo.

1.2 OBJETIVOS

O objetivo original da ferramenta PostGeoOlap é, segundo Colonese (2004), permitir a construção de “sistemas de suporte à decisão onde, desde a fase de modelagem conceitual, seja possível tratar aspectos analíticos e geográficos”. Assim, PostGeoOlap, conforme concebida originalmente, é uma ferramenta que realiza consultas analíticas sobre um *data warehouse* contendo dimensões com atributos geográficos, não contemplando operações OLAP automatizadas.

A meta da presente dissertação é propor extensões à ferramenta PostGeoOlap que dotem-na de capacidades avançadas de análise, com a implementação de operadores OLAP, sempre preocupando-se com a clareza e intuitividade da interface com o usuário, um dos principais aspectos de projeto de um sistema de suporte à decisão. Para isto, discutem-se as operações OLAP mais comuns e OLAP espacial; descrevem-se as extensões à ferramenta e desenvolve-se um estudo de caso para validação da proposta.

A principal contribuição do presente trabalho, deste modo, é apresentar uma proposta de modelagem de visualização OLAP consistente e flexível e incorporá-la a uma ferramenta em Software Livre para suporte à decisão para processamento analítico de dados convencionais e geográficos, dotando seus usuários de capacidade real de enxergar analiticamente os fatores que influenciam sua área de atuação, utilizando os aspectos próprios ao domínio aliados aos componentes temporal e cartográfico, de modo unificado desde o início, sem a necessidade de módulos de integração.

Além disto, a proposta deste trabalho visa ao projeto de uma estrutura de visualização e operação analíticas baseada numa extensão da proposta *Cube Presentation Model* (CPM) (MANIATIS *et al.*, 2003) para a adição de flexibilidade no suporte a operações de *drilling*. A presente dissertação propõe também uma extensão do modelo da ferramenta PostGeoOlap (COLONESE, 2004) para a incorporação de características analíticas e a incorporação de atributos geográficos a estas características. Propõe, ainda, uma arquitetura orientada a objetos visando à eliminação de dependências entre o modelo de visualização, o *toolkit* gráfico e o modelo lógico OLAP subjacente, de modo que um modelo seja completamente independente dos outros, e mesmo independente de tecnologia de visualização. Ao fim, um estudo de caso demonstrará o funcionamento destas características integradas à ferramenta livre PostGeoOlap.

1.3 TRABALHOS RELACIONADOS

Diversos outros trabalhos foram conduzidos no sentido de buscar a integração entre sistemas analíticos e geográficos e, por sua relevância, são aqui citados com o objetivo de estabelecer um paralelo entre o que já foi produzido e o que está sendo realizado nesta dissertação. Além disto, são citados trabalhos que abordam a visualização de dados multidimensionais.

1.3.1 OLAP e Dados Geográficos

No campo formado pela interseção entre OLAP e SIG, há trabalhos relevantes em três áreas: processamento analítico sobre dados geográficos, integração de sistemas OLAP e SIG pré-existentes e propostas para uma abordagem unificada de OLAP e dados geográficos.

1.3.1.1 Processamento Analítico de Dados Geográficos

Entre os projetos que abordam o processamento analítico sobre dados puramente geográficos, podem-se citar o GeoMiner (STEFANOVIC, 1997) e o MapCube (SHEKHAR *et al.*, 2000).

GeoMiner é um projeto de *data mining* espacial, onde extensões são propostas ao modelo estrela de forma a prover ponteiros para objetos espaciais aos itens da tabela fato e das dimensões. Esta abordagem não integra, tecnicamente, um DW e um SIG, mas apenas permite operações OLAP sobre um cubo com dados geo-referenciados.

MapCube é um projeto parcialmente patrocinado pelo exército estadunidense, cujo principal objetivo é ser uma extensão ao conceito de cubo de dados para o domínio espacial. É um operador análogo ao operador Cube usado em consultas sobre um SGBD-R (Sistema Gerenciador de Banco de Dados Relacional), com a diferença de que enquanto o operador Cube retorna uma tabela (relação) como resultado de suas operações, o MapCube retorna mapas e tabelas, executando operações como soma e interseção sobre dados geográficos.

1.3.1.2 Integração de Aplicações DW/OLAP e SIG

Entre projetos que integram aplicações SIG e DW distintas através de módulos de integração, podem-se citar GOAL (KOUBA, MATOUŠEK e MIKŠOVSKÝ, 2000), SIGOLAP (FERREIRA, CAMPOS e TANAKA, 2002) e GOLAPA (FIDALGO, TIMES e SOUZA, 2001).

O projeto GOAL (Geographical Information On-Line Analysis) é parte do programa de pesquisa INCO-COPERNICUS, da União Européia, e seu objetivo é integrar GIS com DW/OLAP de forma a prover aos decisores informações sobre as perspectivas dimensionais e espaciais. GOAL utiliza metadados como apoio ao processo de associação entre dados geográficos e dimensionais.

O SIGOLAP busca a integração entre OLAP e SIG utilizando um modelo de integração em três camadas: Camada de Fontes de Dados, responsável por prover os dados a serem analisados; Camada de Middleware, responsável pela mediação entre a Camada de Dados e a de

Aplicação e cujo principal componente é o módulo de integração, que mapeia os conceitos usados pelas ferramentas OLAP e SIG; e Camada de Aplicação, onde estão os aplicativos que devem prover funcionalidades analíticas e geográficas. Posteriormente, Ferreira (2002) propõe um modelo de integração baseado em metadados, que atua como um modelo de referência para a tradução de conceitos utilizados por ferramentas OLAP e SIG.

O projeto GOLAPA (Geographical On-Line Analytical Processing Architecture) propõe a integração entre DW/OLAP e SIG através do uso de tecnologias abertas e extensíveis, como Java, XML (Extended Markup Language) e *web services*. Esta arquitetura contempla as fases de ETL (Extract-Transformation-Load) para um *data warehouse* geográfico.

1.3.1.3 Abordagem unificada para OLAP e SIG

Entre os projetos que propõem OLAP espacial, podem-se citar a Materialização Seletiva (HAN, STEFANOVIC e KOPERSKI, 1998), o trabalho de Papadias *et al.* (2001), a proposta SOLAP (RIVEST, BÉDARD e MARCHAND, 2001), o trabalho de Bimonte, Tchounikine e Miquel (2005) e a ferramenta PostGeoOlap.

A Materialização Seletiva consiste numa proposta de *data warehouse* espacial, ou seja, um cubo de dados onde as dimensões e as medições poderiam ser atributos convencionais ou espaciais. O trabalho foca justamente nas medições (atributos da tabela fato) espaciais, propondo uma estratégia para a definição de quais objetos geográficos deveriam ser materializados (pré-calculados).

Papadias *et al.* (2001) focam seu esforço nas dimensões espaciais, onde muitas vezes as hierarquias e agrupamentos não podem ser bem definidos antecipadamente e que, por tal motivo, não podem se beneficiar das técnicas já consagradas de materialização das visões, propondo uma hierarquia de agrupamento baseada no índice espacial em sua menor granularidade.

Rivest, Bédard e Marchand (2001) propõem uma categoria de aplicações OLAP batizada de SOLAP (Spatial OLAP), lançando bases conceituais para o suporte espacial à decisão baseado em OLAP e demonstrando características desejáveis em ferramentas OLAP espaciais.

Bimonte, Tchounikine e Miquel (2005) apresentam um modelo espacial multidimensional capaz de prover suporte ao uso de objetos complexos e atributos interdependentes como medidas ou funções de agregação, além do uso de funções de agregação *ad hoc* e suporte a relações muitos-para-muitos entre fato e dimensão.

A ferramenta PostGeoOlap – que constitui parte de um projeto mais amplo, o GeoOlap (MANHÃES *et al.*, 2007) – é uma ferramenta para a construção de soluções OLAP espaciais integradas desde sua modelagem.

1.3.2 Visualização Multidimensional

Estudos acadêmicos envolvendo modelagem de dados para *data warehousing* e OLAP têm sido uma constante nos últimos anos, com a existência de inúmeros modelos sendo propostos e, apesar de nenhum deles ter sido adotado como padrão pela indústria, é um assunto bastante explorado (CODD *et al.*, 1993; INMON, 1996; KIMBALL e ROSS, 2002; THOMSEN, 2002; TRUJILLO *et al.*, 2001; VASSILIADIS e SKIADOPOULOS, 2000; VASSILIADIS e SELLIS, 1999; LECHTENBÖRGER e VOSSEN, 2003; ABELLÓ, SAMOS e SALTOR, 2005).

O mesmo não se pode dizer, contudo, sobre um tópico de fundamental importância: a visualização dos dados. Apesar de, tradicionalmente, a visualização estar longe de ser tratada como um tópico fundamental, quando se trata de suporte à decisão a apresentação dos dados tem sua relevância alçada ao *status* de primordial. De fato, Holsapple e Whinston (1996) definem uma estrutura geral para sistemas de suporte à decisão, descrevendo-a em termos de quatro subsistemas: Language System (LS), Presentation System (PS), Knowledge System (KS) e Problem Processing System (PPS), reservando, assim, aos assuntos concernentes à visualização, um dos quatro pilares de seu *framework* conceitual para sistemas de suporte à decisão.

Como iniciativas para tratar especificamente da visualização multidimensional, podem ser citados CoDecide (GEBHARDT, JARKE e JACOBS, 1997) e Cube Presentation Model (CPM) (MANIATIS *et al.*, 2003).

CoDecide é um *toolkit* para interfaces com o usuário que estende o conceito habitual de planilhas para o suporte a dados multidimensionais. Uma idéia fundamental no CoDecide é o

conceito de fitas (*tapes*), definidos como estruturas infinitas que contêm um número variado de faixas (*tracks*) e permitem que se realizem operações entre si, provendo assim o suporte às operações OLAP.

Cube Presentation Model é um modelo de apresentação para visualização OLAP baseado na representação geométrica de um cubo e sua percepção humana no espaço. A principal idéia do CPM é a separação entre as representações lógica e visual dos dados. Para isto, propõe uma camada lógica – estendendo trabalho de Vassiliadis e Skiadopoulos (2000) para possibilitar a representação de cubos com maior complexidade – e uma camada de apresentação. As duas camadas são independentes, de modo que podem ser utilizadas separadamente em diferentes projetos. O presente trabalho estende a camada de apresentação do CPM para obter possibilidades mais flexíveis de análise por parte do usuário.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação traz, em seu Capítulo 2, a fundamentação teórica necessária para a elaboração e compreensão do presente trabalho, com tópicos tratando de sistemas de suporte à decisão, *data warehouses*, OLAP com dados convencionais e geográficos, sistemas de informação geográfica, entre outros. No Capítulo 3 aborda-se o projeto GeoOlap, no qual a ferramenta aqui descrita se insere, sendo apresentados os estereótipos geográficos para a modelagem de *data warehouses* e a ferramenta PostGeoOlap em si, além de discorrer-se sobre a migração da ferramenta para a plataforma Java. O Capítulo 4 descreve a proposta para visualização OLAP e discute a apresentação de dados geográficos em um contexto multidimensional. O Capítulo 5, finalmente, propõe um estudo de caso para validação da implementação da proposta. O Capítulo 6 conclui a dissertação e aponta tópicos em aberto para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, discorrer-se-á brevemente sobre alguns tópicos fundamentais para o entendimento do restante da presente dissertação. Inicialmente, introduzir-se-ão os Sistemas de Suporte à Decisão, área na qual o presente trabalho se insere. Posteriormente, *data warehouses* e sistemas de informação geográfica (SIG), tecnologias nas quais se baseia o projeto, são discutidos. Segue-se uma explanação sobre o foco principal do trabalho, OLAP e suas operações. Faz-se, necessário, ainda à guisa de fundamentação, apresentar o PostgreSQL e o PostGIS, respectivamente SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional e extensão espacial utilizados no trabalho; o JUMP Unified Mapping Platform, *framework* que fornece as classes de visualização espacial utilizadas no projeto; algumas considerações sobre representação visual multidimensional; e o CPM, proposta de visualização multidimensional que é estendida neste trabalho.

2.1 SISTEMAS DE SUPORTE À DECISÃO

Como consequência lógica da progressão quase geométrica do aumento da importância da informação no mundo pós-terceira revolução industrial, a quantidade de dados manipulados por uma dada instituição acaba por assumir proporções assustadoras, na medida em que praticamente todos os processos passam por alguma forma de processamento e armazenamento

computacionais. Isto abre todo um novo mundo de possibilidades e passou-se, então, a perceber que as tecnologias da informação poderiam ter uma utilização que fosse além de meros registros de entradas e saídas, cadastros e relatórios básicos, ou seja, utilizar toda a base de dados sobre clientes, vendas etc. e extrair destas informações úteis e até mesmo conhecimento, permitindo tomadas de decisões mais acuradas e capazes de melhorar a qualidade dos serviços prestados. Para este fim, surgiram os sistemas de suporte à decisão, que são uma classe de sistemas de informação que têm por objetivo auxiliar, de modo interativo, o processo de tomada de decisão, utilizando dados, conhecimento e modelos computacionais para identificar e solucionar problemas (POWER, 2003).

Assim, Elmasri e Navathe (2002) definiram sistemas de suporte à decisão como sistemas que levam dados aos usuários de altos níveis administrativos de uma instituição para o apoio a decisões complexas. Come (2001) definiu apoio à decisão como “o processo de agrupar, estruturar, manipular, armazenar, acessar, apresentar e distribuir informações de negócios de uma maneira oportuna, ou seja, a informação certa no momento certo e na quantidade certa”. Numa definição simples e direta, são sistemas que ajudam os administradores a “definir tendências, apontar problemas e tomar (...) decisões inteligentes” (BONTEMPO e SARACCO, 1996¹ *apud* DATE, 2000). Esses sistemas possuem conhecimentos específicos sobre o assunto e, mediante regras de avaliação introduzidas pelo usuário, retornam soluções otimizadas para os problemas apresentados, auxiliando o processo decisório.

No que diz respeito à taxonomia dos sistemas de suporte à decisão, não há consenso entre os autores mais destacados. Para citar algumas das principais categorizações, Hättenschwiller (1999) propõe uma classificação, pertinente ao usuário, categorizando os SSD como passivos, ativos ou cooperativos. Um SSD passivo auxilia no processo decisório, mas não apresenta ao usuário decisões ou soluções explícitas. Um SSD ativo aponta decisões ou caminhos a ser seguidos pelo usuário. Finalmente, um SSD cooperativo fornece meios para que o administrador possa refinar as sugestões de decisões apresentadas pelo sistema e reenviá-las para nova avaliação e validação. A ferramenta resultante do presente trabalho enquadra-se na categoria de SSD passivo.

Em uma categorização mais genérica, Bhargava e Power (2001) classificam os sistemas de suporte à decisão em cinco diferentes abordagens:

¹ BONTEMPO, Charles J.; SARACCO, Cynthia M.; Database Management: Principles and Products. Upper Saddle River, N.J, EUA: Prentice-Hall, 1996.

- a) SSD orientados a comunicação centram esforços nas tecnologias de telecomunicações para conectar múltiplos decisores que podem estar separados espacial, temporalmente ou ambos ou para fornecer aos decisores acesso remoto a importantes informações ou ferramentas;
- b) SSD orientados a conhecimento podem sugerir ou recomendar ações aos usuários, baseando-se em conhecimento armazenado na forma de fatos, regras ou estruturas similares;
- c) SSD orientados a documentos integram várias tecnologias de armazenamento e processamento para prover análise e recuperação de informações não-estruturadas;
- d) SSD orientados a modelos utilizam representações formais de modelos de decisão para prover suporte analítico utilizando tecnologias como análise de decisão, otimização, modelagem estocástica, simulação, estatística e modelagem lógica;
- e) SSD orientados a dados auxiliam os usuários a organizar, recuperar e sintetizar grandes volumes de dados importantes utilizando pesquisas a bancos de dados, técnicas OLAP e ferramentas de mineração de dados.

Desta última classificação, as duas últimas categorias são as mais utilizadas e difundidas (BHARGAVA e POWER, 2001). Nesta classificação, o PostGeoOlap seria um SSD orientado a dados.

Além disto, os SSD não necessariamente têm de ser orientados a um único gerente ou diretor. Os GDSS (*Group Decision Support System*) são sistemas de suporte à decisão que “combinam tecnologias de comunicação, computação e suporte à decisão para facilitar a formulação e a solução de problemas não estruturados para um grupo de pessoas” (DeSANCTIS e GALLUPE, 1987² *apud* RAO e TUROFF, 2000, tradução do autor). Assim, conclui-se que os Sistemas de Suporte à Decisão podem ser utilizados inclusive em modelos administrativos que escapem ao paradigma taylorista (RAGO e MOREIRA, 1984).

² DeSANCTIS, G.; GALLUPE, R. B. “A Foundation for the Study of Group Decision Support Systems”. In: Management Science. Volume 33, Issue 5. 1987

2.2 DATA WAREHOUSE

A enorme quantidade de dados colhidos pelos sistemas transacionais, que pareceria um ponto deveras positivo para o uso em sistemas de suporte à decisão, acaba por se configurar em um problema para qualquer forma de abordagem analítica, pois, sob esta magnífica quantidade de bits, preciosas informações armazenadas nestes sistemas tornam-se – devido exatamente à sua imensidão – ininteligíveis, perdidas sob toneladas de dados irrelevantes (MAXWELL e GUTOWITZ, 1997).

Além disto, devido à grande magnitude dos dados, consultas mais detalhadas, imprescindíveis a qualquer processo de tomada de decisão auxiliada pelo computador, tendem a ser terrivelmente morosas em exibir seus resultados. O modelo relacional (CODD, 1970), com suas formas normais, é um modelo bastante elegante e fundamental para a obtenção da coerência e integridade dos dados em um sistema transacional. Se por um lado, a modelagem de dados segundo as formas normais provê uma melhor organização dos dados, por outro os pulveriza em uma grande quantidade de tabelas. Para as operações normais do dia-a-dia do ambiente transacional, este é um ótimo modelo e funciona muito bem para a quase totalidade dos casos. Porém, se o foco é desviado para necessidades analíticas, que envolvem consultas *ad hoc* complexas e em diversos níveis, o desempenho se torna baixíssimo, devido à necessidade de que muitas tabelas – que, como foi dito, são imensas – sofram junções, que resultam em nada menos que produtos cartesianos.

Fez-se necessário, assim, o desenvolvimento de tecnologias que permitissem encontrar, em tempo hábil, informações úteis e importantes em meio à miríade de dados gerados pelos sistemas transacionais. Assim, surgiu o *data warehouse*, que, segundo Kimball (1996), é “o lugar onde as pessoas podem acessar seus dados” (tradução do autor). Uma expressão intimamente relacionada, *data warehousing*, é definida por Elmasri e Navathe (2002) como “um conjunto de tecnologias de suporte para a tomada de decisão, direcionadas para capacitar o profissional envolvido na área de conhecimento – o executivo, gerente ou analista – a tomar decisões melhores e mais rápidas”.

Numa definição clássica e um pouco mais formal, Immon (1997) define o *data warehouse* como “uma coleção de dados orientada por assuntos, integrada, não-volátil, variante no tempo, e que tem por objetivo dar suporte aos processos de tomada de decisão”. A esta definição cabe um exame mais apurado.

Diz-se que é orientado por assunto porque contém informações sobre temas específicos importantes para a instituição e sua característica dimensional permite o acesso aos dados através e sob o ponto de vista destes assuntos.

O *data warehouse* é “variante no tempo”, na medida em que os dados nele contidos possuem um componente temporal que permite que se apreenda o modo como se modificam no decorrer do tempo.

Considera-se o *data warehouse* não-volátil porque recebe apenas uma carga inicial de dados, ficando disponível somente para consultas. Ao contrário do que ocorre com o banco de dados em um sistema transacional, onde os dados são atualizados a todo momento, num *data warehouse* a atualização é incremental, normalmente periódica, de acordo com a política definida pelos projetistas.

Por integrado entende-se que os dados num *data warehouse* podem ter origem em diversas fontes, que podem se distinguir tanto geográfica quanto tecnologicamente, sendo reunidos num todo coeso e homogêneo, propiciando, assim, uma visão plena da instituição. Este processo de integração não corresponde somente à cópia dos dados em uma base de dados única, mas compreende um pré-tratamento destes dados, de modo a dotá-los de consistência. Como exemplo, campos lógicos, que podem ter representações diversas, devem ter sua semiologia unificada no pré-tratamento.

Este pré-tratamento e esta integração são realizadas através de um conjunto de processos conhecido como ETL (KIMBALL e CASERTA, 2004). Deste, pode-se destacar (ELMASRI e NAVATHE, 2002):

- a) A extração consiste na retirada dos dados dos sistemas transacionais de origem, que podem estar nos mais diversos formatos e plataformas tecnológicas, inclusive em modo não-estruturado;
- b) Na fase de transformação, as eventuais inconsistências entre as diferentes representações dos vários tipos de dados devem ser convertidos a formas canônicas comuns, obedecendo às regras estabelecidas para tanto, de modo que se eliminem as discrepâncias e se promova a homogeneidade;
- c) Denomina-se carga o processo através do qual os dados obtidos na fase de transformação são transportados ao *data warehouse* para que possam ser utilizados.

Os *data warehouses* abrem mão da modelagem em terceira forma normal que norteia o modelo relacional, otimizada para processamento transacional, e adotam o modelo

multidimensional (KIMBALL e ROSS, 2002), mais simples de ser entendido pela comunidade usuária e com desempenho mais satisfatório se submetido a processamento analítico, pois não necessita valer-se das cadeias de junções imprescindíveis a consultas complexas em modelos normalizados.

A modelagem multidimensional prescreve, basicamente, dois tipos de tabelas: fatos e dimensões. As tabelas fato são as principais tabelas do modelo dimensional e armazenam as medições numéricas resultantes de um processo de negócio. Conseqüentemente, um fato é a representação de uma medição de negócio, sendo a interseção de todas as dimensões presentes em um processo de negócio, possuindo, normalmente, um grande número de linhas. Todas as medições de negócio em uma tabela de fatos devem ter a mesma granularidade. Normalmente, os tipos de dados mais utilizados em tabelas de fatos são numéricos e, mais especificamente, aditivos, o que condiz com a vocação analítica dos *data warehouses*. As tabelas fatos possuem duas ou mais chaves estrangeiras, que as relacionam às tabelas de dimensão, atendendo, assim, à imprescindível integridade referencial. Normalmente, a chave primária de uma tabela de fatos é composta, sendo constituída do conjunto das chaves estrangeiras.

As tabelas de dimensão acompanham a(s) tabela(s) fato, constituindo pontos de entrada de consultas ao *data warehouse*. Elas armazenam informações sobre as entidades de negócio, normalmente tendo um grande número de atributos e uma quantidade de linhas bem menor que as tabelas de fatos. Uma tabela de dimensão é identificada por uma chave primária, que constitui a base da integridade referencial com quaisquer tabelas de fatos associadas. As dimensões, assim, definem a granularidade da tabela de fatos e determina o escopo das medições.

Para reunir os dois tipos de tabelas de um *data warehouse* num modelo multidimensional, existem dois esquemas de modelagem: estrela e *snowflake*. No modelo estrela, preconiza-se a existência de uma tabela fato centralizada, com todas as dimensões diretamente relacionadas a ela, conforme se pode observar na Figura 1.

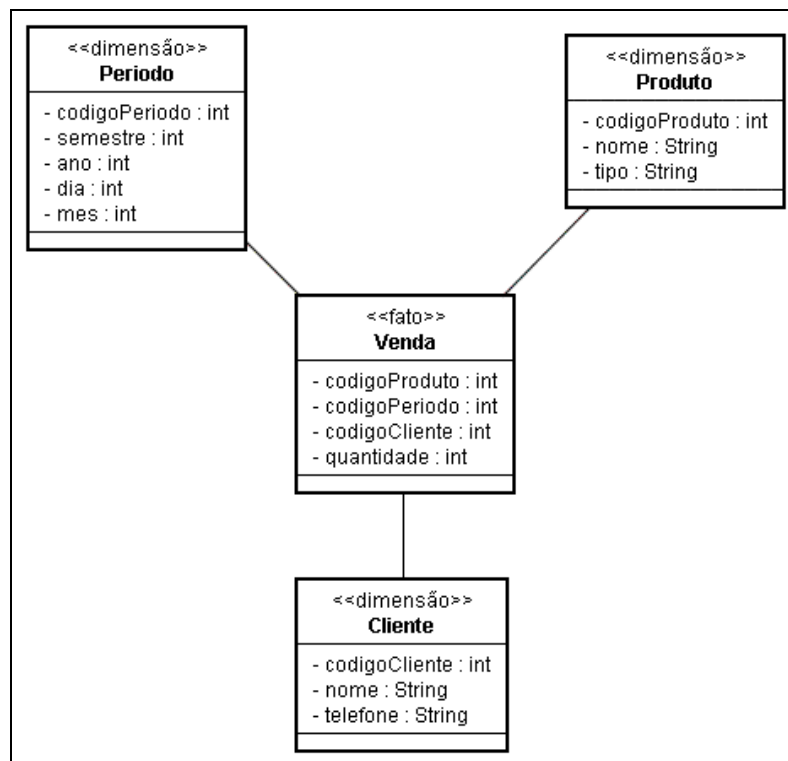


Figura 1. Exemplo de modelo lógico multidimensional em estrela

No modelo *snowflake*, uma variação do modelo estrela, as tabelas dimensão são decompostas, formando hierarquias, o que resulta numa cadeia de tabelas, sendo o motivo do nome do modelo. Normalmente, o modelo *snowflake* é mais complexo para o entendimento do usuário final e resulta em desempenhos mais baixos para consulta, devido à necessidade de junções adicionais, sendo, por isso, desaconselhado na grande maioria dos casos, sendo indicado apenas em circunstâncias bastante específicas (KIMBALL, 2001). Um exemplo de modelagem *snowflake* é mostrado na Figura 2 .

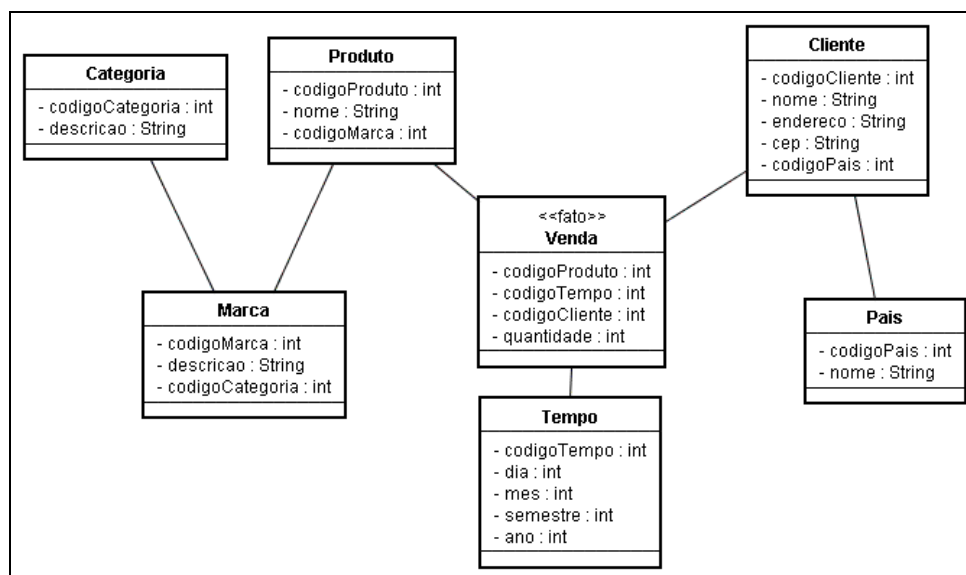


Figura 2. Exemplo de *snowflake*

No que diz respeito à melhor abordagem para a construção de um data warehouse, os mais proeminentes autores neste campo, William “Bill” Inmon e Ralph Kimball, divergem no sentido de que enquanto o primeiro defende que os data marts – subconjuntos lógicos e físicos de um data warehouse – sejam departamentais, o segundo defende que estes estejam orientados a processos de negócio, que podem perfeitamente ser interdepartamentais (PARENTE, 2003).

À guisa de esclarecimento, vale comentar que, a despeito do que é comum na literatura especializada, não se denominará aqui “empresa” (KIMBALL e ROSS, 2002; BARBIERI, 2001) ou “corporação” (MAXWELL e GUTOWITZ, 1997) a pessoa jurídica usuária do *data warehouse*, visto que estes termos resumem sua significação à noção de companhia, empresa capitalista, com fins lucrativos. Usar-se-ão, no decorrer do texto, termos mais genéricos, como “instituição” ou “organização”, pois se sabe que o uso de tecnologias de *data warehousing* e processamento analítico como ferramenta auxiliar no processo decisório tem lugar tanto no âmbito empresarial quanto no terceiro setor, em entidades filantrópicas ou no setor público (GALANTE, 2004; FREITAS, 2007), que é o ramo do estudo de caso proposto ao final deste trabalho.

2.3 SISTEMAS DE INFORMAÇÃO GEOGRÁFICA (SIG)

De acordo com Câmara e Davis (2001), um Sistema de Informação Geográfica é o termo com o qual se denominam as ferramentas computacionais para geoprocessamento, que “permitem realizar análises complexas, ao integrar dados de diversas fontes e ao criar bancos de dados geo-referenciados”, tornando possível, ainda, “automatizar a produção de documentos cartográficos”. Para Rocha (2002), os SIG são sistemas com “capacidade para aquisição, armazenamento, tratamento, integração, processamento, recuperação, transformação, manipulação, modelagem, atualização, análise e exibição de informações digitais georreferenciadas, topologicamente estruturadas, associadas ou não a um banco de dados alfanumérico”.

As aplicações de sistemas de informação geográfica no campo de Suporte à Decisão são inúmeras, como seleção de áreas para um novo empreendimento comercial, prospecção mineral, análise espacial de pontos de venda, estudos de preservação ambiental e muitas outras.

Os dados manipulados pelos SIG, chamados dados geográficos, têm como exemplos típicos mapas e imagens de satélite. Estes dados não informam apenas a localização de acidentes geográficos como rios e montanhas ou construtos antrópicos como ruas e estradas, mas também podem conter dados mais detalhados sobre as localidades, como altitude, tipo de solo ou índice pluviométrico anual (SILBERSCHATZ, KORTH e SUDARSHAN, 1999). Os dados geográficos podem ser representados por dois modos principais:

- a) A representação *matricial* ou *raster* representa o espaço como uma matriz $P(m \times n)$, com m colunas e n linhas, onde cada ponto representa o valor de um atributo fornecido por uma localização geográfica do mundo real.
- b) Na representação *vetorial*, referenciam-se objetos geométricos por intermédio de pontos, linhas e polígonos. Assim, um lago pode ser representado como um polígono ou um rio como uma série de linhas.

Câmara e Monteiro (2001) apresentam um esclarecedor comparativo entre representações matriciais e vetoriais, conforme mostrado na Tabela 2, com os destaques explicitando a forma de representação mais vantajosa para cada quesito.

Aspecto	Representação Vetorial	Representação Matricial
Relações espaciais entre objetos	Relacionamentos topológicos entre objetos disponíveis	Relacionamentos espaciais devem ser inferidos
Ligação com banco de dados	Facilita associação de atributos a elementos gráficos	Associa atributos apenas a classes do mapa.
Análise, simulação e modelagem	Representação indireta de fenômenos contínuos Álgebra de mapas é limitada	Representa melhor fenômenos de variação contínua no espaço. Simulação e modelagem mais fáceis.
Escalas de trabalho	Adequado tanto a grandes quanto a pequenas escalas	Mais adequado para pequenas escalas (1:25.000 e menores)
Algoritmos	Problemas com erros geométricos	Processamento mais rápido e eficiente
Armazenamento	Por coordenadas (mais eficiente)	Por matrizes

Tabela 1. Comparação entre representações para mapas temáticos.

Fonte: Câmara e Monteiro (2001)

2.3.1 JUMP Unified Mapping Platform

O JUMP³ (acrônimo recursivo para JUMP Unified Mapping Platform) é uma aplicação GUI (*Graphical User Interface*) para visualização e processamento de dados espaciais, incorporando funcionalidades existentes em muitos outros SIG direcionados a análise e manipulação de dados geoespaciais. O JUMP também oferece um *framework* altamente extensível para o desenvolvimento e a execução de aplicações customizadas de processamento de dados espaciais.

Esta última característica interessa diretamente ao desenvolvimento do projeto PostGeoOlap, pois neste se utilizam classes de visualização e processamento de dados geográficos fornecidas pelo *framework* JUMP. O JUMP dispõe, assim, de uma API (*Application Programming Interface*) que possibilita pleno acesso, via programação, a todas as

³ <http://www.jump-project.org/project.php?PID=JUMP&SID=OVER>, acesso em 12/03/2007.

funções, visualização de mapas e operações espaciais. As consultas espaciais realizadas pelo PostGeoOlap, deste modo, utilizam-se de classes de visualização e processamento do JUMP para apresentar os resultados, cartograficamente, ao usuário.

O JUMP é inteiramente escrito na linguagem Java, sendo distribuído através da licença GPL (General Public License), o que o torna também concorrente com os conceitos do Software Livre. Em termos de compatibilidade, JUMP é aderente às especificações do OpenGIS⁴ (Open Geodata Interoperability Specification), através do uso de classes de um co-projeto, o JTS Topology Suite (JTS)⁵, que fornece funcionalidades de predicados espaciais, operações topológicas, e outras características, implementando a especificação OpenGIS de Feições Simples para SQL (OPENGIS, 1999).

O JUMP implementa, também, um vasto conjunto de funcionalidades e as oferece inteiramente ao acesso programático, tais como rotulação por atributo com resolução de conflitos, colorização por atributo, estatísticas por camada, estatísticas por feição, operações geométricas como interseção, união, diferença e diferença simétrica, sobreposição de mapas, validação geométrica e outras.

Posteriormente, a empresa responsável pelo desenvolvimento do JUMP descontinuou o projeto, que, por ser Software Livre, pôde ter seu desenvolvimento levado adiante pela comunidade, sob o nome de OpenJUMP⁶.

2.4 OLAP

As tecnologias de *data warehousing* tratam principalmente de armazenamento. Porém, para ser útil a um sistema de suporte à decisão, um *data warehouse* necessita de mecanismos avançados de análise de dados. Para preencher esta lacuna, foi desenvolvido um conjunto difuso de tecnologias conhecido como OLAP, termo cunhado por Codd *et al.* (1993), mas cuja significação já era conhecida há bastante tempo.

Segundo Date (2000), OLAP pode ser definido como “o processo iterativo de criar, administrar e analisar relatórios sobre os dados”. Talvez por ser este um termo mais ligado ao

⁴ O OpenGIS é um consórcio que define padrões abertos na área de sistemas de informações geográficas. As especificações podem ser encontradas em <http://www.opengeospatial.org/specs/?page=specs>, acesso em 12/03/2007.

⁵ <http://www.jump-project.org/project.php?PID=JTS&SID=OVER>, acesso em 12/03/2007.

⁶ <http://openjump.org/wiki/show/HomePage>, acesso em 14/02/2007.

marketing do que a conceituações rigorosas e consistentes (THOMSEN, 2002), Elmasri e Navathe (2002) definem OLAP simplesmente como “uma expressão utilizada para descrever a análise de dados complexos do data warehouse”. Uma proposta um pouco mais formal define OLAP como “uma categoria de software projetada para rápidas exploração e análise dos dados baseado numa abordagem multidimensional com vários níveis de agregação” (CARON⁷ apud RIVEST, BÉDARD e MARCHAND, 2001, tradução do autor).

Quando se fala de OLAP, há que se fazer uma clara distinção entre processamento transacional (OLTP, On-Line Transaction Processing) e processamento analítico (OLAP). O mundo OLTP trata de atividades operacionais do dia-a-dia dos sistemas de informação convencionais: compras, vendas, produção, distribuição *etc.* Quando se fala de OLAP, o público-alvo é aquele que trata de planejamento de recursos, projeção de cenários e iniciativas de *marketing*, ou seja, aquele público que se utiliza de informações orientadas a decisão e baseadas em análise.

As tecnologias OLAP possuem muitas classificações, no que diz respeito a diversos quesitos. Quanto ao armazenamento do *data warehouse*, costumam-se dividir as ferramentas OLAP nas seguintes categorias:

- a) No modelo MOLAP (Multidimensional OLAP), os dados são armazenados em um banco de dados multidimensional, projetado especialmente para o processamento analítico, tendo, por isto, um desempenho bastante satisfatório, além de possuir um rico conjunto de funções de análise de dados e suportar um grande número de usuários sem degradação de desempenho. Como desvantagem, utiliza estruturas proprietárias e não conta com a maturidade dos SGBDs relacionais;
- b) No modelo ROLAP (Relational OLAP), tecnologias tradicionais de SGBDs relacionais são utilizadas para o armazenamento dos dados do *data warehouse*, bem como das agregações, que são mantidas em tabelas. Possui como vantagens a possibilidade de fazer qualquer tipo de consulta e não só as definidas no cubo e o fato de buscar os dados para as agregações no próprio modelo e normalmente no mesmo SGBD em que trabalha a aplicação OLAP. Como desvantagem, tem um desempenho inferior na resposta às consultas.

⁷ CARON, P. Y. Étude du Potential de OLAP pour Supporter l'Analyse Spatio-Temporelle. Dissertação de Mestrado. Département des Sciences Géomatiques, Faculté de Foresterie et Géomatique, Université Laval, 1998.

- c) O modelo HOLAP (Hybrid OLAP) utiliza os dois modelos de armazenamento citados anteriormente, normalmente mantendo os dados em SGBDs relacionais e as agregações em estruturas multidimensionais.

Há outras variantes de OLAP, como DOLAP (Desktop OLAP, Database OLAP), WOLAP (Web OLAP), além de Mobile OLAP e Remote OLAP. Segundo Thomsen (2002), estas classificações possuem um intuito muito mais de *marketing* e vendas do que científico, não havendo qualquer rigor nas classificações.

Não há uma alternativa melhor ou pior entre as estruturas de armazenamento, mas sim aquela que mais se adapta às características da instituição, seus objetivos e restrições (*idem*).

Uma definição importante no que diz respeito a OLAP é o conceito de cubo. Cubo é a metáfora utilizada para representar a multidimensionalidade no tratamento dos dados. Um cubo permite que os dados sejam modelados e visualizados em múltiplas dimensões, sendo uma noção válida tanto no nível lógico quanto em modelos de visualização OLAP.

2.4.1 Hierarquia

Quando um usuário analisa os fatos de um *data warehouse* através de uma dimensão, normalmente os dados são vistos em uma forma sumarizada, podendo ser acessados através de diferentes níveis de detalhamento. Por exemplo, um usuário acessa inicialmente as vendas totais para um ano inteiro. Depois detalha a análise de modo a acessar as vendas sumarizadas por trimestres. Posteriormente, resolve analisar com ainda mais detalhes e desce ao nível dos meses. O que este exemplo mostra é que a *hierarquia* da dimensão tempo é composta de ano, trimestre e mês. As hierarquias de uma dimensão são os caminhos para detalhar ou agregar dados em uma análise.

2.4.2 Operações OLAP

As ferramentas OLAP, via de regra, implementam uma série de operações de suporte à decisão que permitem ao usuário final explorar e analisar dados usando diferentes métodos de visualização. Normalmente, estas operações, que incluem *slicing-and-dicing*, *pivoting*, *drill-*

down, *roll-up*, *drill-across* e *drill-through*, operam sobre um cubo e produzem um novo cubo como saída.

2.4.2.1 Slice-and-Dice

Slice e *dice* são aproximadamente análogos aos operadores da álgebra relacional de seleção e projeção. *Slice* se refere à seleção de dimensões usadas para a visualização do cubo. Na Figura 3, a visão dimensional oferecida é a de Produto (*Product*) por Localização (*Location*), com a dimensão Tempo (*Time*) fixa e sua grade visualmente oculta, com a visualização corrente em destaque. Isto pode ser facilmente modificado através do uso da operação *slice*.

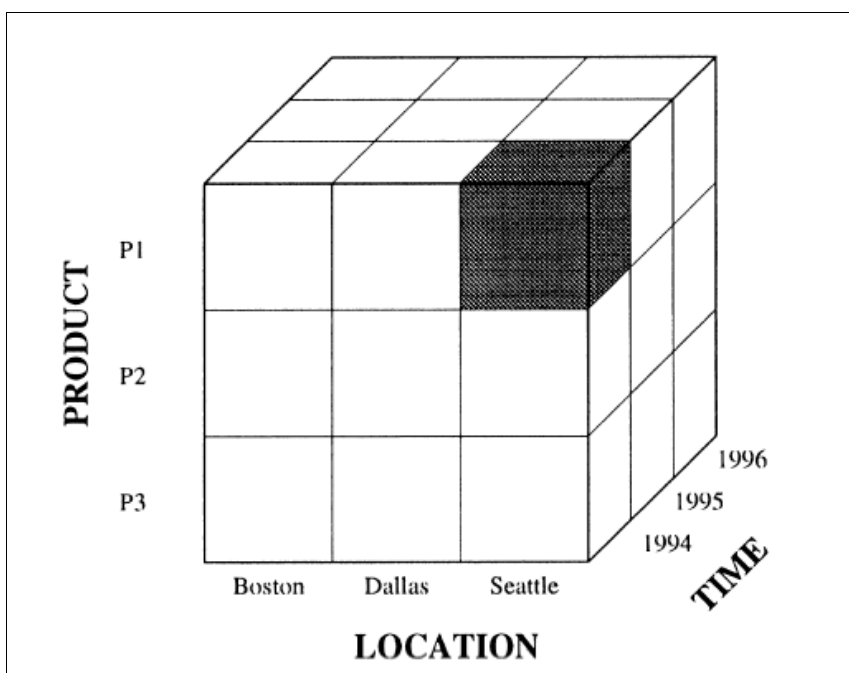


Figura 3. Cubo de dados para uma aplicação de vendas. Fonte: Datta e Thomas (1999).

A operação *dice* se refere à seleção de posições ou valores de uma dimensão. Selecionar “Dallas” como localização é um exemplo da operação *dice*. *Slice* e *dice* juntos têm o efeito de reduzir a dimensionalidade do cubo, possibilitando sua manipulação em modos bidimensionais de visualização.

2.4.2.2 Pivoting

O *pivoting*, ou rotação, está relacionado com a mudança dos eixos das dimensões, permitindo transformações na visualização dos dados. Deste modo, torna-se possível a um esquema de visualização bidimensional mostrar diversas dimensões. Para exemplificar, pode-se iniciar uma análise com o *Tempo* fixo e visualizar as medidas referentes às dimensões *Produto* e *Região*. Posteriormente, através de uma operação de rotação, pode-se fixar *Região* e visualizar os valores dos produtos ao longo do tempo. E, assim, é possível aplicar a operação sucessivamente, realizando as rotações necessárias para a análise requerida.

2.4.2.3 Drill-down

Genericamente, denomina-se *drilling* o processo, normalmente realizado com o auxílio de interfaces gráficas sofisticadas, que permite ao analista do negócio acessar vários níveis hierárquicos de dados sumarizados. No caso específico da operação *drill-down*, o processo tem início em dados mais sumarizados navegando para níveis hierárquicos mais baixos, revelando dados mais detalhados. Estes dados podem ser também sumarizados, porém, num nível de detalhe mais apurado. Este processo pode continuar indefinidamente, até que se atinja o nível base definido no *data warehouse*, que não pode mais ser sumarizado (BARBIERI, 2001). Como exemplo, uma dimensão *Tempo* com os atributos *dia*, *mês*, *ano*, *semestre* e *trimestre*, pode ter uma hierarquia como a mostrada na Figura 4, onde sua ordem hierárquica, do nível mais agregado para o menos agregado, seria *ano*, *semestre*, *trimestre*, *mês* e *dia*.

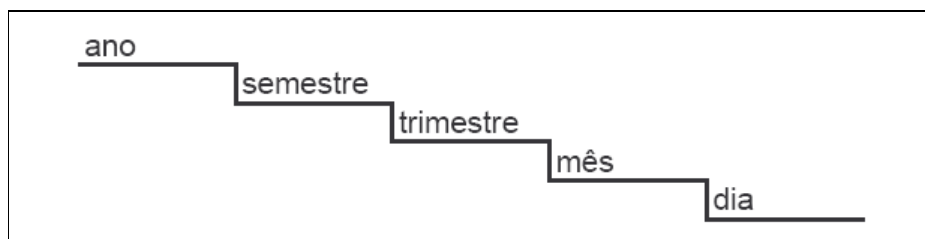


Figura 4. Exemplo de hierarquia da dimensão *Tempo*. Fonte: Galante, 2004.

2.4.2.4 *Roll-up*

A operação *roll-up*, também conhecida como *drill-up*, consiste no proceder oposto ao do *drill-down*, atuando no sentido da navegação de dados menos sumarizados para aqueles mais sumarizados, subindo no nível hierárquico dos dados. Este processo, de modo analogamente oposto ao *drill-down*, pode continuar até que se atinja o mais alto nível hierárquico definido. As operações de *roll-up* e *drill-down* são indispensáveis em qualquer ferramenta OLAP ou que se destine à manipulação dimensional.

2.4.2.5 *Drill-across*

A operação *drill-across* consiste na possibilidade de transferir a análise de uma tabela fato a outra, desde que estas tenham dimensões em conformidade. Dimensões em conformidade são aquelas em que duas de suas instâncias são idênticas, ou, por exemplo, se duas tabelas fato têm uma dimensão *Cliente*, esta irá estar em conformidade se os seus campos utilizados têm o mesmo domínio (KIMBALL, 2003). Na definição de Kimball e Ross (2002),

as dimensões estão em conformidade quando são exatamente idênticas (inclusive as chaves), ou quando uma é um subconjunto perfeito da outra. O mais importante é que os cabeçalhos das linhas produzidas em conjuntos de respostas das duas dimensões em conformidade devem ser capazes de coincidir perfeitamente.

Assim, uma operação de *drill-across* representa a navegação de um cubo a outro (ABELLÓ, SAMOS e SALTOR, 2002), em estruturas conhecidas como constelações (GIOVINAZZO, 2000), numa alusão a um esquema estrela (KIMBALL, 1996) contendo mais de uma estrela.

2.4.2.6 Drill-through

Uma operação *drill-through* é executada quando o analista do negócio deseja obter informações num nível de granularidade menor do que aquele definido na tabela fato. Para exemplificar, um *data warehouse* no domínio de vendas a varejo define uma tabela fato de Venda com uma granularidade de Produto por Dia por Loja. Isto significa que o dado mais detalhado que se poderá obter é a sumarização das vendas diárias por produto e loja.

Ainda neste contexto, sabe-se, evidentemente, que os dados de granularidade ainda mais baixa, no nível do item de venda, existem, porém não foram carregados no *data warehouse* em virtude da modelagem deste não requisitar estes dados, existindo apenas no sistema transacional de origem ou em alguma outra fonte, como algum outro ambiente de *data warehouse*.

O *drill-through* consiste na capacidade de obter dados de fontes externas que sejam capazes de revelar dados em níveis de detalhamento mais altos do que é possível com a estrutura do *data warehouse*. O *drill-through*, assim, seria um *drill-down* cujos dados para um nível mais baixo de granularidade seria obtido em fontes externas, fora do *data warehouse*.

2.4.3 OLAP Espacial

OLAP espacial, ou SOLAP (*Spatial OLAP*), pode ser definido, segundo Bédard (1997)⁸ *apud* Rivest, Bédard e Marchand (2001), como

uma plataforma visual construída especialmente para suportar análise e exploração de dados espaço-temporais de uma forma rápida e fácil, seguindo uma abordagem multidimensional que consiste de níveis de agregação disponíveis em visualizações cartográficas, bem como em formatos tabulares ou diagramas.

Entende-se, assim, OLAP espacial como aplicações cliente que realizam interfaces para processamento analítico em *data warehouses* espaciais. Segundo Rivest, Bédard e Marchand (2001), o projeto de uma aplicação OLAP espacial implica em determinar as dimensões

⁸ BÉDARD, Yvan. Spatial OLAP. Videoconferência no 2éme Forum Annuel sur la R-D Géomatique. VI: Un Monde Accessible. Montreal, Canadá, 1997.

espaciais necessárias, as medidas espaciais a serem incluídas e os operadores espaciais disponíveis para a realização da análise espacial.

No que diz respeito às dimensões, estas podem dividir-se em três tipos (RIVEST, BÉDARD e MARCHAND, 2001):

- a) as dimensões espaciais não-geométricas dizem respeito apenas a dados nominais, como nomes de lugares, não possuindo, assim, nenhuma associação com qualquer representação geométrica ou cartográfica (geo-referenciamento). Este é o tipo de dimensão com dados relativos ao espaço encontrada em sistemas OLAP convencionais;
- b) as dimensões espaciais geométricas possuem atributos que armazenam formas geométricas relativas a um mapa geo-referenciado, de modo a permitir que suas informações sejam visualizadas num formato cartográfico. Para se caracterizar como uma dimensão espacial geométrica, uma dimensão deve ter todos os seus atributos nas hierarquias sendo geométricos;
- c) as dimensões espaciais mistas são análogas às dimensões espaciais geométricas, porém não são obrigadas a manter todos os dados hierárquicos com dados geométricos, podendo ter, de modo misto, dados convencionais e geométricos numa mesma hierarquia.

Em sistemas OLAP espacial, não apenas as dimensões, mas também as medidas podem possuir caráter geográfico, apesar disto não ser abordado no presente trabalho.

Ainda de acordo com Rivest, Bédard e Marchand (2001), uma aplicação OLAP espacial deve implementar um conjunto de características, tais como:

- a) representação dos dados em formato cartográfico e não-cartográfico, simultânea e coordenadamente;
- b) visualização flexível, permitindo que o decisor tenha a opção de visualizar dados cartográficos e tabulares em diferentes granularidades ou, de modo sincronizado, na mesma granularidade; além disto devem ser oferecidas diferentes opções de visualização, como gráficos de barras, gráficos de torta, historamas *etc.*;
- c) prover um mapa de contexto, de modo que as visualizações cartográficas possam estar em um formato inteligível para o analista/decisor, permitindo que este localize os dados plotados;

- d) permitir a representação simultânea de mais de um conjunto de resultados, no mesmo mapa ou em mapas diferentes, utilizando diferentes variáveis visuais;
- e) permitir a alteração da semiologia gráfica, de acordo com as necessidades dos usuários, para que a visualização dos dados se adeque às regras semiológicas do negócio sendo analisado;
- f) apresentação opcional de legendas interativas, com resolução de conflitos;
- g) possibilidade de inclusão de medidas calculadas a partir daquelas já existentes no cubo;
- h) filtragem de membros de uma dimensão, de modo que o usuário final possa restringir os dados sendo visualizados aos dados de interesse;
- i) operações OLAP como *drill-down* ou *roll-up* devem estar diretamente disponíveis em todas as representações, sejam tabulares (em forma de tabelas, diagramas, gráficos, histogramas *etc.*) ou cartográficas. Uma vez realizada uma operação em um item de visualização, esta deve ser instantaneamente replicada a todas as outras representações visíveis.

2.5 POSTGRESQL

O PostgreSQL⁹ é classificado como um sistema gerenciador de banco de dados objeto-relacional, ou seja, é um SGBD capaz de suportar a criação de tipos de dados definidos pelo usuário. A ferramenta descrita neste trabalho, o PostGeoOlap, foi desenvolvida visando o funcionamento em conjunto com este SGBD, exatamente devido a estas características. PostgreSQL é um SGBD extremamente robusto, flexível e confiável, provendo a maioria dos recursos dos modernos SGBDs, além de ser compatível com ANSI/SQL99.

Além disto, PostgreSQL é um SGBD distribuído sob uma licença no estilo BSD (*Berkeley Software Distribution*)¹⁰, que é uma licença clássica¹¹ no mundo *open-source* e permite uso irrestrito e sem qualquer ônus do *software* sob seus auspícios, o que é inteiramente

⁹ <http://www.postgresql.org>, acesso em 11/03/2007.

¹⁰ <http://www.postgresql.org/docs/faqs.FAQ.html#1.2>, acesso em 11/03/2007.

¹¹ <http://www.opensource.org/licenses/bsd-license.php>, acesso em 11/03/2007.

adequado à perspectiva de liberdade de uso com a qual a ferramenta PostGeoOlap é construída. Vale notar que a licença do PostgreSQL não conta com a controversa cláusula 3, “*the obnoxious BSD advertising clause*”¹², como o Projeto GNU¹³ a alcunhou.

O PostgreSQL possui, ainda, além de versão para sistemas operacionais Windows 32 *bits*, versões para os sistemas operacionais ISO/IEC 9945 (Unix) e GNU/Linux Standard Base (Linux), tornando, assim, possível a implementação de uma solução completa disponível como Software Livre no servidor.

Somando-se a isto o fato de a ferramenta PostGeoOlap, cuja execução se dá nas máquinas clientes, ser construída na plataforma Java e, por isto, efetivamente multiplataforma, tem-se uma solução completa disponível em Software Livre.

2.5.1 Desempenho e Viabilidade

No que diz respeito ao desempenho do PostgreSQL em um ambiente de *data warehousing/OLAP*, Almeida (2004), em um estudo de viabilidade, concluiu que o PostgreSQL não é adequado para este tipo de aplicação. Segundo o trabalho, o SGBD não executara satisfatoriamente os testes de desempenho de um *benchmarking* especializado em ferramentas de suporte à decisão, o TPC-H¹⁴. As falhas apontadas recaem principalmente sobre aspectos de otimização de consultas e funcionalidades de agregação. Contudo, a versão do SGBD analisada no trabalho citado foi a 7.4. Nas versões atuais, na linha 8.x, o PostgreSQL tem resolvido muitas das questões apontadas, além de melhorias substanciais no concernente a otimização de consultas.

Além disto, visando cobrir lacunas como as apontadas acima, os desenvolvedores do PostgreSQL iniciaram no ano de 2005 o projeto BizGres¹⁵, cujo objetivo é tornar o PostgreSQL um banco bastante robusto para *data warehousing* em particular e *business intelligence* em geral. Ainda, é de alta relevância para o projeto GeoOlap o fato de que o BizGres, apesar de tratar de toda a infraestrutura para um SGBD orientado a *data warehousing*

¹² <http://www.gnu.org/philosophy/bsd.html>, acesso em 11/03/2007.

¹³ <http://www.gnu.org>, acesso em 11/03/2007.

¹⁴ <http://www.tpc.org/tpch/default.asp>, acesso em 12/03/2007

¹⁵ <http://www.bizgres.org>, acesso em 12/03/2007.

e processamento analítico, não prevê o desenvolvimento de ferramentas específicas para a realização deste processamento. Esta lacuna pode ser perfeitamente preenchida pelo PostGeoOlap.

2.5.2 PostGIS

Para as características de armazenamento e processamento geográfico necessárias ao funcionamento de uma ferramenta OLAP espacial, são necessárias funcionalidades que o PostGreSQL, *per se*, não dispõe, existindo, portanto, a necessidade de extensões. Tais funcionalidades geográficas são implementadas pela extensão PostGIS¹⁶, que adiciona ao PostGreSQL o suporte a objetos geográficos, tornando-o, na prática, um SGBD espacial.

O PostGIS é aderente à especificação OpenGIS que padroniza o acesso a feições geográficas simples com SQL (OPENGIS, 1999), está para ser submetido a testes de conformidade com a versão 1.0 da citada especificação. O suporte a esta especificação permite a operação com comandos geométricos textuais no formato WKT (Well-Known Text).

Além disto, o PostGIS também é consonante com os preceitos *open-source*, sendo distribuído sob a licença GPL¹⁷, garantindo assim livres uso e distribuição.

O PostGIS possui um grupo ativo trabalhando em seu desenvolvimento e tem como proposta de futuras implementações a melhoria do mecanismo de carga e descarga de dados, o desenvolvimento de ferramentas de interface como o usuário para o acesso e manipulação direta de dados e o suporte a estruturas topológicas avançadas, como coberturas, superfícies e redes.

2.6 REPRESENTAÇÃO VISUAL MULTIDIMENSIONAL

O processamento de dados orientado à análise visa permitir que dados relativos a um dado domínio possam ser analisados com base em diferentes visões, que, na análise

¹⁶ <http://postgis.refrations.net>, acesso em 12/03/2007.

¹⁷ <http://www.gnu.org/copyleft/gpl.html>, acesso em 12/03/2007.

dimensional, são conhecidos como dimensões. Porém, ao tratar com múltiplos pontos de vista, ou múltiplas dimensões, vêm à tona o problema de uma correta e adequada representação visual multidimensional.

No campo das artes plásticas, o cubismo tentara reproduzir visualmente várias dimensões ao combinar múltiplos pontos de vista em uma imagem única (BERNARD e CARTER, 2004), podendo assim ser considerado uma tradução artística da idéia de OLAP.

Modelos OLAP utilizam como elemento central de sua arquitetura o conceito de cubo. A metáfora do cubo utilizada em análise dimensional tem se mostrado didaticamente adequada, conforme a imensa quantidade de material introdutório sobre *data warehousing* e OLAP que utiliza graficamente esta metáfora para ilustrar conceitos inerentes à disciplina. O cubo representa muito bem a idéia de multidimensionalidade até três dimensões. Contudo, caso se acrescente uma quarta dimensão, a metáfora falha completamente. Representações visuais de tipo hipercúbico são extremamente complexas e de utilidade nula se o objetivo é comunicação de informação de modo simples e objetivo.

Conforme ensina Wittgenstein (2001), para que uma coisa possa representar outra, é preciso que ambas compartilhem atributos estruturais básicos. Thomsen (2002) retoma esta idéia, atribuindo a falha da metáfora do cubo para mais de três dimensões à diferença entre dimensões lógicas e físicas. Um importante atributo de dimensões físicas é a angularidade, de modo que os eixos das diferentes dimensões são perpendiculares entre si. Nas dimensões lógicas tratadas no âmbito de modelos OLAP, não existe um atributo equivalente à angularidade das dimensões físicas. Não faz qualquer sentido afirmar que produtos são perpendiculares a clientes ou que vendedores são perpendiculares ao tempo. Thomsen (2002) propõe uma estrutura, denominada Multidimensional Type Structure (MTS), onde cada dimensão é representada por um segmento de reta, para a substituição do cubo como metáfora para representação de dados multidimensionais, sem as restrições de complexidade existentes no trato visual com cubos multidimensionais, ou hipercubos.

Para além da representação lógica, é necessário que, para a implementação de sistemas de suporte à decisão baseados em OLAP, múltiplas dimensões possam ser visualizadas em um monitor de vídeo. Assim, surge a necessidade de mapear as múltiplas dimensões lógicas do modelo OLAP em uma representação física bidimensional adequada à visualização na tela do computador. Duas soluções coexistem para a implementação desta necessidade: (1) paginação e (2) combinação de mais de uma dimensão lógica em uma mesma dimensão física.

A paginação pura consiste em se ter apenas duas dimensões lógicas plenamente visíveis, uma em cada eixo do plano bidimensional, com as dimensões restantes tendo valores fixos em seus atributos, de modo que uma visualização instantânea seria como uma “fatia” de um hipercubo, conforme Figura 5. Na citada figura, a dimensão *Product* é fixada em um valor (no caso, “shoes” ou, em português, sapatos), com cada instância possível de produto sendo uma página. A visualização propriamente dita não ocorre para a dimensão de página, ela apenas atua como um indexador.

page		columns				
Product: shoes		Variables: all				
rows		Sales	Direct Costs	Indirect Costs	Total Costs	Margin
	January	520	320	110	430	90
	February	400	250	130	380	20
	March	430	300	120	420	10
	April	490	320	150	470	20
	May	520	310	180	490	30
	June	390	230	150	380	10
	July	470	290	160	450	20
	August	500	360	150	510	-10
	September	450	290	140	430	20
	October	480	290	140	430	50
	November	510	310	150	460	50
	December	550	330	160	490	60
Time: months						

Figura 5. Paginação como representação visual n-dimensional. Fonte: Thomsen (2003).

A estratégia de combinar mais de uma dimensão lógica em uma mesma dimensão visual consiste em aninhar hierarquicamente uma dimensão dentro de outra da qual um exemplo pode ser mostrado na Figura 6, onde as dimensões *Tempo*, *Vendedor* e *Local* são apresentadas, com as duas últimas coexistindo no mesmo eixo visual, com *Local* subordinado a *Vendedor*.

		Venk				Netz				Pitz		
		USA				Japan	USA	Japan				
		USA_N		USA_S	Kanto			Kansai				
		Seattle	Boston		Tokyo			Yokohama	Osaka		Kyoto	
Qtr1	Jan											
	Feb											
	Mar											
Qtr2												
Qtr3												
Qtr4	Oct											
	Nov											
	Dec											

Figura 6. Grade multidimensional contendo as dimensões Vendedor, Local e Tempo.

Este trabalho focaliza na segunda alternativa, por ter maior poder e expressividade. Porém, as implementações desenvolvidas implementam também a primeira alternativa através da possibilidade do usuário estabelecer critérios de consulta em níveis hierárquicos de dimensões que não são visualizadas, o que, na prática, é exatamente o mesmo que prega a proposta de paginação.

2.7 CUBE PRESENTATION MODEL

O Cube Presentation Model (MANIATIS *et al.*, 2003) é um modelo de apresentação para visualização OLAP baseado na representação geométrica de um cubo e sua percepção humana no espaço. É composto por duas partes: uma camada lógica, cujo objetivo é a formulação de cubos; e uma camada de apresentação, que se destina à produção da visualização destes cubos. A existência de duas camadas distintas visa a uma clara separação entre entidades que detêm os dados e entidades que os apresentam, de modo que as duas camadas sejam independentes e possam funcionar isoladamente, em conjunto com outros esquemas. Isto é exatamente o que será feito na presente dissertação: a camada de apresentação do CPM funcionará em conjunto com o modelo lógico do PostGeoOlap.

Far-se-á aqui uma descrição breve. Para maiores detalhes sobre o CPM, pode-se consultar o trabalho que originalmente lhe definiu (MANIATIS *et al.*, 2003).

A camada lógica do CPM é uma extensão de uma proposta anterior (VASSILIADIS e SKIADOPOULOS, 2000), tornando o modelo capaz de computar cubos mais complexos do que o modelo da citada proposta permitia. A camada lógica define diversas entidades necessárias à formulação de um multicubo OLAP, como dimensão, conjunto de dados, condição de seleção, cubos e outras. Neste trabalho, a camada lógica do CPM não será abordada.

A camada de apresentação do CPM se baseia em representações geométricas para definir suas entidades, baseando-se em eixos e pontos. No CPM, um cubo de dados possui n eixos, sendo que 2 eixos são visuais e perpendiculares entre si, não podendo compartilhar dimensões. Cada eixo possui um conjunto de pontos que o caracterizam. O diagrama de classes UML (Unified Modeling Language) para a camada de apresentação do CPM pode ser visto na Figura 7.

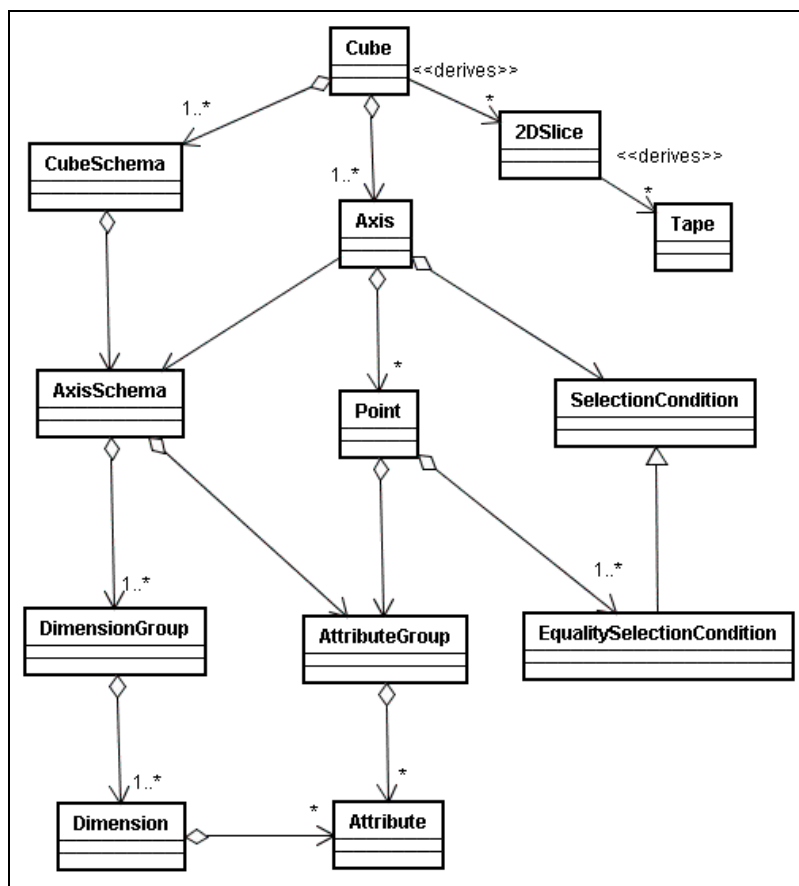


Figura 7. Camada de apresentação do CPM. Fonte: Maniatis *et al.* (2003)

O modelo do CPM estende o conceito de dimensão para incorporar qualquer tipo de atributo (inclusive medidas, resultados de funções etc). Deste modo, considera-se que todo

atributo que não está associado a alguma dimensão pertencente a uma dimensão de nível único com o mesmo nome do atributo. Uma dimensão como normalmente é concebida em modelos OLAP seria, na terminologia do CPM, uma dimensão de nível (*level dimension*). A definição de atributo-dimensão é nomeada dimensão de atributo (*attribute dimension*). Quando uma dimensão de atributo contém uma medida numérica, é chamada dimensão de medida (*measure dimension*).

Nas próximas seções, as principais entidades do CPM serão explicadas.

2.7.1 AttributeGroup

CPM define um *AttributeGroup* qualquer (denominado *AG*) sobre um conjunto de dados (*data set*, denominado *DS*) como um par $[A, DA]$, onde A é uma coleção de atributos pertencentes a *DS* e DA é uma coleção de atributos dependentes dos atributos de A . Como exemplo de um grupo de atributos, pode-se citar $AG = ([Venda, Custo], [Lucro])$, onde define-se que o atributo *Lucro* é dependente dos atributos *Venda* e *Custo*. A primeira lista é chamada de chave do *AttributeGroup*.

2.7.2 DimensionGroup

Um *DimensionGroup* qualquer (denominado *DG*) sobre um conjunto de dados *DS* é definido como um par $[D, DD]$, onde D é uma coleção de atributos pertencentes a *DS* e DD é uma coleção de dimensões dependentes das dimensões em D , sendo em tudo análogo à classe *AttributeGroup*. Como exemplo de um grupo de dimensões, pode-se citar $DG = ([Produto, Local])$.

Por razões de brevidade, um *AttributeGroup* ou *DimensionGroup* contendo apenas sua chave será representado somente por ela própria. Um grupo de dimensões contendo, por exemplo, apenas uma dimensão $([Produto])$ pode ser representado tão-somente como *Produto*.

2.7.3 Ancestor Function

Uma função $anc_{L_1}^{L_2}$, ou *ancestor*, é proveniente da camada lógica do CPM e estabelece que para cada par de níveis dimensionais L_1 e L_2 , com $L_1 \prec L_2$, onde \prec é uma ordenação parcial definida sobre os níveis de uma dimensão, a função mapeia cada elemento de $dom(L_1)$ a um elemento de $dom(L_2)$. Por $dom(L_n)$ entenda-se o domínio de um determinado nível hierárquico, ou seja, o conjunto de valores que este nível pode assumir. Apesar desta função ser originada da camada lógica, é descrita aqui por ser utilizada nas definições de algumas entidades da camada de apresentação.

É importante notar que o uso comum de entidades entre as camadas lógica e de apresentação aumenta a sua interdependência.

Como exemplo de *ancestor function*, tem-se $anc_{mês}^{ano}(Ano) = "2006"$, onde o atributo *ano*, na instância “2006”, é mapeado para instâncias do atributo *mês*, retornando um conjunto com as instâncias de *mês* associadas a “2006”.

2.7.4 SelectionCondition

Uma condição de seleção é um conceito originado da camada lógica do CPM e se constitui de uma cláusula na forma $x \partial y$, x e y podem ser níveis, valores ou *ancestor functions* e ∂ é um conectivo lógico. Um exemplo de *SelectionCondition* é $ano > 2004$.

2.7.5 EquallySelectionCondition

Uma condição de igualdade é uma especialização de *EquallySelectionCondition* da qual todas as instâncias possuem a igualdade como conectivo lógico. Um exemplo é a condição $ano = 2006$.

2.7.6 AxisSchema

Um *AxisSchema* representa os metadados de um eixo, sendo definido como um par $[DG, AG]$, onde DG é uma lista ordenada de *DimensionGroups* e AG uma lista ordenada de listas ordenadas, cada uma correspondendo a um *DimensionGroup* em DG . Cada uma das listas internas de AG é associada a um *DimensionGroup*, assinalando quais níveis hierárquicos estarão sendo para cada dimensão no eixo que tiver o *AxisSchema* corrente como esquema. Formalmente, um objeto *AxisSchema* é definido como

$$AS_K = \left([DG_1 \times DG_2 \times \dots \times DG_K], \right. \\ \left. \llbracket ag_1^1, ag_1^2, \dots, ag_1^{K_1} \rrbracket \times [ag_2^1, ag_2^2, \dots, ag_2^{K_2}] \times \dots \times [ag_K^1, ag_K^2, \dots, ag_K^{K_K}] \rrbracket \right).$$

Como instância de um esquema de eixo, tem-se

$$Horizontal_S = \{ [Vendedor \times Local], [vendedor] \times [cidade, região, país] \},$$

onde *Vendedor* e *Local* são as dimensões previstas para o eixo e os grupos contendo *vendedor* e *cidade, região e país* são, respectivamente, os níveis hierárquicos definidos para as dimensões do eixo.

2.7.7 Axis

Um eixo (do inglês *axis*) pode ser visto como um conjunto de pontos (no caso de um eixo cartesiano, dispostos em uma reta). Além dos dois eixos perpendiculares entre si necessários à representação física bidimensional, CPM prevê dois tipos especiais de eixos: *Invisible* e *Content*. O eixo *Invisible* abriga as dimensões que não estão presentes nos outros eixos, ou seja, dimensões que estão sendo ignoradas na análise corrente. O eixo *Content* tem a função de, caso haja apenas um atributo de medida no cubo corrente, conter a “dimensão” da medida.

Compete esclarecer que, caso haja mais de um atributo de medida sendo analisado no cubo, a “dimensão” fato passa a ser uma dimensão visual real em um dos eixos. Caso haja apenas um atributo de medida, a dimensão visual desaparece e o atributo único é representado apenas por seus valores no plano. A Figura 8 mostra um exemplo fictício, onde uma tabela visual OLAP mostra duas dimensões *Tempo* (no eixo vertical) e *Produto* (no eixo horizontal). A primeira é mostrada em seu atributo *ano* e a segunda em seu atributo *Categoria*. Os valores mostrados correspondem ao valor das vendas. O suposto usuário solicitou apenas o atributo *venda* da tabela fato. Portanto a tabela-fato não aparece nos cabeçalhos como uma dimensão visual, nem nos metadados dos eixos visuais, mas no eixo *Content*.

	Eletrônicos	Acessórios
2003	139.872,00	12.121,00
2004	128.976,00	32.434,00
2005	122.121,00	23.211,00
2006	132.233,00	12.321,00

Figura 8. Visualização OLAP com fato como dimensão implícita.

Porém, se mais um atributo medida da tabela for chamado à visualização, digamos, *margem*, a dimensão fato passa a constituir uma dimensão visual, como se pode ver na Figura 9. Na Figura 8, cada valor numérico na tabela se associava visualmente a uma instância de *Produto* e a uma instância de *Tempo*, sendo implicitamente ligada a um atributo da fato previamente selecionado (no caso, *valor*). Na Figura 9, porém, há mais de um atributo da tabela fato sendo visualizado, o que obriga a que esta se materialize visualmente como dimensão.

	Eletrônicos		Acessórios	
	Valor	Margem	Valor	Margem
2003	139.872,00	12.121,00	13.987,00	1.212,00
2004	128.976,00	32.434,00	12.897,00	3.243,00
2005	122.121,00	23.211,00	12.212,00	2.321,00
2006	132.233,00	12.321,00	13.223,00	1.232,00

Figura 9. Visualização OLAP com fato como dimensão visual explícita.

Formalmente, um eixo é definido como um par contendo um *AxisSchema* contendo K *DimensionGroups* e uma lista ordenada finita de K listas ordenadas finitas de condições de seleção, onde cada membro das listas internas envolve apenas a respectiva chave do *AttributeGroup*. Um eixo, assim pode ser definido como $a = (AS_K, [\varphi_1, \varphi_2, \dots, \varphi_K])$, onde

φ é uma condição de seleção (objeto *SelectionCondition*). Como exemplo de instância de um eixo, tem-se

$$Horizontal = \left\{ \begin{array}{l} Horizontal_S, \\ \left\{ \left[\begin{array}{l} vendedor = 'Venk', vendedor = 'Netz' \end{array} \right], \right. \\ \left. \left\{ \left[\begin{array}{l} anc^{região}_{cidade}(cidade) = 'USA_S', região = 'USA_S', país = 'USA' \end{array} \right] \right\} \right\} \end{array} \right\},$$

onde *Horizontal_S* é o esquema para o eixo, seguido das condições de igualdade para os níveis hierárquicos.

Deste modo, um eixo é completamente definido pelo seu esquema, e alterações da estrutura do *AxisSchema* acarretarão necessariamente em alterações no(s) eixo(s) por ele representado(s).

2.7.8 Point

O ponto é a unidade básica do CPM e corresponde ao conceito de ponto na geometria. Porém, a classe *Point* não modela pontos quaisquer num espaço n-dimensional, mas apenas os pontos que se situam sobre eixos. Como numa visualização 2-D um eixo visual eventualmente conterá mais de uma dimensão – e, neste caso, necessariamente mais de um atributo – um objeto *Point* é definido como um par formado por um conjunto de *AttributeGroups* e um conjunto de mapeamentos atributo/valor (na forma de objetos *EquallySelectionCondition*), com cada mapeamento se referindo a cada um dos grupos.

Formalmente, um ponto é definido como um par contendo um conjunto de *AttributeGroups* e um conjunto de condições de seleção de igualdade (objetos *EquallySelectionCondition*) para cada uma das chaves de cada grupo de atributos.

2.7.9 CubeSchema

Um *CubeSchema* representa os metadados de um cubo, sendo definido como um conjunto finito de objetos *AxisSchema*. A chave de cada objeto *DimensionGroup* deve pertencer

a apenas um eixo. Os atributos pertencentes a um *DimensionGroup* têm que estar todos no mesmo eixo. Definem-se dois eixos de propósito especial, *Invisible* e *Content*, sendo que este último pode apenas conter dimensões de medidas; cada medida é ligada a uma função de agregação sobre uma medida do *data warehouse*; todas as dimensões de nível (*level dimensions*) no conjunto de dados são encontradas na união dos objetos *AxisSchema*, contidas nos eixos visuais ou no eixo *Invisible*.

2.7.10 Cube

Um cubo é definido como um conjunto de eixos (objetos *Axis*), cujos metadados (objetos *AxisSchema*) em conjunto formam um objeto *CubeSchema*.

2.7.11 2D-Slice

Um *2D-Slice* sobre um cubo *C* composto de K eixos é um conjunto de $(K - 2)$ pontos, cada um de um eixo diferente. Um *2D-Slice* evoca o intuitivo conceito de fatia de um cubo, fixando os eixos do cubo em pontos específicos, exceto os dois eixos visuais, que variam no espaço, tornando possível a apresentação bidimensional. Uma grade de visualização OLAP é a materialização de um *2D-Slice*.

2.7.12 Tape

Seja um *2D-Slice* (denominado *SL*) qualquer sobre um cubo composto de K eixos. Um *tape* sobre *SL* é um conjunto de $(K - 1)$ pontos, onde $(K - 2)$ pontos são os pontos de *SL*. Um *tape* sempre é paralelo a um eixo específico, pois um dos eixos não fixos (visuais) de *SL* também é fixado a um ponto específico, podendo ser considerado, apenas para efeito de compreensão, uma “linha” ou “coluna” do *2D-Slice*. Na verdade, um *tape* pode abranger mais de uma linha ou coluna física, pois pode ser delimitado por uma *ancestor function*.

2.7.13 Cross-join

Seja um *2D-Slice* qualquer SL sobre um cubo composto de K eixos. Sejam também dois *tapes* t_1 e t_2 não paralelos entre si. Um *cross-join* sobre t_1 e t_2 é um conjunto de K pontos, onde $(K - 2)$ pontos são os pontos do SL e cada um dos dois pontos restantes advém de cada um dos eixos visuais. Em outras palavras, um *cross-join* constitui-se do conjunto de condições de igualdade resultantes da interseção entre dois *tapes*.

3 O PROJETO GEOOLAP

Nos dias atuais, sistemas de suporte à decisão são ferramentas indispensáveis a grandes instituições. As vantagens advindas destes sistemas também podem e devem ser usufruídas por instituições de pequeno e médio porte. Contudo, impedimentos relativos a custo de aquisição de licenças, normalmente associados a este tipo de tecnologia, acabam por afastar muitas instituições do uso destes sistemas. O projeto GeoOlap é uma solução para este tipo de instituição, oferecendo um ambiente de baixo custo baseado em Software Livre para o desenvolvimento de aplicações de suporte à decisão desde o nível conceitual até a visualização.

O projeto GeoOlap provê um método unificado para a modelagem de sistemas multidimensionais com dados geo-referenciados onde as dimensões em um esquema estrela podem conter atributos geográficos. Deste modo, pretende reunir muito do necessário para a criação de sistemas OLAP espaciais desde a fase de concepção de uma aplicação. A aplicação da metodologia prescrita pelo GeoOlap é indicada especialmente para instituições que ainda não possuem aplicações OLAP e SIG ou não desejam integrar as pré-existentes.

De acordo com o GeoOlap, *data warehouses* espaciais podem ser modelados conceitualmente com o auxílio de diagramas de classes UML, valendo-se de estereótipos geográficos para a representação das entidades geográficas. Segundo Trujillo *et al.* (2001), o uso de UML em lugar de modelos tradicionais como entidade-relacionamento pode ser justificado quando se considera que as propriedades dinâmicas e estruturais de um sistema de

informação no nível conceitual podem ser modeladas mais naturalmente com a modelagem orientada a objetos e a UML do que com abordagens clássicas como o modelo E-R. Além disto, a UML oferece o OCL (Object Constraint Language) para a especificação de requisitos e restrições e suporta a representação de estereótipos, que podem simplificar a representação de extensas hierarquias de classes.

Além da proposta de modelagem, o projeto prevê uma ferramenta OLAP espacial, o PostGeoOlap, que constitui o objeto do presente trabalho, para viabilizar a criação de sistemas de suporte à decisão baseados em *data warehousing* utilizando dados convencionais e geográficos. Assim, o GeoOlap compreende a totalidade do processo de desenvolvimento para a criação de uma aplicação OLAP espacial desde sua fase conceitual.

O processo preconizado pelo GeoOlap é composto das seguintes atividades:

- a modelagem do *data warehouse* utilizando UML com estereótipos geográficos;
- mapeamento do modelo conceitual expresso em UML para o esquema lógico do *data warehouse* espacial;
- o uso de uma ferramenta OLAP espacial – o PostGeoOlap – para manipular o *data warehouse* com o intuito de oferecer ao usuário recursos para pesquisar, analítica e geograficamente, os dados e visualizar os resultados, simultaneamente, de forma tabular e cartográfica.

3.1 ESTEREÓTIPOS GEOGRÁFICOS

O Open Geospatial Consortium (OGC), através de suas especificações OpenGIS, permite que qualquer objeto geográfico seja modelado e representado. Assim, o mesmo tipo de dado definido no modelo conceitual poderá ser utilizado no modelo lógico e no esquema de banco de dados, sem qualquer transformação ou conversão de conceitos.

Colonese *et al.* (2005), no âmbito do projeto GeoOlap, propõem um modelo de *data warehouse* integrando conceitos multidimensionais e espaciais em um único diagrama. Para isto, são empregados estereótipos geográficos para representar as classes geográficas e a ausência de estereótipos para a representação das classes convencionais, sem atributos geográficos.

Tipo de Dados	Estereótipo
<i>Point</i>	•
<i>LineString</i>	/
<i>Polygon</i>	◻
<i>MultiPoint</i>	••••
<i>MultiLineString</i>	
<i>MultiPolygon</i>	◻◻◻◻
<i>GeometryCollection</i>	★

Tabela 2. Estereótipos para os tipos de dados da OpenGIS.

Uma abordagem similar foi utilizada no projeto GeoFrame (LISBOA FILHO e IOCHPE, 1999), porém com atenções voltadas para aplicações OLTP. A Tabela 2 apresenta os estereótipos geográficos propostos por Colonese (2004) para os tipos de dados da OpenGIS.

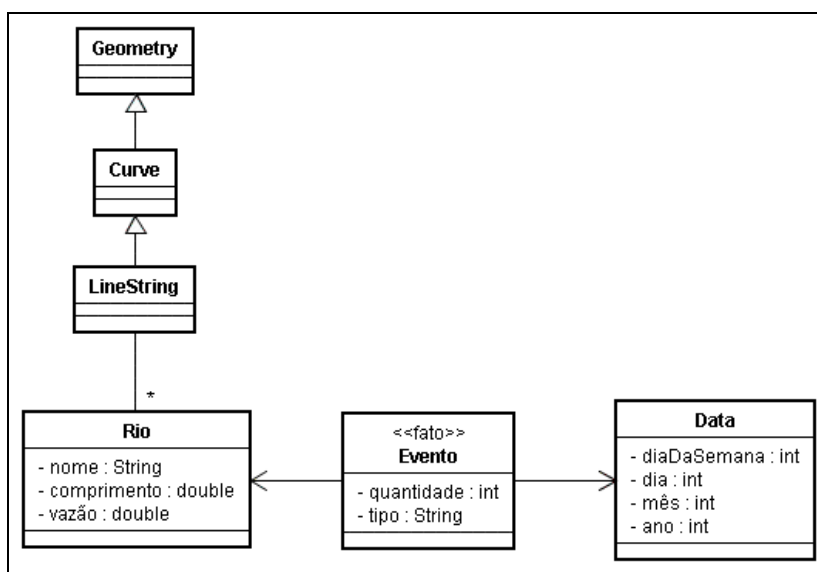


Figura 10. Associação com uma hierarquia extensa para a dimensão Rio.

A representação por estereótipos é vantajosa, na medida em que deixa o diagrama mais limpo e de fácil compreensão, mantém a riqueza semântica e simplifica a coexistência de conceitos de diferentes domínios – no caso, conceitos geográficos e não-geográficos. A Figura 11 mostra o uso de um estereótipo para representar a associação entre a classe **Rio** e a hierarquia de tipos geométricos à qual **LineString** pertence, abreviando a representação estendida vista na Figura 10.

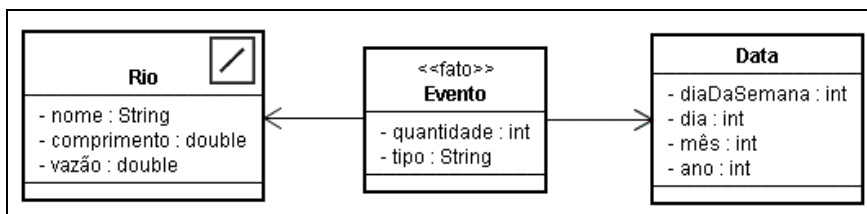


Figura 11. Estereótipo representando a associação de Rio com LineString.

Posteriormente, Malinowski e Zimányi (2004) e Pestana e Silva (2005) também propuseram estereótipos para representação de conceitos geográficos em modelos conceituais multidimensionais.

3.2 A FERRAMENTA POSTGEOOLAP

PostGeoOlap é uma ferramenta para criação de soluções OLAP espaciais, projetada para funcionar sobre o SGBD PostGreSQL e sua extensão espacial, o PostGIS. O nome PostGeoOlap, assim, teve origem no propósito da ferramenta: a integração entre características geográficas, tecnologias OLAP e o SGBD PostGreSQL.

3.2.1 Princípios de Projeto

Para a construção da ferramenta PostGeoOlap, adotou-se o paradigma de Software Livre como princípio de projeto. Assim, todas as APIs, *frameworks* e *softwares* de persistência de dados utilizados neste projeto são *open-source*. Deste modo, pode-se atingir um dos objetivos do projeto, que é prover uma plataforma de baixo custo para OLAP espacial, permitindo a pequenas e médias organizações o desenvolvimento de aplicações de *data warehousing* e SIG, o que outrora, lhes seria dificultado devido aos altíssimos custos de licenciamento de soluções proprietárias para estes fins.

Para o projeto da ferramenta PostGeoOLAP adotou-se ROLAP como modelo de armazenamento do *data warehousing* para obter as vantagens da maturidade da tecnologia de SGBD objeto-relacionais, que possuem a capacidade de armazenar tanto dados convencionais quanto geográficos e usar funções de agregação e funções espaciais sobre eles, além de permitir

a adição customizada de funcionalidades. Assim, ambas as pesquisas, geográficas ou analíticas convencionais, serão processadas e respondidas pelo SGBD PostGreSQL e todos os dados, desde o nível base até as agregações, serão mantidos no modelo relacional.

3.2.2 Modelo

O diagrama de classes na Figura 12 representa os metadados utilizados originalmente pelo PostGeoOlap para a manutenção dos dados no *data warehouse*. A classe *Esquema* representa o banco de dados (*database*) no PostGreSQL que contém o *data warehouse* de interesse. A classe *Cubo* representa um hipercubo, ou, no nível do usuário, cada perspectiva de negócio pela qual um *Esquema* pode ser analisado. A classe *Tabela* representa todas as relações existentes no *data warehouse*. A classe *Dimensao* é uma subclasse de *Tabela* e se refere a todos os componentes de um cubo, tanto tabelas fato quanto dimensões. A classe *Atributo* representa os dados existentes em cada *Tabela*. O auto-relacionamento indica que um atributo convencional está funcionando como rótulo para um ou mais atributos geográficos. A classe *Agregacao* representa as várias hierarquias para os atributos de uma dimensão. A classe *ItemHierarquia* aloca cada *Atributo* a uma *Hierarquia*.

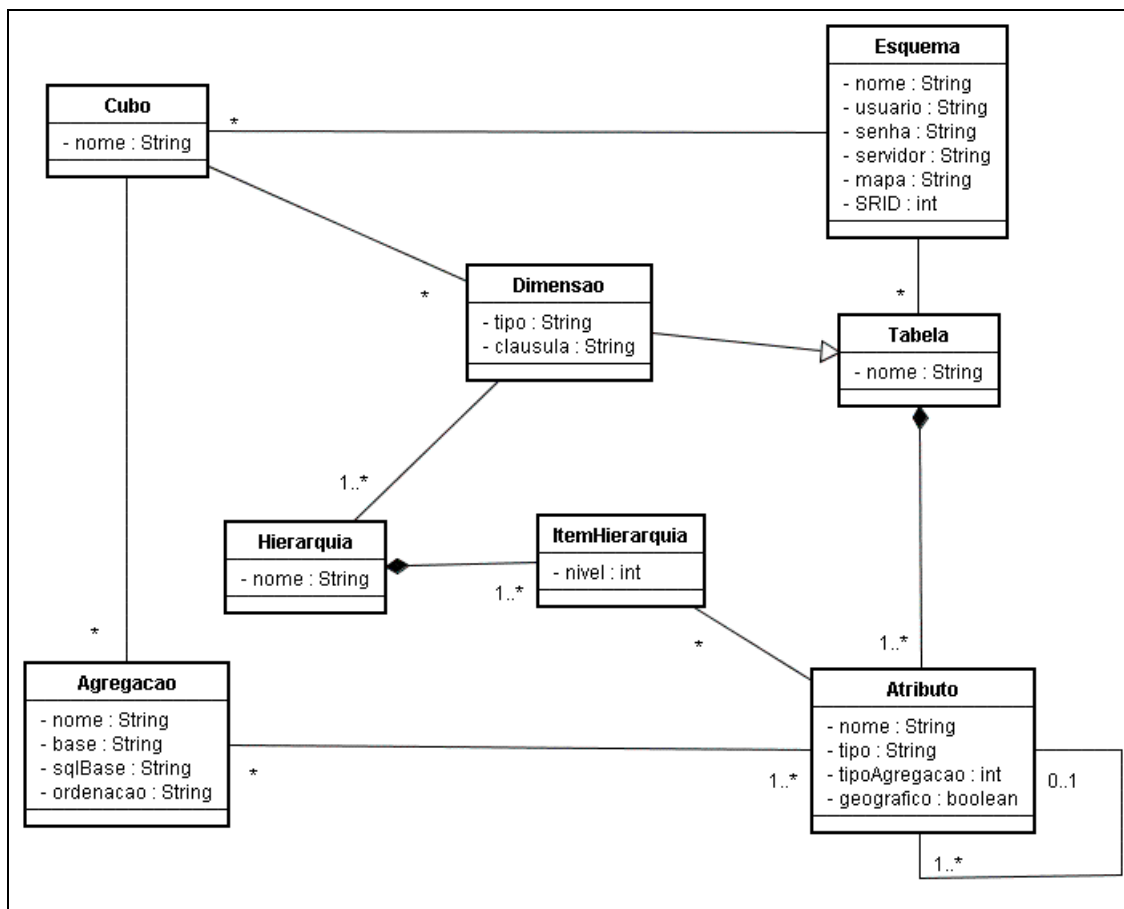


Figura 12. Modelo conceitual do PostGeoOlap. Fonte: Colonese (2004)

A implementação descrita em Colonese (2004) não realizou este modelo completamente, restringindo o número de hierarquias a um por dimensão, levando a um modelo sem as classes *Hierarquia* e *ItemHierarquia* e com o deslocamento do atributo *nivel* da classe *ItemHierarquia* para a classe *Atributo*, conforme mostra a Figura 13.

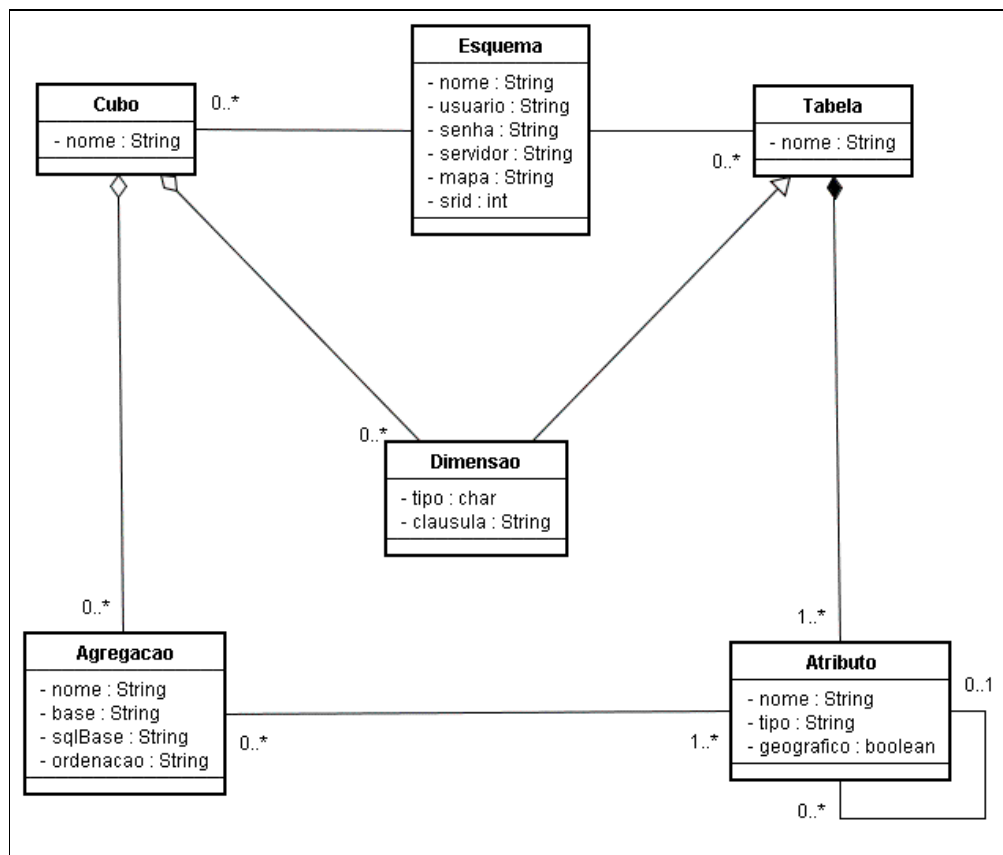


Figura 13. Diagrama original de projeto do PostGeoOlap. Fonte: Colonese (2004)

3.2.3 Funcionalidades

Os casos de uso do PostGeoOlap estão resumidos na Tabela 3. Os casos de uso *Criar esquema*, *Criar cubo* e *Adicionar dimensão* trabalham na construção da estrutura dos cubos que serão verificados e processados pelo PostGeoOlap.

CASO DE USO	DESCRIÇÃO
Criar esquema	Cria uma conexão com um banco de dados (<i>database</i>) no PostgreSQL e persiste os metadados necessários.
Criar cubo	Cria um cubo dentro de um esquema, selecionando uma tabela fato e definindo seus itens numéricos e as operações desejadas sobre estes itens.

Adicionar dimensão	Cria uma perspectiva para a análise dos dados contidos na tabela fato previamente definida, selecionando uma das tabelas relacionadas ao esquema. Também define a hierarquia dos atributos da dimensão, atribuindo um nível numérico para cada um.
Processar cubo	Verifica a massa de dados referenciada pelo cubo e tenta inferir o desempenho de consultas em tempo de execução. As consultas avaliadas como tendo um baixo desempenho são otimizadas através de agregações. Isto permite que o cubo possa ser analisado através de qualquer das dimensões, dentro de um tempo de resposta aceitável.
Adicionar dimensão não agregável	Cria uma dimensão que, apesar de não constituir uma perspectiva de análise e, assim, não participando do processamento do cubo, serve como referência para comparações geográficas com outras dimensões que possuam atributos geográficos.
Analisar dados	Oferece uma interface visual que permita a seleção de atributos para consulta utilizando restrições convencionais e/ou geográficas. Os resultados são visualizados tanto na forma tabular quanto na forma cartográfica, caso necessário.

Tabela 3. Lista de casos de uso do PostGeoOlap.

Após a definição do esquema e do cubo, com a adição das dimensões, a ferramenta está pronta para o processamento do cubo. Nesta operação, explicitada pelo caso de uso *Processar cubo*, o desempenho das pesquisas no cubo é inferido e, caso caia para abaixo de um patamar mínimo estabelecido na definição do cubo, a aplicação cria uma nova estrutura de agregação representada por uma tabela. A agregação é realizada em três passos:

- a) criação da tabela que conterá os dados agregados;
- b) transferência dos dados agregados para a nova tabela;
- c) criação dos índices para a nova tabela, utilizando estruturas de árvore-B para atributos convencionais e GiST (*Generalized Search Tree*) (HELLERSTEIN, NAUGHTON e PFEFFER, 1995) para os atributos espaciais. Ambas as estruturas de indexação são oferecidas pelo PostGreSQL e seu uso para as respectivas finalidades é recomendado na documentação do SGBD.

Durante o processamento do cubo, a ferramenta sempre inicia com a geração das agregações na perspectiva mais alta. Isto significa que para cada dimensão, os atributos devem receber um nível hierárquico, de 9 (representando a informação menos agregada) até 1 (para a informação mais agregada), com vários atributos podendo compartilhar o mesmo nível hierárquico. Cada agregação em uma perspectiva é gerada utilizando a maior perspectiva abaixo dela, caso haja, o que traz evidentes ganhos em desempenho.

Uma dimensão não agregável é uma dimensão de natureza geográfica. Ela não participa do processamento do cubo e não gera agregações, daí a sua classificação. O propósito deste tipo de dimensão é servir de referencial para comparações com outras dimensões contendo atributos geográficos. A criação de dimensões não agregadas é opcional e não possui relação com o processamento do cubo, sendo indiferente se é realizada antes ou depois desta atividade. Um exemplo de tabela candidata a se tornar uma dimensão não-agregável pode ser uma tabela que contenha dados de logradouros, a fim de servir de referencial para comparações como, por exemplo, distâncias.

3.2.4 Processamento do Cubo e Geração das Agregações

Detalhes de implementação da ferramenta PostGeoOlap, como o algoritmo de processamento do cubo e a criação das tabelas de agregação podem ser examinados com detalhes no trabalho de Colonese (2004), bem como descrições minuciosas das classes, atributos e casos de uso do modelo. Neste trabalho não serão propostas alterações no algoritmo de processamento do cubo.

3.3 EXTENSÕES AO MODELO DO POSTGEOOLAP

O trabalho que gerou a versão original do PostGeoOlap (COLONESE, 2004) teve como objetivos, de considerável amplitude, propor uma forma de modelagem conceitual que integre noções dimensionais e espaciais, formular estereótipos UML para representação de associações com hierarquias espaciais e a produção de uma ferramenta que permitisse a criação e processamento do cubo de dados e a geração das agregações para o aumento de desempenho. Ainda, a ferramenta produzida permitia submeter consultas ao *data warehouse*, obtendo os dados requisitados como resposta e, caso houvesse atributos geográficos, estes seriam plotados em um mapa de contexto fornecido pelo usuário na ocasião da definição do cubo.

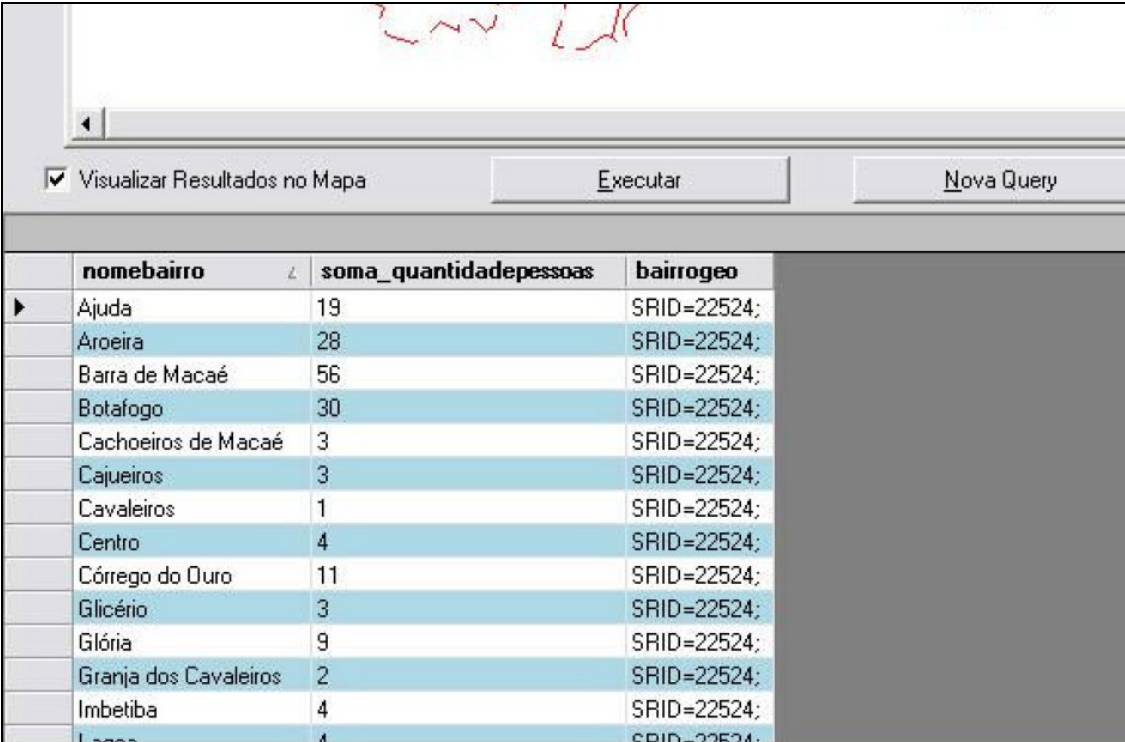
3.3.1 Tratamento de Dados Multidimensionais

Os dados retornados pelas consultas submetidas à ferramenta PostGeoOlap eram apresentados em formato tabular baseado em coluna, do mesmo modo que retornado por consultas SQL no banco de dados. Tal retorno não pode ser considerado dimensional, pois traz apenas dados distribuídos em colunas simples, sem qualquer indicação de dimensionalidade, conforme mostram a Figura 14 e a Figura 15.



Figura 14. Consulta no PostGeoOlap original. Fonte: Colonese (2004)

Na Figura 14, os atributos `soma_valorvendabruto` e `soma_valorlucrodist` pertencem à tabela fato `Venda`; o atributo `ano` pertence à dimensão `Tempo`; e, finalmente, os atributos `nrponto venda` e `ponto vendageo` pertencem à dimensão `Ponto venda`. Na Figura 15, os atributos `nomebairro` e `bairrogeo` pertencem à dimensão `setorcensitario` e o atributo `soma_quantidadepessoas` é uma sumarização feita a partir da tabela fato. Em ambos os casos, como os dados são mostrados do mesmo modo como foram recuperados do banco de dados relacional, não há indicação de dimensionalidade ou hierarquização. Do mesmo modo, operações OLAP como *drilling* ou *pivoting* ficam impossibilitadas de serem obtidas de modo interativo, obrigando o usuário a refazer uma consulta para conseguir uma análise através de outro nível hierárquico, o que torna impraticáveis certos tipos mais detalhados de análise. Outra limitação deste tipo de interação é que obriga que o usuário possua um razoável conhecimento da estrutura hierárquica e dimensional do cubo, já que esta não lhe aparece clara na visão tabular.



	nomebairro	soma_quantidadepessoas	bairrogeo
▶	Ajuda	19	SRID=22524;
	Ároeira	28	SRID=22524;
	Barra de Macaé	56	SRID=22524;
	Botafogo	30	SRID=22524;
	Cachoeiros de Macaé	3	SRID=22524;
	Cajueiros	3	SRID=22524;
	Cavaleiros	1	SRID=22524;
	Centro	4	SRID=22524;
	Córrego do Ouro	11	SRID=22524;
	Glicério	3	SRID=22524;
	Glória	9	SRID=22524;
	Granja dos Cavaleiros	2	SRID=22524;
	Imbetiba	4	SRID=22524;
	Imbetiba	4	SRID=22524;

Figura 15. Consulta no PostGeoOlap original. Fonte: Galante (2004)

Do exposto, decorre a necessidade de que o PostGeoOlap seja capaz de lidar com dados em um modelo dimensional, para que possa ser efetivamente uma ferramenta que realize operações OLAP de modo interativo.

Em sua versão original, o PostGeoOlap apenas tratava com a dimensionalidade no nível dos metadados, com a definição de dimensões, cubos e agregações. Aqui, tratar-se-á de extensões que tornem a ferramenta capaz de manipular dados dimensionais, identificando os dados obtidos do SGBD a dimensões do modelo.

O exposto acima significa que o modelo OLAP proposto no trabalho original sobre o PostGeoOlap se limitava a um modelo estático, orientado somente a metadados. Contudo, para os objetivos aqui propostos, é necessário incluir no modelo suporte ao funcionamento dinâmico da ferramenta, qual seja, a alocação dos dados obtidos do *data warehouse* em dimensões e atributos, em lugar de *record sets* baseados em linhas e colunas de dados.

Para tratar dos dados obtidos pela ferramenta sob o ponto de vista multidimensional, é necessário possuir meios de mapear cada dado obtido a um atributo de uma dimensão (fato ou não) do modelo. Além disto, é necessário traçar uma linha divisória entre dados provenientes de dimensões e dados oriundos de tabelas fato. Os dados dimensionais têm um tratamento mais simples no que concerne à representação OLAP, estando sempre associados em uma relação de igualdade com um atributo da dimensão. Os dados de medidas, por sua vez, são normalmente resultados de agrupamentos e estão associados a uma ou mais relações de igualdade entre um atributo de dimensão e seu dado correspondente. Por exemplo, se há uma dimensão *Vendedor* e um atributo *nome* cujo valor obtido do *data warehouse* é “Maria”, estabelece-se uma relação de igualdade *Vendedor.nome* = “Maria”, que vincula diretamente um dado a uma entidade do metamodelo, no caso, o atributo *nome* da dimensão *Vendedor*. Estendendo o exemplo, uma consulta ao *data warehouse* retornou R\$ 50.000,00 de vendas para Maria no mês de junho de 2006. Estabelecem-se, assim, duas outras igualdades: *Tempo.mês* = “Junho” e *Tempo.ano* = 2006. Contudo, o dado (medida) 50.000,00 não pode ser atribuído simplesmente a um atributo, mas, sim, está associado às três igualdades citadas, pois Maria vendeu R\$ 50.000,00 no mês de junho de 2006, não sendo possível conferir significado ao valor 50.000,00 sem implicar o vendedor de nome Maria, o mês de junho e o ano 2006 em sua definição.

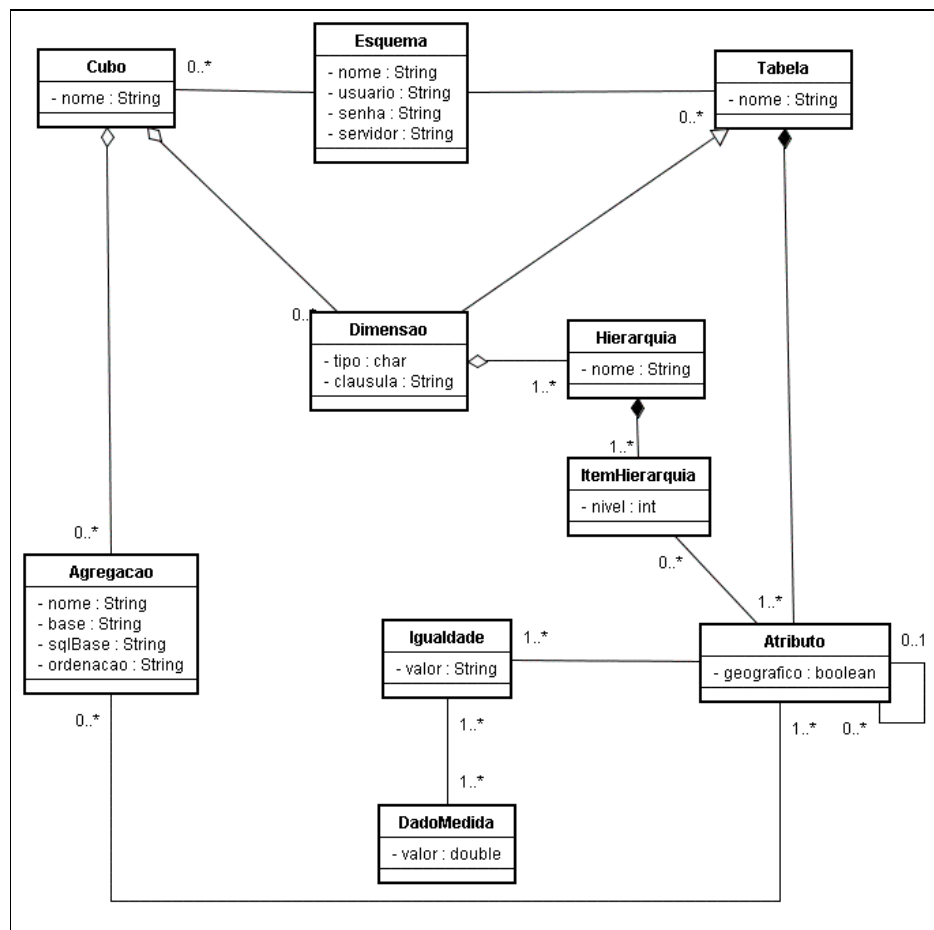


Figura 16. Extensão ao modelo do PostGeoOlap.

Assim, propõe-se que o modelo conceitual do PostGeoOlap seja estendido com as classes *Igualdade* e *Medida*, representando a relação entre um atributo de uma dimensão de dados e um valor dimensional obtido do *data warehouse* e a medida obtida da tabela fato. Esta medida é, então, associada a um conjunto de igualdades. O diagrama de classes resultante desta extensão pode ser visto na Figura 16.

É importante ressaltar que a abordagem desenvolvida neste tópico difere de propostas como a de Dittrich, Kossmann e Kreutz (2005), na medida em que estes propõem, em um só processo, transformar resultados de consultas SQL em interfaces OLAP. A presente abordagem segue uma linha diversa, adotada também por Maniatis *et al.* (2003), de dividir o processo de apresentação OLAP em modelos lógico e modelo de visualização. A extensão do PostGeoOlap proposta na presente seção consiste em um modelo lógico, sem qualquer preocupação com a apresentação, que será tratada em capítulo posterior.

3.3.2 Dimensões e Atributos

No metamodelo original do PostGeoOlap (COLONESE, 2004), existem as classes *Tabela*, *Dimensao* e *Atributo*, conforme mostrado no diagrama de classes da Figura 17. A classe *Tabela* representa as relações existentes no banco de dados (i.e., no esquema) sobre o qual se irá operar. A classe *Dimensao* representa os principais componentes de um cubo, ou seja, as dimensões de dados e dimensões de medidas ou fatos¹⁸. Em termos de operações, na classe *Tabela* são definidas aquelas responsáveis pela manipulação de tabelas no banco de dados, ou seja, pelos acessos aos metadados do SGBD. Na classe *Dimensao* – subclasse de *Tabela* – as operações preocupam-se quanto à representação de tabelas enquanto dimensões de um cubo, definindo os comportamentos necessários para tal.

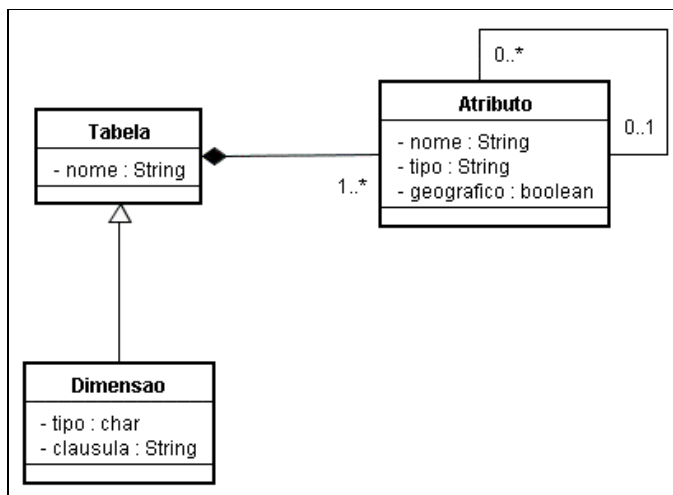


Figura 17. Relação entre tabelas, atributos e dimensões. Fonte: Colonese (2004)

A diferenciação entre tabela e dimensão, ou seja, entre representação física e representação lógica, confere uma importante separação de responsabilidades, além de tornar o modelo mais manutenível e coeso. Porém, esta mesma distinção não foi realizada com os atributos, que acumulam, na classe *Atributo*, comportamento responsável pela representação física de campos em uma tabela e pela representação lógica de atributos em uma dimensão. Isso significa que a classe *Atributo*, ao mesmo tempo, se refere aos campos como definidos

¹⁸ É importante notar que o modelo lógico do PostGeoOlap, conforme definido em Colonese (2004) e confirmado em trabalhos posteriores (COLONESE *et al.*, 2005; COLONESE *et al.*, 2006), trata dimensões de dados e dimensões de medidas (fatos) de modo uniforme, diferenciando-as apenas com um atributo. Em todo o resto, porém, são semelhantes.

metadados do banco de dados onde reside o *data warehouse* e aos atributos das dimensões definidas nos metadados do cubo OLAP.

Percebe-se, assim, que a classe `Atributo` acumula responsabilidades de um campo de tabela e de um atributo de dimensão, tornando-se necessário dividir a classe em duas: `Campo` e `Atributo`, para modelar separadamente as questões físicas e lógicas, aumentando a coesão no modelo.

Contudo, não seria a melhor solução criar uma relação de especialização entre `Campo` e `Atributo`, tal como ocorre no modelo original com `Tabela` e `Dimensão`, pois esta relação também deve ser revista. A relação de herança dita que toda dimensão seja uma tabela. Logo, uma tabela poderia “gerar” apenas uma única dimensão, na medida em que ela própria seria uma dimensão. Na realidade, porém, o que se deseja é que de uma mesma tabela se possam derivar várias dimensões. Por exemplo, a versão atual do PostGeoOlap permite apenas uma hierarquia por dimensão. Esta limitação pode ser contornada através da criação de mais de uma dimensão referente à mesma tabela, mas com uma hierarquia alternativa.

Para solucionar a limitação propiciada pela modelagem com o uso de herança, propõe-se aqui que o relacionamento entre `Tabela` e `Dimensao` seja uma associação um-para-muitos, de modo que uma tabela possa estar associada a várias dimensões. O mesmo se propõe para as classes `Campo` e `Atributo`. A Figura 18 mostra um diagrama de classes com a modelagem proposta.

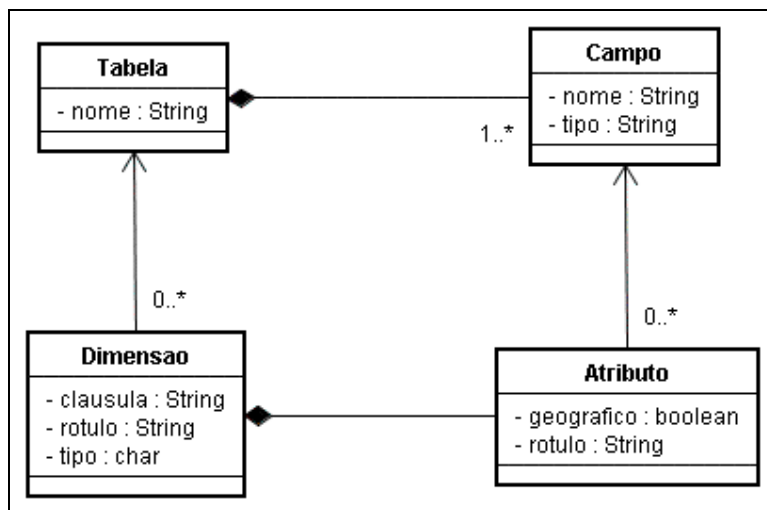


Figura 18. Modelo proposto para a relação entre dimensões e atributos.

Outro aprimoramento incluído na modelagem proposta é constituído pelos atributos `rotulo` em `Dimensao` e em `Atributo`. Na versão original, as dimensões e atributos eram

apresentados aos usuários com o mesmo nome das tabelas e campos o que, além de ser prejudicial à análise por parte do usuário, inviabilizaria uma tentativa de se ter mais de uma dimensão por tabela, pois ambas seriam apresentadas com o mesmo nome. Na presente proposta, o atributo `rotulo` tem como meta armazenar um nome amigável ao usuário para as dimensões e atributos, facilitando o entendimento e permitindo que o usuário estenda a semântica de sua análise.

3.3.3 Mapas

O modelo lógico do PostGeoOlap original possui uma classe denominada `Esquema`, responsável por representar o banco de dados (fisicamente, um *database* do PostgreSQL) no qual está contido o *data warehouse* com o qual se deseja operar. Dois atributos da classe `Esquema` em sua definição original são `mapa`, uma *string* contendo a localização de um mapa temático para o qual há dados geo-referenciados no *data warehouse*, e `srid`, que significa o “identificador do sistema de referenciamento geográfico para a região do mapa em questão” (COLONESE, 2004). Com o mapa na qualidade de atributo, tem-se que somente haverá um mapa por esquema. Há ocasiões, porém, em que pode ser desejável ter mais de um mapa em um mesmo esquema. Deste modo, propõe-se uma extensão no modelo do PostGeoOlap, mostrada na Figura 19, com a definição da classe `Mapa`, para o suporte a múltiplos mapas por esquema. Ademais, a inclusão do mapa na classe `Esquema` trazia consigo problemas de coesão, pois um mapa não é parte ou atributo de um esquema, estando, sim, associado a ele.

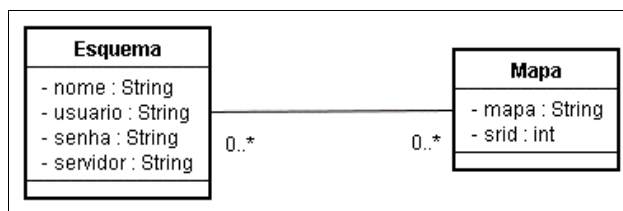


Figura 19. Associação entre mapas e esquemas.

3.3.4 Atributo Principal por Nível Hierárquico

Tipicamente, uma hierarquia pode ser caracterizada como uma seqüência ordenada de atributos para uma dada dimensão. Para exemplificar, pode-se imaginar uma dimensão *Tempo* contendo uma hierarquia com os atributos *ano*, *semestre*, *trimestre*, *mês* e *dia*, em ordem decrescente de agregação. Para adotar a terminologia estabelecida por Colonese (2004), pode-se dizer que o atributo *ano* está no nível hierárquico 9, *semestre* no nível 8, *trimestre* 7, *mês* 6 e, finalmente, o atributo *dia* no nível 5. Porém, no diagrama de classes do *core* do PostGeoOlap, conforme a Figura 20, vê-se que um *ItemHierarquia* pode estar associado a um ou muitos objetos da classe *Atributo*. Deste modo, tem-se que um mesmo item hierárquico pode estar associado a vários atributos, o que é natural, pois no mesmo nível hierárquico que *mês*, podemos ter, por exemplo, o nome e o número do mês.

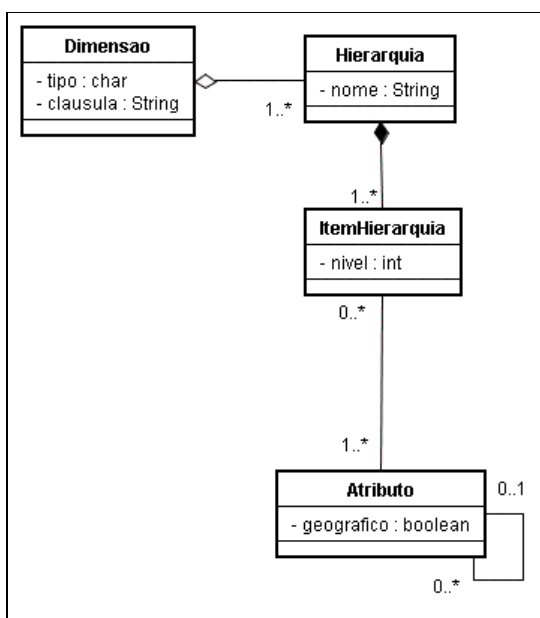


Figura 20. Associação um-para-muitos entre *ItemHierarquia* e *Atributo*.

Assim, apresenta-se a necessidade de que se estabeleça para cada nível hierárquico o atributo que se apresentará como fio condutor para a análise do usuário, com os outros atributos do nível sendo auxiliares a este. Deste modo, optou-se por criar uma nova associação de *ItemHierarquia* com *Atributo* para expressar a associação com o atributo preferencial, conforme a Figura 21.

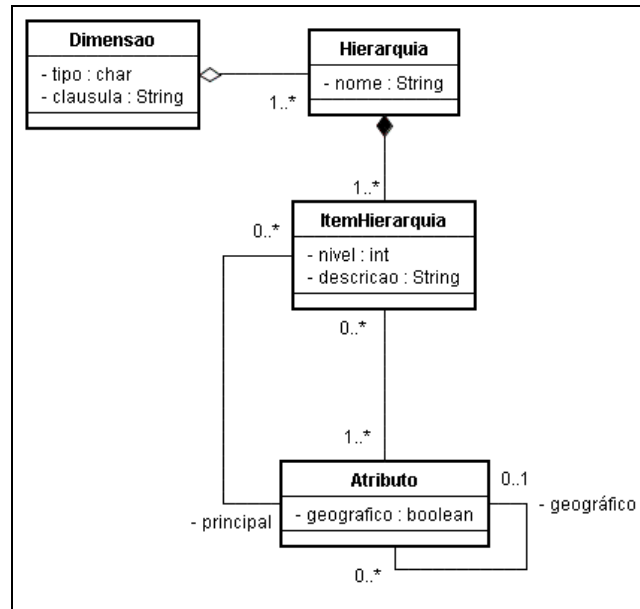


Figura 21. ItemHierarquia associado a um Atributo principal e a vários auxiliares.

3.3.5 Resultado das extensões

O resultado das extensões propostas ao núcleo lógico do PostGeoOlap é mostrado no diagrama de classes da Figura 22, onde é possível examinar as extensões propostas nos tópicos anteriores em um todo coerente.

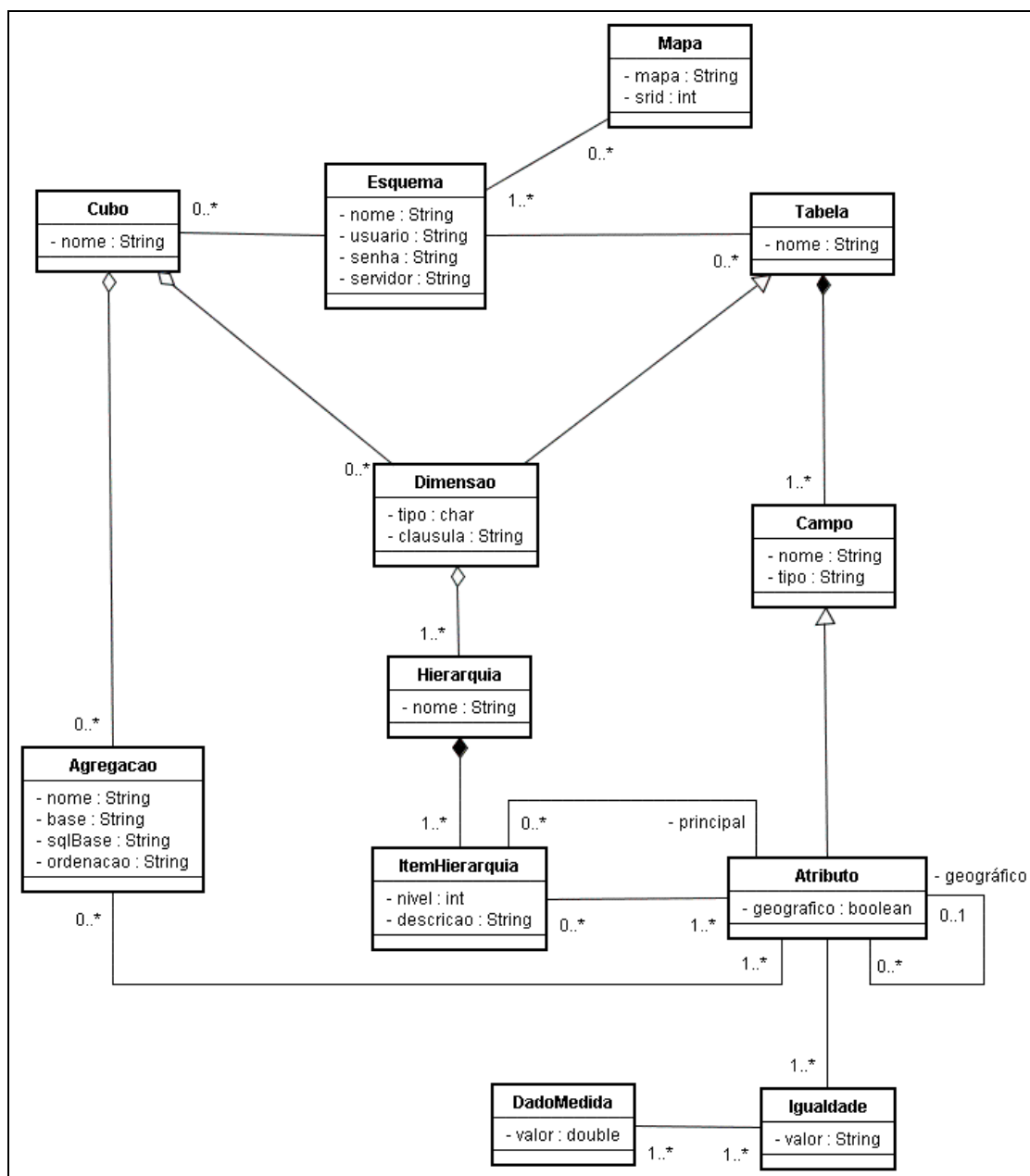


Figura 22. Resultado das extensões ao PostGeoOlap.

3.4 MIGRAÇÃO PARA A PLATAFORMA JAVA

O PostGeoOLAP foi escrito originalmente (COLONESE, 2004) na linguagem VisualBasic.NET¹⁹. Como esta ferramenta é distribuída sob licenças de caráter proprietário e seu uso é restrito a sistemas operacionais da família Windows 32 *bits*, optou-se neste trabalho, pela construção de uma nova versão escrita na plataforma Java²⁰. Esta plataforma, além de ser amplamente utilizada e aceita na comunidade de Software Livre, oferece muito mais recursos, é mais estável e robusta e possui uma linguagem muito mais rica, além de ser realmente orientada a objetos. Isto permite futuras incorporações de uma miríade de APIs, *frameworks* e ferramentas livres (como já é o caso do OpenJUMP). Além disto, o uso do Java torna PostGeoOlap, efetivamente, uma ferramenta multiplataforma, já que há máquinas virtuais Java disponíveis para a maioria dos sistemas operacionais. Outra vantagem é a possibilidade de uso das mesmas classes e modelos tanto para aplicações baseadas em interfaces *web* quanto em interfaces *desktop*, alterando-se apenas a camada de apresentação e deixando todo o resto da aplicação intocada, o que garante extrema flexibilidade para o crescimento da ferramenta.

Assim, foi necessária a reescrita completa do código original, valendo destacar alguns tópicos relevantes. A versão do PostGeoOlap migrada para Java encontra-se disponível para *download* em <http://postgeoolap.projects.postgresql.org>.

3.4.1 Java e Desempenho

Há um mito bastante difundido de que a plataforma Java, devido à obrigatoriedade do uso de uma máquina virtual e a existência do tão discutido “coletor de lixo” (*garbage collector*), teria um desempenho bastante baixo em relação a plataformas e linguagens que se valem de compilação estática e desalocação manual de memória. Além disto, há o argumento de que o Java seria interpretado, o que faria que seu desempenho caísse ainda mais.

Tal linha de raciocínio, porém, carece de atualidade. Na década de 1990, quando ainda estava longe da maturidade que hoje exhibe, Java era realmente uma plataforma pesada e de

¹⁹ <http://msdn.microsoft.com/vbasic/>, acesso em 14/03/2007.

²⁰ <http://java.sun.com>, acesso em 14/03/2007.

baixo desempenho, se comparada a linguagens como C++. Porém, após a introdução da chamada compilação *Just In Time* (JIT) nas JVMs (*Java Virtual Machine*), já se podia prever que o desempenho do Java suplantaria em um futuro não distante o desempenho do C++ (REINHOLTZ, 2000), linguagem tida como de alto desempenho. A compilação JIT, introduzida na plataforma Java com o lançamento pela Sun da JVM (*Java Virtual Machine*) HotSpot em 1999, torna possível que a máquina virtual detecte “pontos quentes” (ou *hot spots*) de código que são freqüentemente ou repetidamente executados. Estes trechos de código passam, então, por processos de otimização e compilação, levando a um aumento geral de desempenho de uma aplicação. Além disto, aplicações que utilizam compiladores JIT tendem a superar em desempenho aquelas que utilizam compilação estática, pois os primeiros dispõem de informações de tempo de execução que permitem otimizações bastante agressivas, algo vedado, por sua própria natureza, a processos estáticos de compilação.

A situação atual, com isso, se modificou bastante. Hoje, pode-se constatar que, em vários *benchmarkings*, especialmente no que tange à computação numérica (LEWIS e NEUMANN, 2004), que a plataforma Java tem desempenho bem semelhante ao C++, e até mesmo superior em vários casos. A isto se some o fato de que o C++ é uma linguagem cujo ritmo evolutivo atual é bastante lento; o Java, por outro lado, evolui a passos largos. A versão 6 do JSE (*Java Standard Edition*) inclui técnicas avançadas de otimização como análise de escape²¹ e *lock-coarsening*²², além da contínua evolução de técnicas de *profiling*²³ (DOEDERLEIN, 2005b).

²¹ A análise de escape é uma técnica na qual o compilador determina quais variáveis locais em um método são usadas única e exclusivamente dentro do próprio método e quais podem eventualmente “escapar” ao método (através de um retorno ou passagem como argumento a outro método, por exemplo). Caso uma variável local não escape ao método, pode ser alocada no *stack* privado do *thread* que está executando o método, em lugar de ser alocada no *heap*. O custo de uma alocação no *heap* é mais alto, pois aumenta o trabalho do *garbage collector*, responsável por varrê-lo em busca de referências que não são mais utilizadas. Um dos trabalhos pioneiros sobre análise de escape na plataforma Java é o artigo de Choi *et al.* (1999).

²² *Lock-coarsening* é uma otimização que permite que blocos adjacentes de exclusão mútua que utilizem o mesmo objeto como *lock* tenham automaticamente eliminados os pares intermediários de operações *unlock/lock*, diminuindo o *overhead*. O trabalho de Diniz e Rinard (1996) oferece maiores detalhes sobre esta técnica.

²³ *Profiling* é uma técnica que busca analisar o desempenho de um programa enquanto está rodando. O resultado desta análise é um fluxo de eventos registrados (chamado *trace*), ou alguma informação sumarizada sobre estes eventos (chamada *profile*). As ferramentas que realizam este tipo de análise de desempenho chamam-se *profilers*. As JVMs modernas possuem *profilers* embutidos.

3.4.2 Java e Software Livre

Há até bem recentemente, boa parte da comunidade de Software Livre, especialmente alas mais apegadas à questão filosófico-ideológica do *open-source* (STALLMAN, 2004), questionavam o uso da plataforma Java em projetos cujo caráter seja de liberdade irrestrita de uso, modificação e distribuição. Isto se devia ao fato de que, a despeito das especificações da plataforma Java serem abertas, ainda não havia nenhuma máquina virtual Java livre e certificada. Este problema foi superado no fim de 2006, com a liberação, sob a licença GPL, da JVM da Sun.

Todas as tecnologias Java, e não só a máquina virtual, são baseadas em especificações criadas pelo Java Community Process (JCP)²⁴, cujo “processo de padronização exige que as especificações resultantes possam ser implementadas sem pagamento de royalties, e portanto que possam ser criadas como software livre – além de certificadas como compatíveis” (SOUZA, 2004). Além disto, tais especificações não são criadas somente pela Sun. A participação no JCP é aberta a instituições e indivíduos (estes últimos, gratuitamente, e com direito a voto). Como exemplos de instituições participantes podem ser citadas Adobe, America Online, Apache Software Foundation, AT&T, Borland, Cisco Systems, Creative Labs, Ericsson AB, Fujitsu, Google, Hewlett-Packard, IBM, JBoss Group, Macromedia, Motorola, MySQL, Nokia, Oracle, Sega, Samsung, além de contar com um grande número de indivíduos, inclusive brasileiros, colocando o “JCP lado a lado com o W3C (World Wide Web Consortium)²⁵ como duas das grandes organizações de padronização que suportam implementações open source das especificações” (*idem*), ficando, neste quesito, bem à frente de organizações como ISO²⁶ (International Organization for Standardization) ou Ecma International²⁷, que não dão garantias de que suas especificações sejam abertas (*idem*).

Além da máquina virtual, há também compiladores Java em Software Livre, como o distribuído junto com a IDE Eclipse²⁸, que, de acordo com Doederlein (2005a) é melhor e mais eficiente – e claramente muito mais rápido – que a implementação oficial do JDK (Java

²⁴ <http://www.jcp.org>, acesso em 15/03/2007.

²⁵ <http://www.w3.org>, acesso em 15/03/2007.

²⁶ <http://www.iso.org>, acesso em 15/03/2007.

²⁷ <http://www.ecma-international.org>, acesso em 15/03/2007.

²⁸ <http://www.eclipse.org>, acesso em 15/03/2007.

Development Kit) da Sun, a ponto de constituir, por si só, motivo suficiente para o uso do Eclipse como IDE para desenvolvimento Java.

Portanto, não há mais, nos dias de hoje, qualquer incompatibilidade entre a plataforma Java e o Software Livre.

3.4.3 Visualização de Mapas

Na versão original do PostGeoOlap, o componente utilizado para a visualização dos mapas e plotagem dos atributos geográficos advinha do SIG PlanetGIS²⁹, na forma de um componente OCX, possuindo compatibilidade apenas com os sistemas operacionais da linha Windows, tendo, deste modo, seu uso bastante dificultado em ambientes multiplataforma. Além disto, apesar de poder ser adquirido gratuitamente na Internet, o citado componente não está disponível sob uma licença livre nem tem seu código disponibilizado.

Assim, para a adequação aos princípios de projeto do PostGeoOlap foi necessário encontrar facilidades para plotagem de mapas e objetos geográficos que, além de ser compatíveis com Java, sejam distribuídas sob licenças livres. O projeto OpenJUMP foi, assim, escolhido devido à sua adequação aos princípios de projeto do PostGeoOlap. Além disto, o OpenJUMP é tido como um dos melhores SIG distribuídos como Software Livre existentes.

A Figura 23 mostra o OpenJUMP em ação na ferramenta PostGeoOlap, em sua versão escrita em Java, livre e multiplataforma.

²⁹ <http://www.planetgis.co.za>, acesso em 15/03/2007.

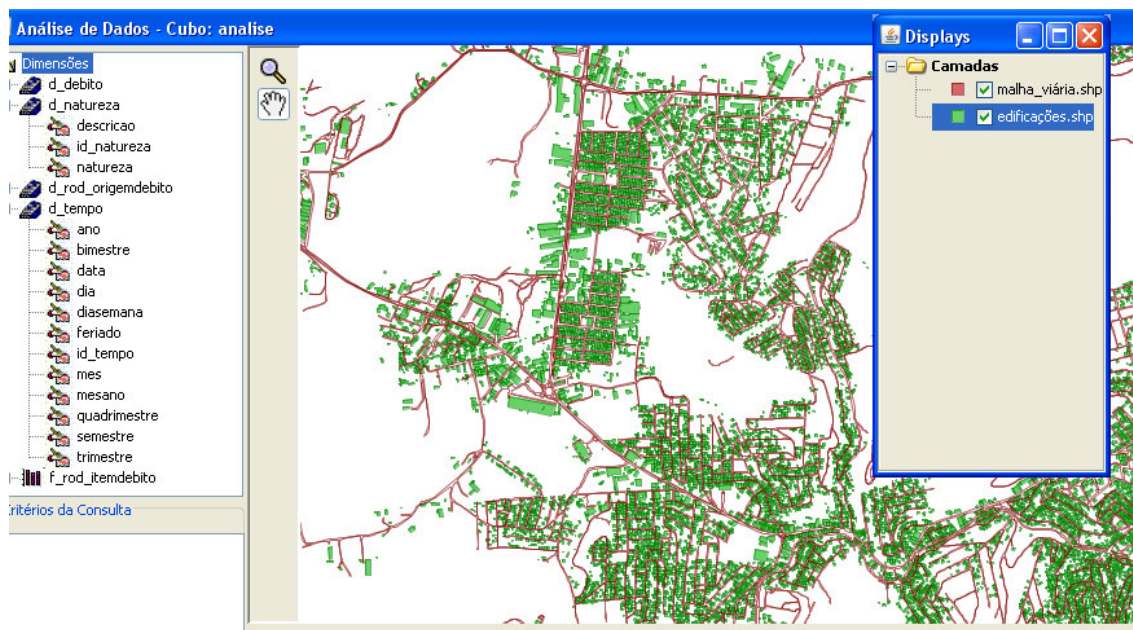


Figura 23. Visualizador do OpenJUMP incorporado ao PostGeoOlap.

O visualizador do JUMP oferece recursos de *zooming* e possibilita que se arraste o mapa, de modo a trazer à visualização partes ocultas do mapa. Há também um componente para o trato das camadas. No exemplo acima, há dois mapas no esquema sendo utilizado, um referente à malha viária da cidade e outro referente às edificações. Cada um dos dois mapas é tratado como uma camada de visualização, de modo que a janela “Displays” possibilita ocultar camadas de modo que possam ser visualizadas independentemente. Assim, pode-se, apenas clicando na caixa de verificação na janela “Displays”, optar pela visualização apenas da malha viária, apenas das edificações ou de ambas. Ao se fazer uma consulta que retorne dados geográficos, as formas geométricas plotadas também aparecem como uma camada extra e com a possibilidade de terem, do mesmo modo, sua visualização ou ocultação manipuladas.

Além destes recursos, o JUMP é altamente extensível, possuindo uma arquitetura orientada a *plugins* que permite que se acrescentem comportamentos materializados como botões para acionamento por parte do usuário.

3.4.4 O Fator Geográfico

A porção geográfica da ferramenta PostGeoOlap não sofreu alterações substanciais, na medida em que não se buscou propor, neste trabalho, a adição de medidas geográficas, assim como o trabalho que originalmente definira o PostGeoOlap (COLONESE, 2004) e trabalhos posteriores (COLONESE *et al.*, 2005; COLONESE *et al.*, 2006) também não propuseram a existência de atributos geométricos nas tabelas-fato. De fato, Han, Stefanovic e Koperski (1998), afirmam que “se um cubo espacial contém dimensões espaciais mas não medidas espaciais, suas operações OLAP, como *drilling* ou *pivoting*, podem ser implementadas de uma maneira similar a cubos de dados não espaciais” (tradução do autor).

Uma alteração substancial no trato de operações OLAP em *data warehouses* espaciais seria gerada apenas pela inclusão de medidas geográficas, o que não é realizado neste trabalho, apontando tais evoluções como proposta para trabalhos futuros.

3.4.5 Interface Gráfica

O PostGeoOlap, do ponto de vista da tecnologia de interface com o usuário, é uma aplicação desktop. Em Java, o mais popular e poderoso framework gráfico é o Swing (ROBINSON e VOROBIEV, 2003), de modo que a adoção deste ao projeto foi uma escolha natural. Porém, o grande poder proporcionado pelo Swing tem um alto preço no que diz respeito à sua complexidade, sendo criticado por ser exageradamente acadêmico e cumprir de modo extremo todos os requisitos de um projeto estritamente orientado a objetos (MARINACCI, 2003). No entanto, consideramos isto uma qualidade, na medida em que fornece ao programador a possibilidade de construir virtualmente qualquer coisa na interface com o usuário, sem que para isto tenha de recorrer a qualquer tipo de artifício que fuja dos consagrados paradigmas da orientação a objetos e padrões de projeto, sendo, por isto, bastante adequado para uso em ambientes acadêmicos. Além disto, o Swing estimula boas práticas de projeto orientado a objetos, como o uso extensivo de padrões arquiteturais como Model-View-Controller (MVC) (BUSCHMANN *et al.*, 1996) e padrões de projeto como Observer (GAMMA *et al.*, 1995) – no jargão do Swing são chamados *listeners* –, que adicionam

flexibilidade ao projeto da interface com o usuário, na medida em que permitem uma completa separação de responsabilidades entre quem detém a informação e quem a apresenta.

No que diz respeito à codificação da interface gráfica, optou-se por utilizar diretamente os layouts do AWT (Abstract Windowing Toolkit), preterindo soluções visuais no estilo arrastar-e-soltar, na medida em que estes recursos proprietários³⁰ necessariamente amarram a edição do código à dependência de IDEs específicas.

Para a construção das interfaces gráficas da ferramenta, utilizou-se, basicamente, o gerenciador de *layout* `GridBagLayout` com codificação manual.

3.4.6 Organização do Projeto

Como a ferramenta tem se tornado maior e mais complexa, optou-se por, aproveitando a reimplementação, dividi-la em subprojetos, propiciando um melhor gerenciamento. Assim, o PostGeoOlap dividiu-se em cinco subprojetos:

- Core: Núcleo lógico da ferramenta, realizando definição e processamento de cubos, recebendo consultas e devolvendo dados na forma de objetos em uma forma multidimensional. O processamento dos mapas e dados geo-referenciados também é realizado pelas classes deste subprojeto.
- AdminGUI: Conjunto de interfaces gráficas para a administração dos metadados para o *data warehouse*, ou seja, definição de esquemas, cubos, dimensões e atributos.
- PresentationModel: Implementação da camada de apresentação, baseada na extensão proposta neste trabalho ao Cube Presentation Model.
- SwingGeoOlap: Implementação da interface de análise OLAP para ambiente *desktop* multiplataforma utilizando o *toolkit* Swing.
- WebGeoOlap: Implementação da interface de análise OLAP para ambiente *web*.

³⁰ A palavra proprietários se refere à edição visual dos formulários. O código gerado pelas principais alternativas livres, o Visual Editor (<http://www.eclipse.org/vep>, acesso em 09/03/2007) do Eclipse e o Matisse (<http://www.netbeans.org/kb/articles/matisse.html>, acesso em 09/12/2006) do NetBeans pode ser rodado em qualquer computador, de modo independente de qualquer IDE. Ao contrário do código, porém, a manutenção visual de uma GUI só pode ser realizada na IDE de origem.

De todos os subprojetos, o único que não foi implementado como subproduto do presente trabalho é o WebGeoOlap.

3.4.7 Arquitetura e Dependências

Com o processo de migração, o PostGeoOlap não sofreu alterações arquiteturais de grande impacto, tendo sido mantida a arquitetura original básica.

O PostGeoOlap, na versão original, utilizava uma ferramenta de visualização de dados proprietária chamada PlanetGIS. Na versão atual, foi substituída por classes do *framework* OpenJUMP.

O SGBD PostgreSQL é utilizado pelo PostGeoOlap para a realização das funções convencionais de agregação padrão SQL (e.g. *sum*, *max*, *min*, *avg*, *count*) e todas as funções geográficas definidas pelo Open Geospatial Consortium, nas especificações OpenGIS (OPENGIS, 1999).

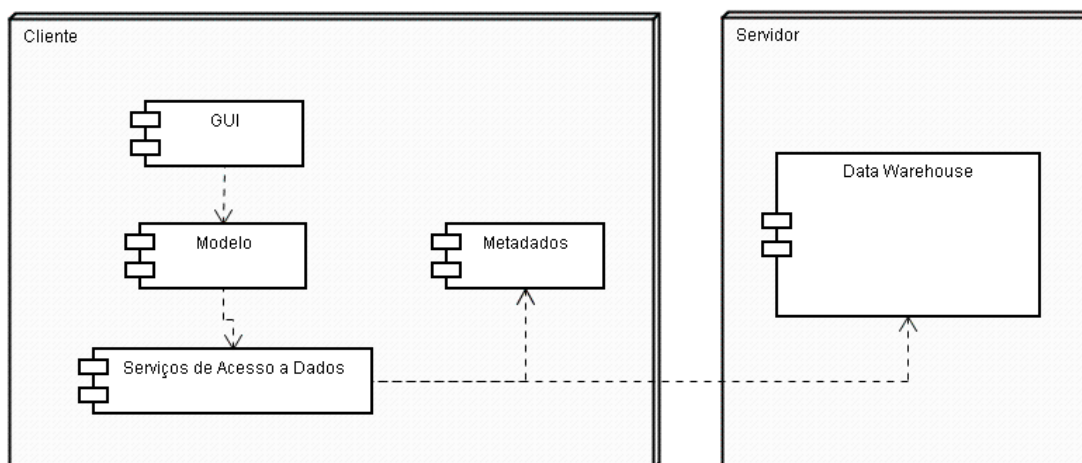


Figura 24. Arquitetura do PostGeoOlap

O *deployment diagram*³¹, na Figura 24, mostra a arquitetura da extensão proposta nesta dissertação ao modelo do PostGeoOlap. Optou-se, aqui, por manter os metadados no cliente, tal como concebido originalmente. O motivo é a possibilidade de que o cliente possa interagir com

³¹ Como não há uma tradução consagrada do *deployment diagram* da UML na literatura em língua pátria, optou-se pela utilização do nome original. Os tradutores de Fowler (2005), por exemplo, chegam a utilizar, na mesma página (102), dois termos diferentes – “diagrama de distribuição” e “diagrama de implantação” – para referir-se ao *deployment diagram*. Em outra tradução (LARMAN, 2004), ainda, o termo utilizado é “diagrama de distribuição física”.

os metadados mesmo se o servidor estiver indisponível. Na implementação original, utilizou-se para os metadados locais o Microsoft Access. Após a migração, o SGBD livre puro-Java Apache Derby³² foi escolhido para esta tarefa. Sua capacidade de rodar em modo embarcado (*embedded*) dispensa instalações, *drivers* e configurações, rodando onde quer que se possa executar o PostGeoOlap.

Além do Apache Derby, o PostGeoOlap utiliza outros *softwares* livres para auxiliar em suas tarefas como: Apache Log4J³³ e Apache Jakarta Commons Logging³⁴ para *logging*; Hibernate para mapeamento objeto/relacional entre as classes de metadados e seus respectivos dados no SGBD; e JUnit³⁵ para testes de unidade automatizados.

A versão atual do PostGeoOlap conta, também, com suporte à internacionalização baseada em Unicode, possuindo versões em português e inglês, selecionadas automaticamente a partir do idioma padrão do sistema operacional. Traduções para outros idiomas podem ser realizadas através da simples edição de um arquivo de propriedades em formato texto puro.

³² <http://db.apache.org/derby>, acesso em 19/03/2007.

³³ <http://logging.apache.org/log4j>, acesso em 19/03/2007.

³⁴ <http://jakarta.apache.org/commons/logging>, acesso em 19/03/2007.

³⁵ <http://www.junit.org>, acesso em 19/03/2007.

4 .PROPOSTA DE VISUALIZAÇÃO MULTIDIMENSIONAL

Este capítulo tem por objetivo expor a proposta de visualização multidimensional baseada no Cube Presentation Model e um modelo arquitetural para a implementação desta visualização como uma camada que funcione de modo desacoplado tanto do modelo lógico quanto do *toolkit* gráfico utilizado para a apresentação.

4.1 EXTENSÕES AO CUBE PRESENTATION MODEL

4.1.1 Definições

A camada de apresentação do Cube Presentation Model, cujo projeto e implementação são descritos nesta dissertação, é uma proposta funcional, na medida em que logra criar uma arquitetura de *software* que permite operar com dados dimensionais para visualização, independentemente de considerações tecnológicas (nem mesmo a limitação de uma tela bidimensional). Cabe destacar o uso de metadados, os *schemata* de cubos e eixos, como forte elemento de flexibilização da arquitetura. A despeito de suas muitas e evidentes qualidades, o modelo possui pontos onde cabem extensões e aprimoramentos.

Inicialmente, é necessário definir certos termos doravante utilizados neste trabalho. Chamar-se-á *instância de um nível hierárquico* cada item de dados individual de um certo nível hierárquico. Por exemplo, na Figura 26 o mais alto nível hierárquico da dimensão *Tempo*, representado pelo atributo *ano*³⁶, possui as instâncias “2003”, “2004” e “2005”. Analogamente, o mais alto (e único) nível hierárquico da dimensão *Vendedor*, representado pelo atributo *Nome*, possui as instâncias “José” e “Maria”. Chamaremos, ainda, *grupo de instâncias* cada repetição do mesmo conjunto de dados dimensionais dentro de uma tabela visual OLAP. Por exemplo, na Figura 33, há dois grupos encabeçados pela instância “USA”: um está sob a instância “Venk” e outro sob a instância “Netz”. Ou seja, haverá potencialmente um grupo encabeçado por “USA” e outro para “Japan” para cada instância da dimensão *Vendedor*. Isto se dá porque a dimensão *Local* está agregada pela dimensão *Vendedor*.

Outra importante definição terminológica é a de nível hierárquico *aberto* ou *fechado* em uma instância de um dado grupo. Diz-se que um nível está aberto em uma instância de um grupo se esta é a mais agregada no ponto CPM a que pertence. De outro modo, ela é fechada. Ou seja, uma dada instância estará fechada se estiver no nível hierárquico mais baixo e for passível de *drill-down* e aberta se sobre esta já tiver sido aplicado um *drill-down*. Por exemplo, na Figura 28 a instância “2003” do atributo *ano* da dimensão *Tempo* está aberta, enquanto as instâncias “2004” e “2005” estão fechadas.

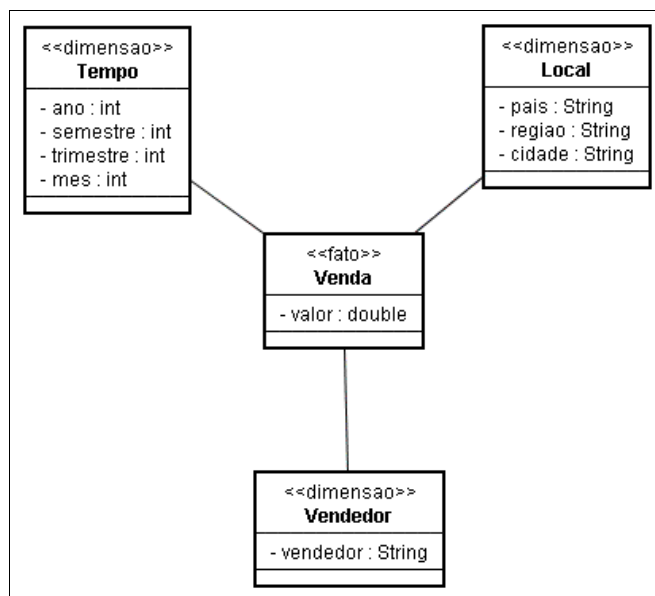


Figura 25. *Data warehouse* de exemplo.

³⁶ No decorrer do texto, optou-se pela convenção de denominar dimensões com inicial maiúscula e atributos em caixa baixa.

A Figura 25 mostra o modelo conceitual do *data warehouse* utilizado duante este capítulo como exemplo.

4.1.2 Interação OLAP flexível

Uma característica indispensável de uma interface OLAP é a possibilidade de se realizar operações de *drilling* sobre instâncias de níveis hierárquicos de dimensões. Uma operação de *drilling* pode ser aplicada sobre uma dimensão inteira, obrigando que todas as suas instâncias alterem seu nível hierárquico.

Na Figura 26, por exemplo, há uma dimensão *Tempo* e uma dimensão *Vendedor*. A primeira é mostrada com o atributo *ano* e a segunda com o atributo *vendedor*. Cada um dos itens numéricos representa a quantidade de vendas efetuadas por um determinado vendedor em um determinado ano. No exemplo da Figura 26, José efetuou 30 vendas no ano de 2003.

	José	Maria
2003	30	50
2004	70	25
2005	110	120

Figura 26. Dimensão *Tempo* sendo visualizada por *ano*.

Um eventual usuário pode requisitar a visualização das vendas em uma menor granularidade temporal. Assim, se o analista decidir que necessita de uma visão mais detalhada das vendas em relação ao tempo, ele realiza um *drill-down* na dimensão *Tempo*, passando a visualizar as vendas por semestre, conforme a Figura 27.

		José	Maria
2003	1	10	20
	2	20	30
2004	1	30	15
	2	40	10
2005	1	50	60
	2	60	60

Figura 27. Dimensão *Tempo* sendo visualizada por *semestre*.

Um mecanismo flexível de visualização OLAP deve fornecer meios para que o usuário realize *drill-down* sobre apenas uma instância de um nível hierárquico, alterando somente esta e mantendo o estado do restante da dimensão. No exemplo do parágrafo anterior, o analista optou

por um *drill-down* sobre toda a dimensão. Porém, o usuário pode desejar visualizar um nível menos agregado para apenas uma instância, como no exemplo da Figura 28.

		José	Maria
2003	1	10	20
	2	20	30
2004		70	25
2005		110	120

Figura 28. A instância "2003" da dimensão *Tempo* sendo visualizada por semestre.

Do mesmo modo, uma interface OLAP flexível deve permitir que se adicionem dimensões aos eixos visuais. Por exemplo, é possível adicionar uma dimensão *Local* ao eixo horizontal, de modo que fique hierarquicamente subordinada à dimensão *Vendedor*, conforme a Figura 29. A dimensão *Local* aparece apenas com seu nível hierárquico mais agregado, país.

		José		Maria	
		Brasil	Japão	Brasil	Japão
2003	1	5	5	10	10
	2	12	8	18	12
2004		55	15	15	10
2005		30	80	50	70

Figura 29. Interface OLAP acrescida da dimensão *Local*.

Para o desenvolvimento da discussão, assumir-se-á que a dimensão *Local* possui uma hierarquia de três níveis: país, região e cidade. Analogamente ao mostrado em figuras *supra*, uma interface OLAP flexível deve permitir um *drill-down* do modo como mostra a Figura 30, ou seja, apenas para uma instância em especial (no caso, o nível “Japão” do ponto dado por [“Maria”, “Japão”]).

		José		Maria		
		Brasil	Japão	Brasil	Japão	
					Kanto	Kansai
2003	1	5	5	10	5	5
	2	12	8	18	8	4
2004		55	15	15	2	8
2005		30	80	50	33	37

Figura 30. Dimensão *Local* após *drill-down* em uma das instâncias “Japão”.

Do mesmo modo, outras instâncias também poderiam sofrer *drill-down* repetidas vezes, formando configurações visuais complexas e de flexível manipulação, conforme mostra a Figura 31.

		José				Maria		
		Brasil			Japão	Japão		
		São Paulo	Rio de Janeiro			Brasil	Kanto	Kansai
			Campos	Macaé				
2003	1	2	2	1	5	10	5	5
	2	10	1	1	8	18	8	4
2004		30	10	15	15	15	2	8
2005		10	10	10	80	50	33	37

Figura 31. Interface OLAP com interações complexas.

Além disto, o analista deve ter a opção de remover dimensões dos eixos visuais, como, por exemplo, a dimensão *Vendedor*, reduzindo a análise a *Tempo* e *Local*, conforme a Figura 32.

		Brasil			Japão	
		São Paulo	Rio de Janeiro		Kanto	Kansai
			Campos	Macaé		
2003	1	8	4	3	8	7
	2	18	7	5	12	8
2004		35	15	20	12	13
2005		30	25	25	63	87

Figura 32. Interface OLAP após remoção da dimensão *Vendedor*.

Resumindo, em uma interface OLAP flexível não pode haver nenhum impedimento para o usuário obter qualquer configuração que desejar. O único limite é a hierarquia de níveis dimensionais definidos para o cubo. Em termos de interação, é preciso dotar o usuário de completo poder de manipulação sobre os dados a ser analisados, sendo tal característica fundamental para que uma ferramenta de análise OLAP possa melhor apoiar a análise para a tomada de decisão.

4.1.3 Limitações de Flexibilidade no CPM

Maniatis *et al.* (2003) utilizam, durante toda a exposição da camada de apresentação do CPM, a grade OLAP mostrada na Figura 33. Ocorre que tal exemplo é demasiado inflexível e suscita maior atenção e análise do que lhe é dado no artigo citado. Basicamente, a figura consiste em uma grade bidimensional que emula uma interface OLAP contendo em seu eixo vertical a dimensão *Tempo* (denominada *Quarter* – ou seja, trimestre – no original, mas chamada aqui de *Tempo*) com dois níveis hierárquicos – *trimestre* e *mês* – sendo que o nível

do mês apenas está visível para as instâncias de trimestre “Qtr1” e “Qtr4”. Em seu eixo horizontal encontram-se as dimensões *Vendedor* e *Local* (Geography no original). A primeira possui um único nível hierárquico, representado pelo atributo *vendedor*, e a segunda apresenta três níveis hierárquicos, representados pelos atributos *país*, *região* e *cidade*. As indicações “R1” a “R4” e “C1” a “C6” se referem a marcações didáticas indicativas das *tapes*.

Year = 1991			Venk		Netz					
Product = ALL			USA		Japan		USA		Japan	
			USA_N		USA_S		USA_N		USA_S	
			Seattle	Boston			Seattle	Boston		
		Size(city)								
R1	Qtr1	Jan								
		Feb	C1		C2	C3		C4	C5	C6
		Mar								
R2	Qtr2									
R3	Qtr3									
R4	Qtr4	Jan								
		Feb								
		Mar								

Figura 33. Exemplo utilizado para demonstrar o CPM. Fonte: Maniatis *et al.* (2003)

A representação gráfica utilizada no exemplo mostrado é confusa e dificulta o entendimento à primeira vista. Doravante, as referências ao exemplo utilizado o artigo serão direcionadas à sua réplica na Figura 34. Esta é esquematicamente mais simples, possibilitando a compreensão direta e rápida do modelo. Ademais, optou-se por alterar os meses subordinados à instância “Qtr4”, pois se entende que o quarto trimestre de um determinado ano abrange os meses de outubro a dezembro, e não janeiro a março.

		Venk				Netz			
		USA			Japan	USA			Japan
		USA_N		USA_S		USA_N		USA_S	
		Boston	Seattle			Boston	Seattle		
Qtr1	Jan								
	Feb								
	Mar								
Qtr2									
Qtr3									
Qtr4	Oct								
	Nov								
	Dec								

Figura 34. Réplica do exemplo padrão do CPM.

Para auxiliar na análise, cabe um exame mais aprofundado deste exemplo. O atributo *country* possui duas instâncias, “USA” e “Japan”. Sendo este atributo o de mais alto nível na dimensão *Geography* e esta dimensão estando subordinada à dimensão *Salesman*, teremos a repetição das instâncias “USA” e “Japan” para cada instância de *salesman*. Espera-se que uma interface OLAP flexível seja capaz de manter diferentes configurações para cada repetição de instâncias em cada um dos níveis hierárquicos da dimensão. Porém, a estrutura das duas metades do eixo horizontal (*i.e.*, tudo que há sob cada uma das duas instâncias de *salesman*, “Venk” e “Netz”) é exatamente a mesma, o que não permite que se analisem de modo satisfatório as capacidades do modelo no que diz respeito à flexibilidade de análise. Por exemplo, não é possível afirmar, baseando-se apenas na Figura 33, se as duas “metades” do eixo horizontal têm necessariamente que ser iguais ou se trata de uma coincidência.

Continuando a análise da tabela, a instância “USA” do atributo *país* se encontra aberta, com o nível hierárquico imediatamente inferior, *região*, tendo as instâncias “USA_N” e “USA_S”. A instância “USA_S” não possui detalhamento, enquanto a instância “USA_N” está detalhada ainda em um nível menos agregado, *cidade*, possuindo as instâncias “Seattle” e “Boston”. Para a instância “Japan” não há qualquer detalhamento.

O fato de as duas “metades” do eixo serem exatamente iguais limita sobremaneira a expressividade do exemplo dado, o que acaba por ocultar limitações na formulação do modelo CPM, conforme se verá.

De acordo com Maniatis *et al.* (2003), um *axis schema* é representado por uma lista de dimensões, cada uma associada a uma lista de atributos. A interface OLAP emulada na Figura 34 tem, segundo o artigo, os metadados de seus eixos definidos conforme as Equações 1 e 2.

$$Row_S = \{ [Tempo], [mês, trimestre, trimestre, mês] \} \quad (1)$$

$$Column_S = \{ [Vendedor \times Local], [vendedor] \times [cidade, região, país] \} \quad (2)$$

De acordo com a Equação 1, o eixo vertical (chamado “Row_S”) possui apenas a dimensão Tempo. Seus níveis dimensionais, representados por uma lista de atributos associada, são mês, trimestre, trimestre e mês, sendo estes os níveis hierárquicos *para cada instância* do atributo trimestre da dimensão Tempo. A primeira e a quarta instâncias (“Qtr1” e “Qtr4”) do atributo quarter são visualizadas em um nível menos agregado, ou seja, no nível dado pelo atributo month. A segunda e a terceira instâncias (“Qtr2” e “Qtr3”), por outro lado, são sofreram um *drill-down*, sendo visualizadas em seu nível mais agregado, dado pelo atributo quarter.

No eixo horizontal (chamado “Column_S”), de acordo com a Equação 2, estão as dimensões Vendedor e Local. A dimensão Vendedor possui apenas um nível hierárquico, dado pelo atributo vendedor. A dimensão Local é agregada pela dimensão Vendedor e possui três níveis hierárquicos, dados pelos atributos cidade, região e país, sendo o nível mais agregado dado pelo atributo país e o menos agregado dado pelo atributo cidade. O atributo vendedor possui duas instâncias, “Venk” e “Netz”. O atributo país possui, também, duas instâncias, “USA” e “Japan”, para cada grupo de instâncias pertencente a cada instância de vendedor. Ou seja, há um par de instâncias [“USA”, “Japan”] para cada instância de vendedor. Os outros níveis hierárquicos, região e cidade, e suas instâncias, podem ser compreendidos de modo análogo.

4.1.3.1 Examinando a relação AxisSchema/Axis no modelo CPM

Observando-se a Equação 1, é possível perceber que cada instância do atributo trimestre na interface OLAP, emulada na Figura 34, é representada nos metadados do eixo (*axis schema*). A primeira e a quarta instâncias, “Qtr1” e “Qtr4”, estão abertas e seu nível menos agregado é month. A segunda e a terceira instâncias, “Qtr2” e “Qtr3”, estão fechadas. É de fundamental importância notar que há um atributo nos metadados do eixo para cada instância de nível hierárquico do atributo quarter. Caso houvesse dezenas de instâncias de trimestre, o

axis schema se estenderia por dezenas de atributos, sobrepondo responsabilidades de controle de dados e de metadados.

A Equação 2, relativa ao eixo horizontal, traz um pouco mais de complexidade. O *axis schema* descrito nesta equação contém como dimensões *Vendedor* no nível hierárquico mais alto e, abaixo desta, *Local*. À primeira dimensão está relacionado o atributo *vendedor*, que se refere ao nome do vendedor. Os atributos *cidade*, *região* e *país* estão relacionados à segunda dimensão, em ordem crescente de agregação. O esquema expresso na Equação 2 descreve apenas uma das “metades” do eixo horizontal da interface OLAP na Figura 34, supondo implicitamente que operações de *drilling* em instâncias individuais em uma das metades irá sempre acarretar em mudanças espelhadas na outra “metade”. Isto é explicitado pelo próprio artigo, quando define o eixo vertical no exemplo com um atributo para cada instância de *trimestre*. Na verdade, o artigo utiliza uma regra para a definição do eixo horizontal (colunas) e outra para a definição do eixo vertical (linhas), conforme se percebe ao confrontar as Equações 1 e 2 com a Figura 34. Se há uma lógica subjacente que explique as duas definições, esta não aparece claramente no artigo. Esta definição leva a uma situação na qual a flexibilidade é prejudicada, já que, a julgar pela definição nas equações citadas, instâncias individuais não podem ser livremente manipuladas sem violar a definição formal dos metadados do eixo (*axis schema*).

Assim, diferentemente da definição na Equação 1, não existe na Equação 2 uma entrada nos metadados para cada instância do atributo de nível mais alto (no caso, *salesman*). Note que, aqui, os metadados não descrevem a quantidade de instâncias de dados, como na primeira equação. Independente de qual alternativa se considere a melhor, constata-se que há duas definições divergentes para os metadados dos dois eixos. Este problema não foi citado pelos autores no trabalho original (MANIATIS *et al.*, 2003), nem em trabalhos subsequentes (MANIATIS *et al.*, 2003(a); MANIATIS, 2003; MANIATIS, 2004), que se utilizaram do CPM para desenvolvimentos posteriores.

		Venk				Netz		
		USA				USA		
		USA_N		USA_S	Japan	USA_N		Japan
		Boston	Seattle			USA_N	USA_S	
Qtr1	Jan							
	Feb							
	Mar							
Qtr2								
Qtr3								
Qtr4	Oct							
	Nov							
	Dec							

Figura 35. Tela OLAP após operação de *roll-up*.

Na Figura 35, é apresentada a mesma interface OLAP da Figura 34, após aplicada uma operação de *roll-up* sobre a instância “USA_N” do atributo *região* da dimensão *Local*, pertencente ao grupo de instâncias sob a instância “Netz” do atributo *vendedor* da dimensão *Vendedor*. De acordo com a definição formal de *axis schema* em Maniatis *et al.* (2003), metadados para um eixo são conforme a Equação 3.

$$AS_K = \left(\begin{array}{l} [DG_1 \times DG_2 \times \dots \times DG_K], \\ \left[[ag_1^1, ag_1^2, \dots, ag_1^{K_1}] \times [ag_2^1, ag_2^2, \dots, ag_2^{K_2}] \times \dots \times [ag_K^1, ag_K^2, \dots, ag_K^{K_K}] \right] \end{array} \right) \quad (3)$$

Um axis schema para o eixo horizontal na visão OLAP da Figura 35, que violaria a definição formal na Equação 3, seria como o demonstrado na Equação 4.

$$Column_S = \left\{ \begin{array}{l} [Salesman \times Geography], \\ [salesman] \times [city, region, country], \\ [salesman] \times [region, country] \end{array} \right\} \quad (4)$$

Assim, para representar duas “metades” distintas no eixo horizontal foi necessário duplicar a quantidade de metadados para o eixo. Deste modo, a definição de um *axis schema*, expressa na Equação 3, não se mostra capaz de representar interfaces OLAP com configurações complexas e diversificadas. Se houvesse um número maior de instâncias de *vendedor*, cada qual possuindo uma configuração hierárquica diferente em seus níveis menos agregados, a representação dos metadados do eixo fatalmente se tornaria complexa e extensa a níveis extremos, a ponto de inviabilizar o seu uso.

4.1.4 Extensões à camada de apresentação do CPM

Face às limitações expostas na seção anterior, torna-se necessário propor extensões à camada de apresentação do Cube Presentation Model a fim de representar interfaces OLAP de maior complexidade, permitindo que o usuário tenha mais flexibilidade para a realização de suas análises.

Inicialmente, é preciso detalhar o conceito de *ponto*. Um ponto, no CPM, não é um ponto qualquer em um plano n-dimensional, mas um ponto sobre um eixo, o que, geometricamente, implicaria que este possui todas as suas coordenadas em zero, exceto uma. Mais precisamente no que tange à visualização multidimensional, um ponto é uma entidade associada a um eixo que se constitui, basicamente, em um conjunto de condições de igualdade (*equally selection conditions*). Uma condição de igualdade se constitui de um atributo e um valor. Obrigatoriamente, os atributos em uma condição de igualdade associada a um eixo devem ser todos provenientes de dimensões que constam do esquema do eixo. Alguns pontos são mostrados na Figura 36.

		Venk				Netz		
		USA			Japan	USA		Japan
		USA_N		USA_S		USA_N	USA_S	
		Seattle	Boston					
Qtr1	Jan							
	Feb							
	Mar							
Qtr2								
Qtr3								
Qtr4	Oct							
	Nov							
	Dec							

Figura 36. Pontos em destaque em uma interface OLAP.

Um ponto é composto por todas as células que o formam. Por exemplo, o ponto marcado na Figura 36 como 1 é definido pelas condições de seleção $[Quarter.quarter = "Qtr4"]$ e $[Quarter.month = "Oct"]$. O ponto 2 é definido pelas condições de seleção $[Salesman.salesman = "Venk"]$, $[Geography.country = "USA"]$, $[Geography.region = "USA_N"]$ e $[Geography.city = "Seattle"]$. O ponto 3, por fim, abrange as condições de seleção $[Salesman.salesman = "Netz"]$ e $[Geography.country = "Japan"]$.

É importante ressaltar que a definição de condição de seleção de igualdade aqui apresentada difere um pouco daquela que é implicitamente apresentada no artigo original, pois este define um ponto conforme a Equação 5, na qual se pode perceber que há uma condição de igualdade dada por uma *ancestor function*, que age sobre região e cidade e tem como parâmetro cidade. Isto acarreta em que, no CPM original, o que foi marcado na Figura 36 como sendo o ponto 1, na verdade seria apenas parte de um único ponto, que abrangeria tudo sob a instância “Qtr4”, pois o conjunto de condições de seleção de igualdade para o ponto seria definido como $[anc_{month}^{quarter}(month) = "Qtr4"]$.

$$p_2 = ([salesman, [city, size(city)]], [salesman = "Venk", anc_{city}^{region}(City) = "USA_N"]) \quad (5)$$

Neste trabalho, porém, optou-se por não utilizar tal função, pois se entende que é uma funcionalidade intrinsecamente ligada ao núcleo de uma máquina OLAP, e não a uma

arquitetura de visualização. Assim, para preservar a independência da camada de apresentação em relação à camada lógica (que é uma premissa do próprio trabalho original que define o CPM), escolheu-se não exportar esta função para a visualização, deixando-a restrita à camada lógica³⁷, tomando-se cada instância de nível hierárquico mais baixo como origem de um ponto.

A Equação 1 mostra uma definição que especifica instâncias individuais de dados. Por exemplo, quando a citada equação, supostamente referente aos metadados de um eixo, especifica `[month, quarter, quarter, month]` como atributos, está se referindo, na realidade, às instâncias de dados da dimensão, ou seja, precisamente às quatro instâncias de trimestre, com a diferença de que a primeira e a quarta instâncias estão sendo visualizadas no nível do mês, conforme se pode verificar na Figura 34 utilizada como exemplo.

O problema da baixa coesão se evidencia quando se pretende utilizar o mesmo *schema* para vários eixos. Em um caso destes, cada mudança nas instâncias de uma dimensão acarretaria em mudanças nos metadados e, logo, sua propagação para todos os eixos. E, ainda, este baixo coeficiente de coesão leva a um aumento do acoplamento entre os objetos auxiliares e a implementação do eixo. Por exemplo, caso uma condição de seleção (*SelectionCondition*, cf. Figura 30) seja modificada para um cubo, seriam modificadas as seleções de suas dimensões, alterando as instâncias do eixo e, conseqüentemente, os metadados do eixo. Esta propagação de alterações em dados até um nível que deveria estar restrito a metadados é altamente desaconselhável.

Assim, para aumentar a coesão da classe *AxisSchema*, é preciso eliminar de sua formulação todos os aspectos concernentes a instâncias de dados, permitindo-lhe focar exclusivamente nos metadados necessários à definição de um eixo. A sintaxe da definição de um *AxisSchema* conforme definida por Maniatis *et al.* (2003) e expressa na Equação 3 não necessita ser modificada. O que deve ser reelaborado é sua semântica.

Originalmente, um *AxisSchema* sustenta metadados dos eixos e metadados das instâncias. Para uma alta coesão, propõe-se aqui a que objetos da classe *AxisSchema* deixem definitivamente de tratar os metadados das instâncias, concentrando-se nos metadados dos eixos.

Usando esta interpretação, as Equações 1 e 2 podem ser reescritas, respectivamente, como as Equações 6 e 7.

³⁷ Camada lógica que, na implementação proposta na presente dissertação, corresponde ao *core* do PostGeoOlap. A ferramenta, porém, não implementa explicitamente uma função similar à *ancestor function*. Um efeito similar à aplicação desta função, no PostGeoOlap, se estabelece pela mera ausência de seleções em um determinado nível de uma dimensão.

$$Row_S = \{ [Quarter], [quarter, month] \} \quad (6)$$

$$Column_S = \{ [Salesman \times Geography], [salesman] \times [city, region, country] \} \quad (7)$$

As Equações 6 e 7, porém, não poderiam, sozinhas, gerar a interface de exemplo, pois, em sua nova conceituação, possuem menos informação que a semântica do CPM original. Assim, estas equações gerariam uma configuração como a mostrada na Figura 37.

		Venk								Netz							
		USA				Japan				USA				Japan			
		USA_N		USA_S		Kanto		Kansai		USA_N		USA_S		Kanto		Kansai	
		Seattle	Boston	Florida	Texas	Tokyo	Yokohama	Osaka	Kyoto	Seattle	Boston	Florida	Texas	Tokyo	Yokohama	Osaka	Kyoto
Qtr1	Jan																
	Feb																
	Mar																
Qtr2	Apr																
	May																
	Jun																
Qtr3	Jul																
	Ago																
	Sep																
Qtr4	Oct																
	Nov																
	Dec																

Figura 37. Visualização de todas as instâncias individuais dos níveis definidos.

É importante ressaltar que as definições nas Equações 6 e 7 não guardam qualquer relação com a quantidade de instâncias de dados das dimensões. Os conjuntos de atributos (*attribute sets*) em *Row_S* e *Column_S*, em suas novas formulações, se referem apenas aos níveis hierárquicos (*i.e.*, atributos e dimensões) envolvidos na interface OLAP. Como consequência, as Equações 6 e 7 possuem menos informação que as Equações 1 e 2, não possuindo quaisquer dados relativos a operações de *drilling* sobre instâncias específicas. A opção de projeto, aqui, foi manter nos metadados do eixo apenas informações estruturais gerais. Assim, todas as instâncias de todos os níveis hierárquicos envolvidos nos eixos estão no mesmo estado, conforme mostra a Figura 37.

A Equação 6, que mostra o *schema* para o eixo vertical, informa que a dimensão *Quarter* é a única envolvida neste eixo. Nesta dimensão, os níveis hierárquicos selecionados para visualização são *quarter* e *month*. Com base nestas informações, a interface é montada com todos os trimestres e todos os meses disponíveis, limitados apenas pelas definições de condições de seleção para o eixo.

Do mesmo modo, a Equação 7, que traz o *schema* para o eixo horizontal informa que estão envolvidas as dimensões *Salesman* e *Geography*, com a primeira no nível hierárquico mais agregado. Para primeira dimensão o nível hierárquico envolvido é *salesman* e para a

segunda, *city*, *region* e *country*, em ordem do menos agregado para o mais agregado. Todos os vendedores, todos os países, regiões e cidades tornam-se visíveis, e limitados pelas condições de seleção.

Do exposto, percebe-se que os metadados do eixo não logram dar conta de operações de *drilling* sobre instâncias individuais. Para lidar com o funcionamento dinâmico de uma interface OLAP, propomos a inclusão de uma entidade ao modelo conceitual do CPM. A classe *PointSchema* representa os metadados para um ponto e é definido de modo similar a um *AxisSchema*. Inicialmente, um ponto não possui qualquer definição de metadados e sua estrutura segue as definições contidas no *schema* do eixo ao qual pertence. Se um usuário requisita um *roll-up* ou *drill-down* sobre uma instância em particular de um nível hierárquico, ao ponto contendo esta instância é atribuído um objeto de metadados (*point schema*). Se, posteriormente, com as operações do usuário, a instância retornar a um estado de concordância com o esquema geral dos metadados do eixo, o ponto passa a não ter qualquer metadado novamente, voltando a seguir o esquema do eixo.

Um *point schema* possui ainda outra associação com objetos do tipo *Point*. Quando o usuário aplica uma operação de *roll-up* sobre uma instância específica, vários pontos são unidos a um só, como mostra a Figura 38, onde (a) mostra um detalhe de interface OLAP antes de uma operação *roll-up* e (b) o mesmo após a aplicação da operação, onde os três pontos iniciais são unidos a um só.

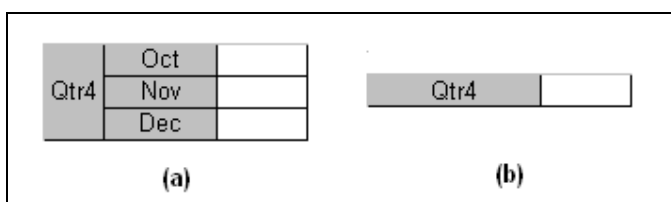


Figura 38. Detalhe de interface OLAP (a) antes e (b) depois de uma operação *roll-up*.

Ocorre que a informação ligada aos pontos iniciais é perdida quando se aplica este tipo de operação, gerando um novo ponto no eixo. Ora, se a informação é perdida, caso o usuário, imediatamente após o *roll-up*, solicitar um *drill-down*, que levaria a interface do estado (b) de volta ao estado (a), os dados perdidos teriam que ser solicitados novamente ao *engine* OLAP subjacente. Para evitar solicitações redundantes ao *core*, pode-se, no momento da implementação, desenvolver um esquema de *caching*, valendo-se da característica não-volátil dos *data warehouses* (INMON, 1997). O diagrama de classes UML mostrando as extensões propostas é mostrado na Figura 39.

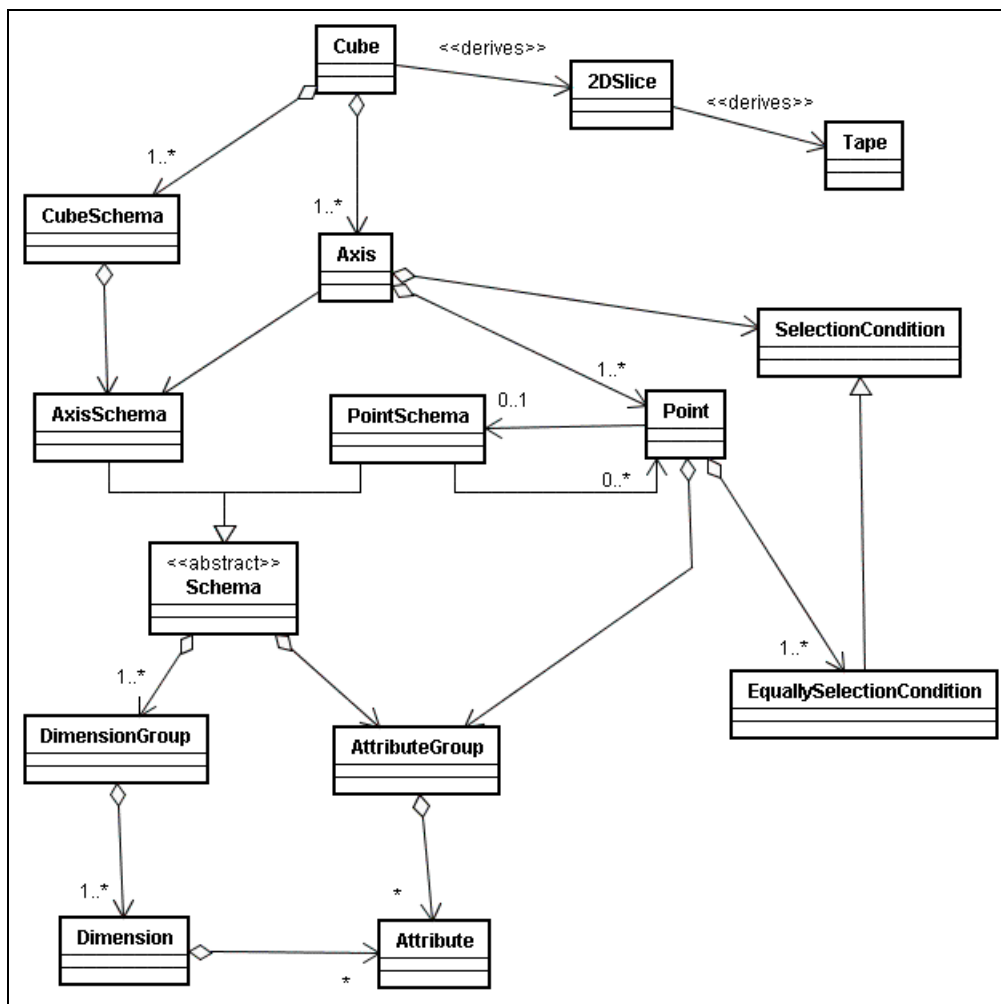


Figura 39. Modelo conceitual do CPM acrescido de extensões.

O uso de metadados especificamente para pontos confere flexibilidade à manipulação de dados multidimensionais, na medida em que permite a livre realização de operações OLAP de modo independente e eficiente para quaisquer instâncias específicas. O exemplo na Figura 40 mostra uma interface OLAP que poderia ser gerada a partir das extensões aqui propostas.

			Venk				Netz				Pit:	
			USA			Japan	USA	Japan				
			USA_N		USA_S			Kanto		Kansai		
			Boston	Seattle				Tokyo	Yokohama	Kyoto		Osaka
Qtr1	Jan											
	Feb											
	Mar	1										
		2										
Qtr2												
Qtr3												
Qtr4	Oct											
	Nov											
	Dec											

Figura 40. Tela OLAP após múltiplas operações de *drilling*.

4.1.5 Exemplificando a Extensão Proposta

Mostrar-se-á aqui um exemplo passo-a-passo da evolução dos metadados para os pontos partindo de um estado inicial dado por um *axis schema* para atingir a configuração mostrada na Figura 40. Inicialmente, com os *axes schemata* dados pelas Equações 8 e 9, tem-se uma interface OLAP com a configuração mostrada na Figura 41.

$$Row_S = \{ [Quarter], [quarter] \} \quad (8)$$

$$Column_S = \{ [Salesman \times Geography], [salesman] \times [country, region] \} \quad (9)$$

	Venk				Netz				Pitz			
	USA		Japan		USA		Japan		USA		Japan	
	USA_N	USA_S	Kanto	Kansai	USA_N	USA_S	Kanto	Kansai	USA_N	USA_S	Kanto	Kansai
Qtr1												
Qtr2												
Qtr3												
Qtr4												

Figura 41. Interface OLAP dada pelos *axes schemata*.

Dada a interface na Figura 41, o decisor aplica um *roll-up* sobre o vendedor “Pitz”, obtendo a configuração mostrada na Figura 42. Após este passo, o *point schema* para o ponto dado na Equação 10 é mostrado na Equação 11.

	Venk				Netz				Pitz
	USA		Japan		USA		Japan		
	USA_N	USA_S	Kanto	Kansai	USA_N	USA_S	Kanto	Kansai	
Qtr1									
Qtr2									
Qtr3									
Qtr4									

Figura 42. *Roll-up* no vendedor “Pitz”.

$$p_1 = ([salesman = "Pitz"]) \quad (10)$$

$$PS_{p_1} = \{ [Salesman \times Geography], [salesman] \times [] \} \quad (11)$$

Agora, o usuário deseja visualizar as vendas de Venk para o Japão e de Netz para os EUA sem levar em conta as regiões. Para isto aplica operações de *roll-up* sobre as instâncias citadas, obtendo a configuração mostrada na Figura 43. Após este passo, os novos pontos gerados são mostrados nas Equações 12 e 13 e os seus metadados são definidos na Equação 14.

	Venk			Netz			Pitz
	USA		Japan	USA	Japan		
	USA_N	USA_S			Kanto	Kansai	
Qtr1							
Qtr2							
Qtr3							
Qtr4							

Figura 43. *Roll-up* em país|região.

$$p_2 = ([salesman = "Venk"], [country = "Japan"]) \quad (12)$$

$$p_3 = ([salesman = "Netz"], [country = "USA"]) \quad (13)$$

$$PS_{p_i} = \{ [Salesman \times Geography], [salesman] \times [country] \} \quad (14)$$

Como próximo passo, o usuário deseja visualizar as informações relativas ao primeiro e quarto trimestres mês a mês. Para isto, aplica um *drill-down* às instâncias “Qtr1” e “Qtr4”, obtendo uma configuração semelhante à Figura 44 em sua interface OLAP. Neste caso, os seis pontos resultantes estão todos ligados ao *schema* dado na Equação 15.

		Venk			Netz			Pitz
		USA		Japan	USA	Japan		
		USA_N	USA_S			Kanto	Kansai	
Qtr1	Jan							
	Feb							
	Mar							
Qtr2								
Qtr3								
Qtr4	Oct							
	Nov							
	Dec							

Figura 44. *Drill-down* em “Qtr1” e “Qtr4” .

$$PS = \{ [Quarter], [quarter, month] \} \quad (15)$$

O usuário decide, ainda, visualizar em um maior nível de detalhamento as vendas na região norte dos EUA realizadas por Venk e as vendas no Japão (ou seja, com a seleção corrente, em Kanto e Kansai) para Netz. Para isso, aplica um *drill-down* sobre as três regiões, obtendo a configuração dada na Figura 45. Os seis pontos resultantes estão associados à definição de metadados na Equação 16.

		Venk				Netz				Pitz
		USA			Japan	USA	Japan			
		USA_N		USA_S			Kanto		Kansai	
		Boston	Seattle				Tokyo	Yokohama	Kyoto	
Qtr1	Jan									
	Feb									
	Mar									
Qtr2										
Qtr3										
Qtr4	Oct									
	Nov									
	Dec									

Figura 45. Mais aplicações da operação *drill-down* (Equação 16) .

$$PS_{p_i} = \{ [Salesman \times Geography], [salesman] \times [country, region, city] \} \quad (16)$$

Finalmente, o usuário focaliza sua atenção no mês de março, requisitando visualizar as vendas por quinzena, obtendo a interface mostrada na Figura 40. Os dois pontos resultantes são associados ao *schema* mostrado na Equação 17.

$$PS = \{ [Quarter], [quarter, month, quinzena] \} \quad (17)$$

4.2 ARQUITETURA DO SISTEMA OLAP PROPOSTO

O modelo de visualização multidimensional proposto neste trabalho foi implementado na forma de um *engine*. Assim, é desejável que a implementação deste modelo possa ser utilizada em diferentes contextos, não ficando atrelada diretamente ao modelo lógico dimensional do PostGeoOlap nem ao *toolkit* gráfico ou tecnologia utilizada para apresentação das informações.

A implementação do modelo poderia assim ser utilizada por qualquer tipo de interface visual (*desktop*, *web*, móvel) e utilizar qualquer modelo lógico OLAP. A escolha é relevante nos dias atuais, em que é crescente a diversidade de plataformas e meios para acesso a computadores.

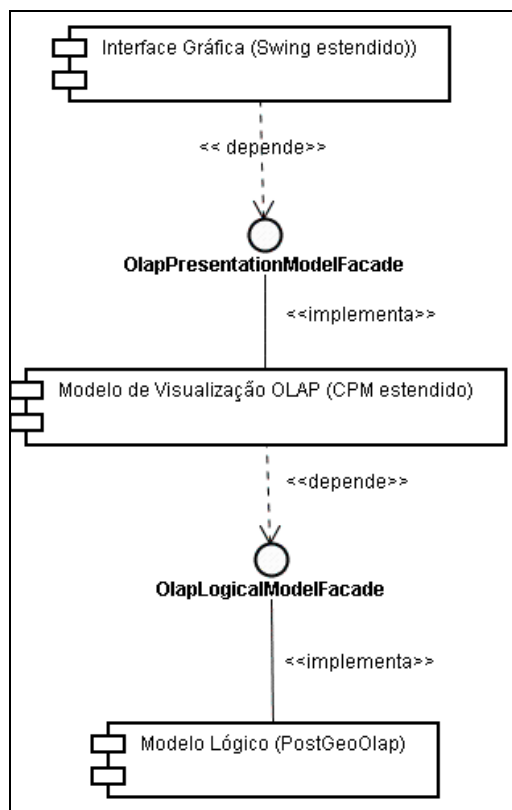


Figura 46. Diagrama de componentes do sistema OLAP.

Inicialmente, é preciso descrever a forma de comunicação entre os componentes, a qual é inteiramente realizada no sentido das camadas externas do *software* para suas camadas mais internas. O usuário faz suas requisições à interface gráfica, que as repassa ao modelo de

visualização multidimensional, que as responde ou repassa ao modelo lógico OLAP subjacente, aguardando a resposta deste para retorná-la à GUI, conforme o diagrama de componentes mostrado na Figura 46. A implementação da interface gráfica enviará suas requisições a um objeto – pertencente ao modelo de visualização OLAP – que implemente a interface `OlapPresentationModelFacade`. A GUI, contudo, não precisa conhecer quem é realmente o objeto, apenas a interface, com o objeto real podendo ser obtido através de uma fábrica (padrão de projeto *Factory Method* proposto por Gamma *et al.* (1995)), configurável através de XML, eliminando quaisquer dependências da interface gráfica com classes concretas de camadas mais baixas. O mesmo ocorre entre o motor de visualização OLAP e a interface `OlapLogicalModelFacade`. Esta deve ser implementada para um modelo lógico específico – no caso, o `PostGeoOlap` – mas o modelo de visualização conhece apenas esta interface e nenhuma classe do `PostGeoOlap`. Deseja-se que o acoplamento entre os subsistemas seja o menor possível e que haja total independência entre os subsistemas que compõem as funcionalidades OLAP propostas aqui.

Nesta seção serão explanadas as formas pelas quais é obtida a eliminação de dependências entre *toolkit*/tecnologia gráfica, modelo de visualização multidimensional e modelo lógico OLAP.

4.2.1 Comunicação entre o Modelo de Visualização e o *Toolkit* Gráfico

Para que o código referente à criação de uma GUI para OLAP possa funcionar independente de modelo de visualização, é necessário que o código que implementa a interface gráfica possa se comunicar com o *engine* de visualização de forma o mais desacoplada possível.

Para proporcionar isto, utilizou-se o padrão de projeto *Facade* (*op. cit.*), de modo que todo o subsistema de visualização multidimensional possa ser acessado através de uma interface única que abstraia todos os seus detalhes internos. Esta interface, escrita na linguagem Java, é mostrada na Listagem 1.

```
import java.util.List;
import java.util.Map;
import java.util.Set;

public interface OlapPresentationModelFacade
{
    Set<String> getCubes();
```

```

    void setActiveCube(String cube);
    String getActiveCube();
    Map<String, Set<List<String>>> getDimensions();
    Set<String> getAllAttributes(String dimension);
    OlapContent pivot(String dimension1, String dimension2);
    OlapContent rollUp(String dimension, String value)
        throws CannotRollUpException;
    OlapContent drillDown(String dimension, String value)
        throws CannotDrillDownException;
    OlapContent drillAcross(String newCube)
        throws NonConformityDimensionException;
    boolean conformity(String currentCubeDimension,
        String otherCube, String otherDimension);
    OlapContent execute(List<String> selectedDimensions,
        List<String> selections);
}

```

Listagem 1. Interface do *façade* para o modelo de visualização multidimensional.

A interface mostrada na Listagem 1 traz todas as operações que a GUI OLAP proposta necessita solicitar às camadas mais baixas do *software*. Antes de serem detalhadas todas as operações, é importante observar que o único tipo utilizado, conforme se pode ver nas parametrizações³⁸ das coleções *List* e *Map*³⁹ é o tipo *String*. Optou-se pela utilização de *strings* por motivos de minimização de acoplamento e ocultação de informação (*information hiding*) acerca da implementação do subsistema de visualização multidimensional. Caso não fosse utilizado o tipo *String*, as classes que implementam a GUI teriam que conhecer classes como *Dimension*, *Attribute* ou *Axis*, específicas da implementação do CPM. Ademais, deseja-se que a interface gráfica não possua conhecimento sobre visualização OLAP – devendo este se concentrar no modelo de visualização – mas simplesmente apresente os dados obtidos das camadas “inteligentes” e receba os eventos gerados pelo usuário, repassando-os à operação correta do *façade*. A interface gráfica deve meramente fornecer o arcabouço de *widgets* necessário para a apresentação, mas, por motivos de coesão e reúso, não deve ter conhecimento sobre a semântica do conteúdo que apresenta.

³⁸ Leitores não familiarizados com evoluções recentes da linguagem Java podem estranhar construções como a mostrada no tipo de retorno da operação. Este tipo de construção se deve à introdução na linguagem Java do recurso, chamado na terminologia Java de *generics* (BRACHA, 2004), mas bastante conhecido em C++ como *template* (STROUSTRUP, 1997). Uma das aplicações mais corriqueiras deste recurso é a parametrização de instâncias de coleções, de modo a que sejam atreladas a tipos específicos, em lugar da classe base *Object*. Isto evita *castings*, aumenta a robustez, minimiza a incidência de erros no tratamento com coleções e estimula o projeto de coleções homogêneas.

³⁹ Tecnicamente, a interface *java.util.Map* não é uma coleção, visto que não é um subtipo de *java.util.Collection*. Porém, afora este detalhe técnico, mapas têm tamanha similaridade com as coleções propriamente ditas do Java que não hesitamos em chamá-las também de coleções, em um sentido não estrito.

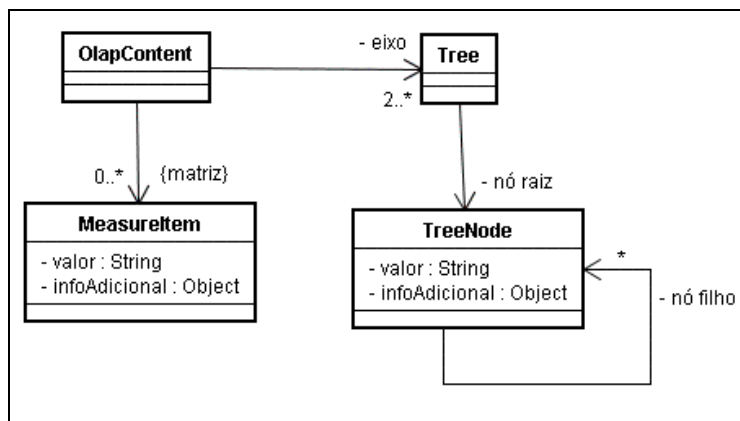


Figura 47. Classes que transportam o conteúdo de uma interface gráfica OLAP.

Quanto às operações da interface `OlapPresentationModelFacade`, a primeira é `Set<String> getCubes()`, que retorna um conjunto contendo os nomes dos cubos disponíveis para que o usuário possa selecionar o cubo no qual será realizada a análise.

Os métodos `void setActiveCube(String cube)` e `String getActiveCube()` têm a função de, respectivamente, alterar e receber a informação sobre qual cubo, dentre os possíveis, é aquele que estará ativo para a instância do modelo de visualização OLAP.

A operação `Map<String, Set<List<String>>> getDimensions()`⁴⁰ não toma parâmetros e retorna um mapa⁴¹ cujas chaves são os nomes de todas as dimensões disponíveis no cubo corrente. Os valores ligados às chaves são conjuntos de listas contendo as hierarquias definidas para a dimensão. Cada lista desse conjunto contém uma hierarquia de atributos para a dimensão dada como chave. No caso do `PostGeoOlap`, seu modelo permite apenas uma hierarquia por dimensão. Porém, como a intenção é manter as camadas isoladas umas das outras, analogamente as limitações não se propagarão entre elas, possuindo, assim, o *engine* de visualização e o adaptador para a interface gráfica o suporte a múltiplas dimensões. No caso do uso com o `PostGeoOlap`, os mapas retornados por este método terão sempre os valores das chaves como um conjunto contendo apenas uma lista.

A operação `Set<String> getAllAttributes(String dimension)` permite obter todos os atributos de uma dimensão, e não apenas aqueles pertencentes a alguma hierarquia. O

⁴⁰ O uso concomitante de `List` e `Set` (*i.e.*, listas e conjuntos) se deve ao fato de que iterações de listas ocorrem exatamente na mesma ordem em que os elementos foram inseridos. Em um conjunto, não se pode assumir isto. Deste modo, toda vez que houve necessidade do respeito à mesma ordem da inclusão no momento da iteração, optou-se pela utilização de `List` em lugar de `Set`.

⁴¹ Um mapa é uma série de pares objeto/valor, onde os valores são indexados pelos objetos: ao fornecer o objeto dado como índice ao mapa, este é capaz de devolver o valor associado ao objeto.

acesso a estas informações por parte dos usuários é importante para a definição de seleções a ser aplicadas sobre os dados da análise, mesmo para atributos que não constam de hierarquias.

A operação `OlapContent pivot(String dimension1, String dimension2)` permite que seja realizada a operação de pivoteamento – ou rotação –, que consiste em modificar os eixos das dimensões. Assim, obedecendo a ações por parte do usuário, a GUI, através desta operação, pode solicitar que a rotação seja realizada no modelo de visualização, passando os nomes das dimensões envolvidas e receber o resultado através de um objeto `OlapContent`. De posse deste objeto, bastará substituir os valores nos *widgets* (basicamente tabela e cabeçalhos) correspondentes. Evidentemente, os nomes das dimensões devem ser únicos para cada cubo.

A operação `OlapContent rollUp(String dimension, String value)` **throws** `CannotRollUpException` realiza uma operação de *roll up* sobre a dimensão selecionada. Caso o parâmetro passado em `value` seja nulo, a operação é realizada para todas as instâncias da dimensão no atributo de menor granularidade. Caso contrário, apenas a instância com o valor passado sofre os efeitos da operação⁴². Se a dimensão escolhida já estiver sendo visualizada em seu atributo mais agregado, é disparada uma exceção `CannotRollUpException`, que pode ser capturada pela classe de GUI, para a apresentação da mensagem apropriada ao usuário. Um objeto `OlapContent` é retornado para que os dados sejam atualizados nos *widgets*.

A operação `OlapContent drillDown(String dimension, String value)` **throws** `CannotDrillDownException` é em tudo análoga a `rollUp`, com a diferença de que executa uma operação *drill down* sobre a dimensão ou instância. Uma exceção é disparada caso o atributo de menor nível hierárquico sendo visualizado na dimensão passada seja o atributo menos agregado na hierarquia corrente.

A operação `OlapContent drillAcross(String newCube)` **throws** `NonConformityDimensionException` realiza uma operação de *drill-across*, trocando o cubo corrente e mantendo as dimensões selecionadas, caso haja dimensões em conformidade no cubo passado como parâmetro. Se não houver dimensões em conformidade, uma exceção `NonConformityDimensionException` é disparada. Um objeto `OlapContent` é retornado com o resultado da operação. A operação `drillAcross` não possui implementação no *core* do `PostGeoOlap` e não é suportada em nenhuma das implementações intermediárias.

⁴² Isto é exatamente uma das funcionalidades que a extensão proposta neste trabalho para o Cube Presentation Model contempla.

Como suporte à operação *drill-across*, pode ser de interesse do usuário saber quais dimensões estão em conformidade com outras em outro cubo. Esta informação pode ser proporcionada através da operação **boolean** `conformity(String currentCubeDimension, String otherCube, String otherDimension)`, que retorna um valor booleano indicando se a dimensão do cubo indicado está em conformidade com a seleção da dimensão passada para a consulta corrente no cubo ativo.

Finalmente, a operação `OlapContent execute(List<String> selectedDimensions, List<String> selections)` permite que a análise OLAP se inicie. A interface gráfica deve passar uma lista contendo as dimensões selecionadas pelo usuário e as seleções sobre atributos destas dimensões, de modo a limitar o universo da análise. Um objeto `OlapContent` retorna o resultado para ser enviado aos *widgets*. Normalmente, implementações desta operação realizam a consulta inicial com todas as dimensões em seu nível hierárquico mais agregado, sendo, assim, mostrada ao usuário para análise. A partir desta configuração inicial, ele poderá, então, detalhar a análise na direção que desejar.

Após receber um objeto `OlapContent` de qualquer uma das operações de `OlapPresentationModelFacade` que o retornam, a interface gráfica entrega a matriz ao modelo do *widget* de tabela que mostrará o resultado. As duas árvores são percorridas – com auxílio um algoritmo básico de caminhamento em profundidade – e durante o caminhamento a estrutura de cabeçalhos hierárquicos vai sendo criada para ambos os eixos. A implementação dos cabeçalhos hierárquicos no *toolkit* gráfico será discutida na seção 4.3. No algoritmo, são efetuadas comparações que evitam a necessidade de reescrever informações iguais, o que traz ganhos de desempenho. Isto acontece, por exemplo, em operações de *drilling* que afetam um dos eixos (ou seja, em termos de GUI, um dos cabeçalhos), o outro eixo não é afetado, de modo que os dados recebidos para este último pela interface gráfica não precisam ser reenviados à tela.

É necessário ainda discutir as classes das quais depende a interface aqui descrita. `OlapContent`, conforme se pode observar no diagrama da Figura 47, é uma classe que traz todos os dados necessários à atualização de uma interface gráfica OLAP. Um objeto `OLAPContent` se associa a duas ou mais árvores contendo, cada uma, a hierarquia de dados (ou seja, pós-consulta) para cada eixo visual. Permite-se uma multiplicidade de dois ou muitos, pois não se pretende inferir *a priori* que a visualização será sempre bidimensional. Por este motivo, o atributo `matriz` é apresentado como sendo do tipo `Object`, pois como não se assume a

quantidade de dimensões da visualização⁴³, não é possível definir uma matriz em tempo de codificação, sendo definida a matriz em tempo de execução através de mecanismos de metaprogramação⁴⁴ (FORMAN e FORMAN, 2004). Além disto, `OlapContent` se associa a uma matriz – conforme dado na restrição indicada no diagrama de classes – de objetos `MeasureItem`. Este tipo de objeto traz o valor em *string* a ser mostrado na tabela e um objeto adicional com semântica a ser definida de acordo com as necessidades e requisitos de cada implementação. Na implementação corrente, com suporte à visualização de dados geográficos, a informação adicional é um vetor de *strings* no formato WKT contendo as descrições dos polígonos associados à medida para serem plotados no mapa. Caso não haja informações geográficas associadas à medida, o atributo `infoAdicional` é nulo.

4.2.2 Comunicação entre o modelo OLAP e o modelo de visualização

Assim como a comunicação entre a GUI e o modelo de visualização, os modelos OLAP e de visualização também devem ser construídos com baixo acoplamento, necessitando, assim, de uma arquitetura que permita seu funcionamento de modo independente.

Para possibilitar isto, utilizou-se o padrão *Facade*, de modo que o modelo de visualização possa fazer requisições ao modelo lógico sem que necessite conhecer seu funcionamento interno. A interface do *Facade*, escrita em linguagem Java, é mostrada na Listagem 2.

```
interface OlapLogicalModelFacade {
    Set<String> cubes();
    Set<String> dimensions(String cube);
    Set<Attribute> attributes(String dimension);
    OlapResult submitQuery(Set<String> dimensions,
        Set<String> selections);
    Hierarchy getHierarchy(String dimension);
}
```

Listagem 2. Interface do *facade* para o modelo de visualização multidimensional.

As operações `cubes()`, `dimensions(String cube)` e `attributes(String dimension)` são as mais simples do *facade* e permitem ao modelo de visualização obter

⁴³ Isto pode ser realizado, pois em Java um vetor, independente do seu número de dimensões, pertence ao supertipo `Object`.

⁴⁴ Também chamada introspecção ou reflexão computacional, consiste no tratamento programático de itens do próprio programa, como classes, métodos e atributos.

informações a respeito, respectivamente, de cubos, dimensões por cubo e atributos por dimensão, oriundos do modelo lógico. Os atributos retornados pela última operação estão na forma de um objeto que implementa a interface `Attribute`, discutida posteriormente.

A operação `submitQuery(Set<String> dimensions, Set<String> selections)` permite que uma consulta seja submetida ao modelo lógico e retorna um objeto que implementa a interface `OlapResult`.

A operação `getHierarchy(String dimension)` permite ao modelo lógico exportar a informação sobre a hierarquia de uma dimensão. Esta informação é retornada como um objeto que implementa a interface `Hierarchy`.

A interface `Hierarchy`, cujo código é mostrado na Listagem 3, representa a hierarquia de uma dimensão no modelo lógico, possuindo operações para a obtenção da descrição textual da hierarquia – `String getDescription()` – e para a obtenção dos itens hierárquicos, na forma de uma coleção de objetos do tipo `HierarchyItem`, na operação `Set<HierarchyItem> items()`.

```
interface Attribute
{
    String getName();
    boolean isGeographic();
    Attribute geometricalRepresentation();
}

interface HierarchyItem
{
    int getLevel();
    String getDescription();
    List<Attribute> attributes();
    Attribute mainAttribute();
}

interface Hierarchy
{
    String getDescription();
    Set<HierarchyItem> items();
}
```

Listagem 3. Interfaces auxiliares do *façade* para o modelo lógico.

A interface `Attribute` representa um atributo e oferece operações para obtenção do nome do atributo – `String getName()` –, para informar se o atributo é geométrico – `boolean isGeographic()` – e para obter o atributo geométrico equivalente a este atributo, caso exista – `Attribute geometricalRepresentation()`. Este último caso pode ser

exemplificado com um atributo `nomeBairro` que teria como representação geométrica um atributo geográfico `bairroGeo` contendo a georreferência para o bairro.

Mostrada na Listagem 3, a interface `HierarchyItem` representa um item de hierarquia, sendo basicamente análoga à classe `ItemHierarquia` no modelo conceitual do PostGeoOlap. A operação `int getLevel()` retorna um número correspondendo ao nível hierárquico do item dentro da hierarquia da qual é um composto. A operação `String getDescription()` retorna uma descrição textual para o nível hierárquico. A operação `List<Attribute> attributes()` retorna uma coleção contendo todos os atributos associados ao nível hierárquico. Finalmente, a operação `Attribute mainAttribute()` retorna o atributo principal do nível hierárquico.

```

interface OlapResult
{
    Set<Equality> equalities();
    Set<MeasureData> measures();
}

interface Equality
{
    Attribute getAttribute();
    String getEqualityValue();
}

interface MeasureData
{
    double getValue();
    Set<Equality> equalities();
}

```

Listagem D. Interfaces auxiliares do *façade* para o modelo lógico.

A Listagem D mostra mais três interfaces, que completam o arsenal necessário para o acesso ao modelo lógico do PostGeoOlap. A interface `OlapResult` representa o retorno de uma submissão de consulta ao *data warehouse*. Contém duas operações para a obtenção das igualdades e das medidas, dadas, respectivamente, pelas assinaturas `Set<Equality> equalities` e `Set<MeasureData> measures()`.

A interface `Equality` representa uma atribuição de igualdade entre um atributo e um valor dimensional, conforme proposto na Seção 4.1.1, possuindo tão-somente operações para a obtenção do atributo e do valor, dadas, respectivamente, pelas assinaturas `Attribute`

`getAttribute()` e `String getEqualityValue()`. Um exemplo de igualdade seria um atributo `vendedor` associado ao valor “Venk”.

Finalmente, a interface `MeasureData` representa um valor numérico obtido do *data warehouse*. Possui a operação `double getValue()`, que retorna o valor numérico em si e a operação `Set<Equality> equalities()`, que proporciona a obtenção das igualdades associadas ao valor numérico. Por exemplo, a um valor numérico de vendas 50.000 estão associadas as igualdades `Vendedor.vendedor = “Venk”`, `Produto.categoria = “Eletrônicos”`, `Tempo.mes = “Janeiro”` e `Tempo.ano = 2006`.

É importante notar que um cliente deste *façade* (por exemplo, uma classe no modelo de visualização) não necessitará acoplar-se de modo algum a classes concretas do modelo lógico, mas tão-somente às interfaces aqui definidas. A obtenção das classes concretas pode ser realizada através de métodos fábrica (GAMMA *et al.*, 1995) ou, preferencialmente, injeção de dependências (FOWLER, 2004).

4.3 SUPORTE A INTERFACES OLAP NO JAVA/SWING

O Swing, *toolkit* padrão para criação de interfaces gráficas em Java, apesar de bastante extenso e poderoso, está distante de possuir o mínimo necessário para a criação de uma GUI OLAP básica. Deste modo, para a implementação do proposto nesta dissertação foi preciso estender o *toolkit* de forma a suportar visualizações OLAP. Em uma implementação básica para tais visualizações, é necessário um *widget* visual do tipo tabela que suporte cabeçalhos aninhados e agrupáveis, tanto no eixo das colunas quanto das linhas.

No Swing, tabelas são objetos da classe `JTable`. Um objeto `JTable` normalmente opera em conjunto com um objeto da classe `JScrollPane`, em uma estrutura que segue o padrão de projeto *Decorator* (GAMMA *et al.*, 1995). No seu uso mais comum, objetos desta classe decoram o componente principal (no caso uma tabela, mas eventualmente, um *widget* para área de texto, lista ou árvore) com barras de rolagem. Os objetos `JScrollPane` também provêm, opcionalmente, cabeçalhos de coluna e de linha para o componente visual decorado. A classe `JTable` utiliza o cabeçalho de coluna de um `JScrollPane` para apresentar o cabeçalho para suas colunas. Este cabeçalho é um objeto da classe `JTableHeader`. A Figura 48 mostra um exemplo de componente `JTable` combinado a um `JTableHeader` e um `JScrollPane`.

As linhas da tabela são o `JTable`, o cabeçalho é um `JTableHeader` e, contendo ambos, um `JScrollPane` (que, quando não há barras de rolagem a mostrar, é invisível).

Logradouro	Bairro	Cidade	Est...	Tipo
Av. Alberto Lamago 2000	Horto	Campos	RJ	Profissional
Rua Dr Siqueira 273	Pq. Dom Bosco	Campos	RJ	Estudo

Figura 48. Exemplo de objetos `JTable`, `JTableHeader` e `JScrollPane` em ação.

Não há, em Swing padrão, qualquer tipo de tabela estruturalmente mais complexa que a mostrada na Figura 48. Não há qualquer suporte a características fundamentais de uma tabela OLAP como cabeçalhos hierárquicos ou cabeçalhos de linha, muito menos cabeçalhos hierárquicos de linha. Assim, para que o Swing suporte uma interface OLAP, por mais simples que seja, uma boa quantidade de trabalho é necessária. Afortunadamente, o Swing é amplamente flexível, de modo que extensões, mesmo drásticas como a que se necessita aqui, são passíveis de realização, ainda que tal tarefa esteja distante do trivial.

4.3.1 Representação de cabeçalhos hierárquicos

Para modelar cabeçalhos hierárquicos foi utilizada uma estrutura do tipo *Composite* (GAMMA *et al.*, 1995). A aplicação deste padrão foi necessária na medida em que a classe `JTableHeader`, que representa cabeçalhos de tabelas no Swing, cria uma entrada de cabeçalho para cada coluna da tabela associada. Assim, em uma visualização do tipo OLAP, que exige a possibilidade de existir colunas aninhadas, haveria a existência de colunas agrupadas. A estrutura criada corresponde ao diagrama de classes da Figura 49⁴⁵, onde um item de cabeçalho pode ser um objeto do tipo `TableColumn` (uma coluna de tabela no Swing) ou um grupo de itens.

⁴⁵ As classes e métodos estão nomeados em inglês devido a sua disponibilidade como Software Livre, visando a um acesso facilitado por parte da comunidade.

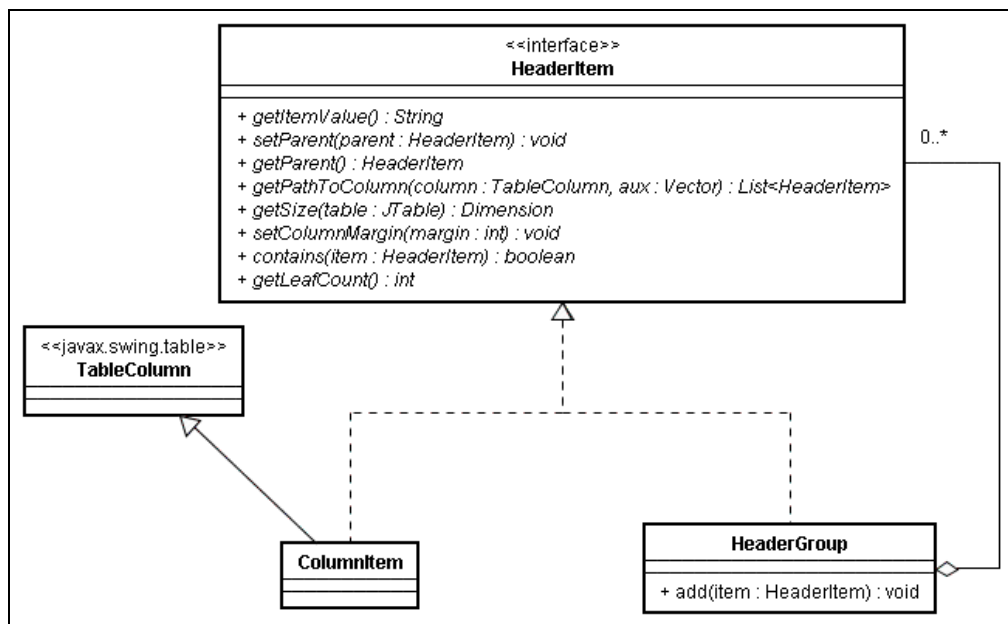


Figura 49. Aninhamento de células no cabeçalho de tabelas

Isto ocorre porque o objeto `JTableHeader` não reconhece aninhamentos, mas apenas colunas simples, em linha. A Figura 50 mostra de modo claro a questão: em preto estão os objetos `TableColumn` e em cinza estão os objetos `HeaderGroup`. Uma metáfora útil para entender o problema é ver o cabeçalho da tabela da Figura 50 como uma árvore cujo nó raiz é invisível e os nós de primeiro nível são “Venk”, “Netz” e “Pitz”. Qualquer item de nível mais baixo – i.e., uma folha da árvore – é, obrigatoriamente um `TableColumn`. Qualquer outro necessariamente conterá um `TableColumn` e será um `HeaderGroup`.

Venk			Netz			Pitz
USA		Japan	USA	Japan		
USA_N	USA_S			Kanto	Kansai	

Figura 50. Em preto, objetos `TableColumn`; em cinza, objetos `HeaderGroup`.

É possível identificar, ainda, que no diagrama da Figura 49 aparece uma classe chamada `ColumnItem`. Esta é meramente um adaptador (GAMMA *et al.*, 1995) para que um `TableColumn` possa ser tratado na estrutura de *composites*.

4.3.2 `JTableHeader` e suporte a cabeçalhos hierárquicos

Com a modelagem de aninhamento de cabeçalhos, o próximo passo é estender os *widgets* Swing para dar suporte a esta funcionalidade. Os componentes envolvidos foram estendidos conforme o diagrama de classes na Figura 51.

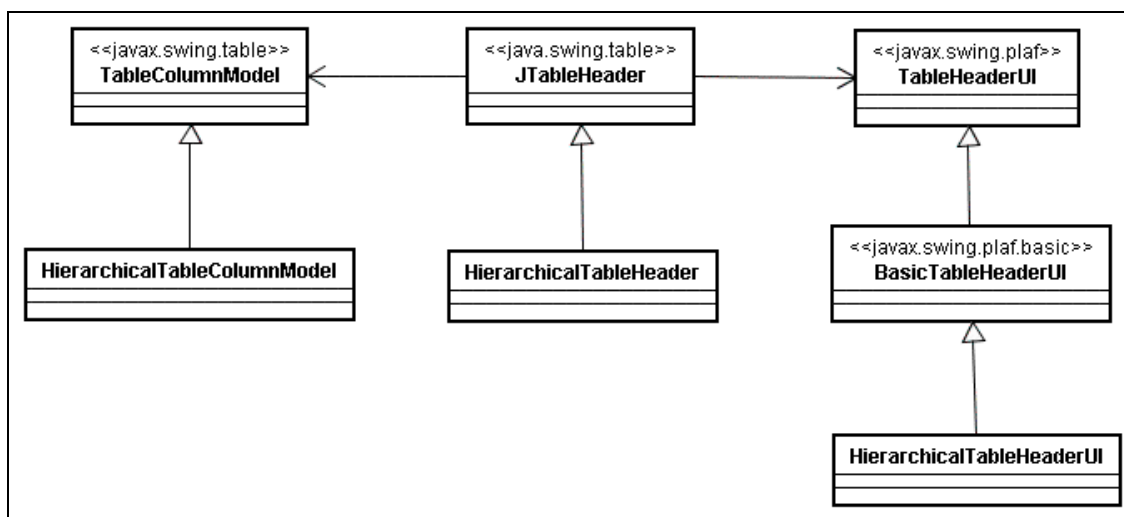
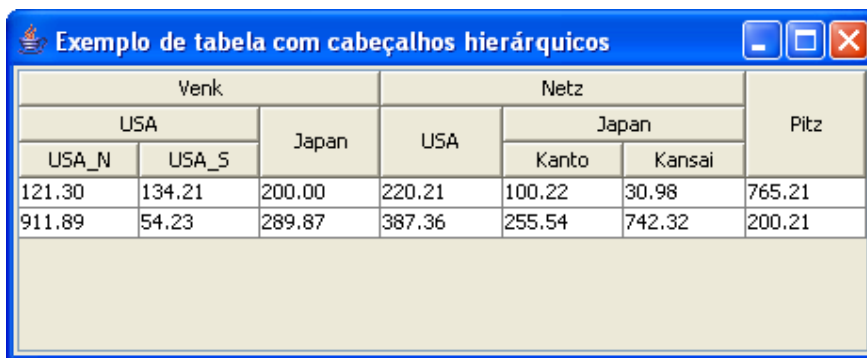


Figura 51. Extensões aos componentes Swing.

As classes estendidas foram `JTableHeader`, `TableColumnModel` e `BasicTableHeaderUI`. Na estruturação MVC subjacente do Swing, os componentes normalmente utilizados pelos programadores atuam como controladores, como é o caso do `JTableHeader`. Todo componente tem um objeto responsável pela visão, que, neste caso, é o `TableHeaderUI`. Este tipo de classe, chamado no jargão do Swing de *UI delegate*, é responsável pelo desenho do componente, tendo normalmente uma implementação de relativamente baixo nível. `TableColumnModel`, por sua vez, é uma classe de modelo (no sentido MVC), responsável por manter os atributos das colunas de uma tabela. Assim, foi necessário estender modelo, visão e controlador para o suporte a cabeçalhos hierárquicos. A Figura 52 mostra um exemplo real de tabela com as extensões citadas em funcionamento.



Venk			Netz			Pitz
USA		Japan	USA	Japan		
USA_N	USA_S			Kanto	Kansai	
121.30	134.21	200.00	220.21	100.22	30.98	765.21
911.89	54.23	289.87	387.36	255.54	742.32	200.21

Figura 52. Exemplo de tabela com cabeçalhos hierárquicos de coluna.

4.3.3 Cabeçalhos de linha

O Swing também não oferece qualquer recurso para cabeçalhos de linha, um item indispensável em uma interface OLAP do tipo tabela. Duas soluções para a questão foram testadas, ambas com seus prós e contras.

Do mesmo modo que um `JTable` utiliza o cabeçalho de colunas do `JScrollPane` para posicionar o cabeçalho `JTableHeader`, será utilizado o cabeçalho de linhas do *scroll pane* para o posicionamento do cabeçalho horizontal. Outro requisito comum é que se deseja que o componente desenvolvido opere do mesmo modo que o cabeçalho de colunas, utilizando objetos do tipo `HeaderItem` para compor seu conteúdo.

A primeira solução proposta é a extensão da classe `HierarchicalTableHeaderUI` de modo a desenhar um `JTableHeader` transposto, em uma operação conforme mostra a Figura 53.

1	2	3	4
5	6	7	8
9	10	11	12

Matriz Original

1	5	9
2	6	10
3	7	11
4	8	12

Matriz Transposta

Figura 53. Exemplo de transposição.

Assim, ao receber os objetos `HeaderGroup` que lhe caibam, o cabeçalho é desenhado de modo transposto, como se fosse uma matriz onde é aplicada uma operação de transposição.

Esta solução é funcional, porém tem um ponto negativo, em termos de arquitetura. Um objeto `JTableHeader` trabalha em conjunto com um `TableColumnModel`, permitindo que haja tantas colunas – ou, com o *UI delegate* estendido, linhas – quantas forem as colunas especificadas pelo modelo. Como o modelo de colunas do componente de tabela não diz respeito a linhas, é necessário estender também a classe `HierarchicalTableColumnModel` de modo a gerar colunas falsas para cada linha de dados na tabela.

Outra solução testada foi abandonar o componente `JTableHeader` e estender `JTable` e `BasicTableUI` de modo a gerar um componente e um renderizador que suportem mesclagem de células. Os atributos de nível mais alto seriam nada mais que células mescladas com um único valor. Neste cenário, o componente de renderização também tem a função de desenhar a tabela com os atributos de um cabeçalho de tabela, para obter uniformidade visual. Isto é possível devido à classe Java `UIManager`, que fornece as características *default* dos componentes do Swing, que são a fonte para o desenho desta tabela, cujo visual emula o visual de um cabeçalho de coluna. O uso do `UIManager` também torna a solução imune a mudanças no *Look and Feel*⁴⁶. O ponto negativo desta alternativa é a necessidade da criação de um componente tabela que imite a aparência de um cabeçalho. É importante notar que com esta solução, o cabeçalho das linhas não seria um objeto do tipo `JTableHeader`, mas um `JTable` com suporte a mesclagem de células e *look and feel* emulando um cabeçalho, visualmente idêntico. Isto faria com que fosse perdida a capacidade de modificar a altura das linhas (conforme o `JTableHeader` permite fazer com a largura das colunas). Porém isto não se traduz em problema para a esmagadora maioria dos casos, pois a alteração da altura das linhas raramente é um requisito. Devido a uma melhor resposta nos testes realizados e à possibilidade de uma implementação mais simples e elegante, esta foi alternativa escolhida.

Uma implementação ideal estenderia `BasicTableHeaderUI` e eliminaria as dependências de `HierarchicalTableHeader` com modelos de colunas. Isto não foi feito aqui, pois demandaria alterações profundas na classe de cabeçalho, pois a classe base `JTableHeader` possui um forte acoplamento com um modelo de colunas. As duas soluções, porém, funcionam a contento para o usuário final. A Figura 54 mostra um exemplo real de

⁴⁶ Look and Feel é uma característica do Swing que permitem que o visual de uma aplicação inteira seja alterado pela troca de classes que definem a aparência geral dos *widgets*. É muito utilizado para emular o visual padrão de sistemas operacionais ou ambientes gráficos. Os exemplos de aplicações reais expostos em figuras nesta dissertação utilizam um look-and-feel que emula o visual do Windows XP.

interface OLAP utilizando a segunda solução proposta. Com isto, já é possível se criar interfaces OLAP básicas do tipo tabela.



Exemplo de cabeçalhos hierárquicos para linha e coluna

			Venk			Netz			Pitz
			USA		Japan	USA	Japan		
			USA_N	USA_S			Kanto	Kansai	
2004			121.30	134.21	200.00	220.21	100.22	30.98	765.21
2005	1	1	911.89	54.23	289.87	387.36	255.54	742.32	200.21
		2	121.30	134.21	200.00	220.21	100.22	30.98	765.21
		2	911.89	54.23	289.87	387.36	255.54	742.32	200.21
2006	1	121.30	134.21	200.00	220.21	100.22	30.98	765.21	
	2	911.89	54.23	289.87	387.36	255.54	742.32	200.21	

Figura 54. Exemplo de cabeçalhos hierárquicos para linha e coluna.

5 ESTUDO DE CASO

O presente capítulo tratará de expor um estudo de caso da aplicação dos modelos e arquitetura propostos nesta dissertação. O estudo será aplicado em um *data warehouse* parcial⁴⁷ do setor de tributos da Prefeitura Municipal de Cachoeiro do Itapemirim (FREITAS, 2007).

5.1 DESCRIÇÃO

Um órgão de administração pública, como uma prefeitura, na concepção econômica liberal, dominante em nossos dias, não possui recursos advindos de atividades econômicas conforme o fazem as empresas privadas, salvo exceções previstas na Constituição Federal. Assim sendo, a principal fonte de recursos de um órgão do aparelho estatal é a captação do setor privado através do instituto jurídico do tributo (MACHADO, 1992).

Deste modo, afigura-se de extrema utilidade a existência de um sistema de suporte à decisão que permita que administradores públicos possam obter valiosas informações a respeito de sua área de atuação e, de posse destas, tomar decisões adequadas.

O trabalho de Freitas (2007) apresenta um detalhado estudo de caso sobre o *data warehouse* aqui discutido. Na presente dissertação, o estudo resumir-se-á à validação das

⁴⁷ Diz-se “*data warehouse* parcial” pois este trabalho não teve acesso a quaisquer dados ou informações que possam identificar contribuintes individuais do município.

propostas apresentadas no capítulo anterior. A modelagem do *data warehouse* mostrada no tópico a seguir é extraída do mesmo trabalho.

5.2 MODELAGEM DO DATA WAREHOUSE

O recorte do *data warehouse* mostrado na Figura 55, resume-se à área de administração dos tributos e, notadamente, à parte tributária que diz respeito às informações sobre itens de débito. Um débito pode ser composto por vários itens, de modo que a granularidade do *data warehouse* aqui utilizado é a mais fina possível, possibilitando análises mais acuradas ao usuário. De fato, conforme ensinam Kimball e Ross (2002), os modelos dimensionais devem ser desenvolvidos visando-se à informação de mais fina granularidade – nas palavras dos autores, atômicas – capturada pelos processos de negócio, proporcionando um alto grau de flexibilidade e dimensionalidade. As restrições tecnológicas que, no passado, dificultavam o uso de granularidades ao nível atômico (KIMBALL, 1996), se encontram há anos superadas pelo estágio de desenvolvimento tecnológico, tanto de *hardware* quanto de *software* (KIMBALL e ROSS, 2002).

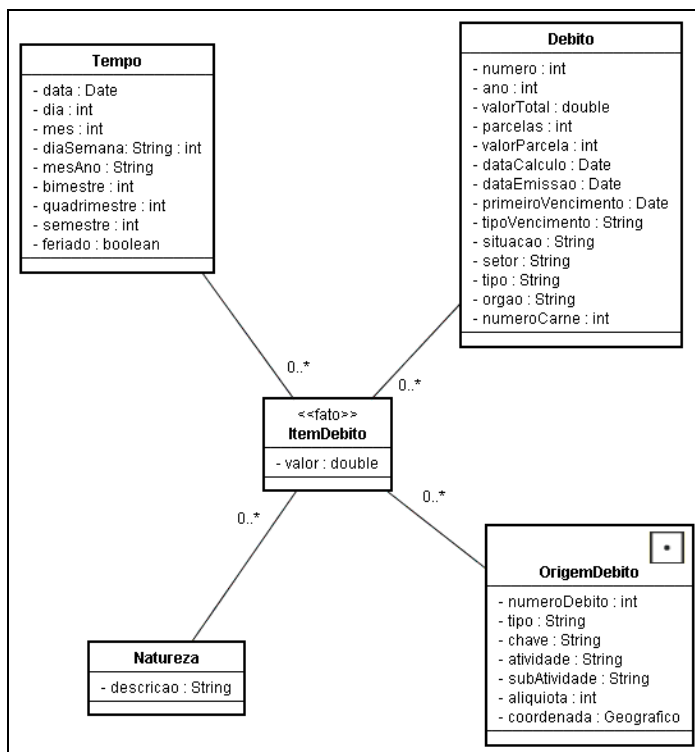


Figura 55. Diagrama de classes conceitual do modelo estrela para o estudo de caso.

5.2.1 Projeto do Banco de Dados

No modelo lógico proposto na Figura 55, encontram-se as dimensões *OrigemDebito*, *Natureza*, *Debito* e *Tempo*; e a dimensão de medidas *ItemDebito*.

A dimensão *Debito* contém informações acerca do débito propriamente dito. A Tabela 4 mostra os atributos da dimensão e uma breve explicação a respeito de sua semântica.

Atributo	Significado
numerodebito	Chave única oriunda dos sistemas transacionais.
anodebito	Exercício no qual o débito foi contraído.
valortotal	Valor total do débito.
parcelas	Número de vezes em que o pagamento do débito foi dividido.
valorparcela	Valor de cada parcela do pagamento, exceto a primeira.
valor1parcela	Valor da primeira parcela do pagamento do débito, que pode ser diferente das subseqüentes.
datavenceito	Contém a data do primeiro vencimento, com as datas das demais parcelas mensais sendo calculadas a partir desta.
tipovencimento	Define o modo pelo qual as datas de vencimento serão desviadas de feriados ou fins de semana, contendo os valores 1 para antes e 2 para depois.
situacao	Armazena o estado do débito: não pago (0), parcelado no setor de origem (1), quitado (2), carnê de dívida ativa (3), parcelado em dívida ativa (4), anistiado (5), cálculo diferenciado (6), carnê de tributação (7), parcelado em tributação de receitas (8) e cancelado (9).
setor	Define se o débito está atualmente localizado nos setores de dívida ativa, cadastro imobiliário, tributação e receitas, taxas avulsas ou fiscalização tributária.
tipo	Contém o tipo de chave do débito: sem chave (dívida ativa, 0), inscrição municipal (1), inscrição imobiliária (2, 3, 4), código único do contribuinte (5) ou auto de infração (6).
siglaorgao	Armazena a sigla do órgão onde o débito foi originado.
usuario	Usuário que cadastrou o débito.

datalanctdivat	Contém o dia em que o débito será transferido para a dívida ativa, caso não seja quitado antes.
numerocarne	Número do carnê para pagamentos parcelados.
Anocarne	Ano do carnê para pagamentos parcelados.

Tabela 4. Atributos da dimensão Debito.

A dimensão *OrigemDebito* funciona como um complemento da dimensão *Debito*, contendo campos relativos à origem do débito, se provém de IPTU (Imposto Predial e Territorial Urbano) ou ISS (Imposto Sobre Serviços) ou outros. Esta dimensão possui um dado geométrico concernente à localização do imóvel envolvido no débito.

A Tabela 5 mostra os atributos da dimensão *OrigemDebito* e uma breve explicação a respeito de seus significados.

Atributo	Significado
numerodebito anodebito tipo	Função idêntica àquela desempenhada pelos atributos de mesmo nome na dimensão <i>Debito</i> .
chavedebito	Inscrição imobiliária ou inscrição estadual do imóvel ou pessoa jurídica responsável pelo débito.
atividade descatividade subatividade descsubatividade aliquota	Estes atributos somente terão função se a origem do débito for o ISS. <i>atividade</i> e <i>descatividade</i> são, respectivamente, código e descrição do ramo de atividade em que a pessoa jurídica atua. O mesmo ocorre com os dois atributos seguintes, com a diferença que se trata do sub-ramo de atividades. O atributo <i>aliquota</i> representa a alíquota cobrada.
coordenada	Este atributo contém um parâmetro geométrico, compatível com a especificação OpenGIS (1999), referente ao ponto central do imóvel.

Tabela 5. Atributos da dimensão OrigemDebito.

A dimensão *Natureza* identifica a natureza do tributo sendo cobrado e é utilizada para armazenar o destino específico de cada receita, servindo para separar, por exemplo, receitas provenientes de edificações ou terrenos, apesar de ambas serem obtidas através do IPTU. Deste modo, é imensa a sua importância no contexto do suporte à decisão para o setor tributário. Esta dimensão possui os atributos mostrados na Tabela 6.

Atributo	Significado
id_natureza	Chave primária seqüencial.
natureza	Natureza do débito.
Descricao	Descrição da natureza do débito.

Tabela 6. Atributos da dimensão *Natureza*.

A dimensão *Tempo*, presente na maioria dos *data warehouses*, permite que o aspecto temporal seja incorporado à análise. Sua vinculação com a tabela-fato *ItemDebito* consiste na marcação do tempo (data e hora) em que se oficializou o débito. Seus atributos são listados e explicados na Tabela 7.

Atributo	Significado
id_tempo	Chave primária seqüencial.
data	Data sendo representada.
dia	Valor numérico representando o dia do mês.
mes	Valor numérico referente ao mês.
ano	Valor referente ao ano.
diasemana	Nome do dia da semana.
mesano	Nome do mês.
bimestre trimestre quadrimestre semestre	Valores numéricos representando, respectivamente, bimestre, trimestre, quadrimestre e semestre do ano corrente.
feriado	<i>Flag</i> indicando se a data é um feriado, pois pagamentos podem ser realizados em terminais de auto-atendimento bancário ou <i>sites</i> de <i>home banking</i> .

Tabela 7. Atributos da dimensão *Tempo*.

A tabela-fato *ItemDebito* contém uma linha de um débito, posto que este pode ser composto por mais de um item. O item do débito constitui a informação atômica, ou seja, de mais fina granularidade, no *data warehouse*. Os atributos da tabela-fato *ItemDebito* podem ser vistos na Tabela 8.

Atributo	Significado
id_tempo id_debito id_natureza id_origdeb	Chave primária composta. Individualmente, so campos são chaves estrangeiras para as dimensões Tempo, Debito, Natureza e ItemDebito.
Valoritem	Valor do item do débito.

Tabela 8. Atributos da tabela fato *ItemDebito*.

5.3 CRIAÇÃO E PROCESSAMENTO DO CUBO

Uma vez definido o *data warehouse* a ser utilizado no estudo de caso, o próximo passo é a criação e o processamento do cubo na ferramenta PostGeoOlap, já com as extensões propostas devidamente implementadas.

A configuração de *hardware* e *software* onde foram realizados os estudos de caso é AMD Athlon 3200+ 64 bits 2.2 GHz, 1 GB RAM, 80 GB HD, sistema operacional Windows XP, JVM Sun HotSpot 1.5 Update 10, PostgreSQL 8.0, PostGIS 1.1.6, OpenJUMP 1.0.1.

5.3.1 Conexão com o SGBD

O primeiro passo é a configuração do esquema de banco de dados utilizado para acesso ao *data warehouse*, conforme Figura 56.

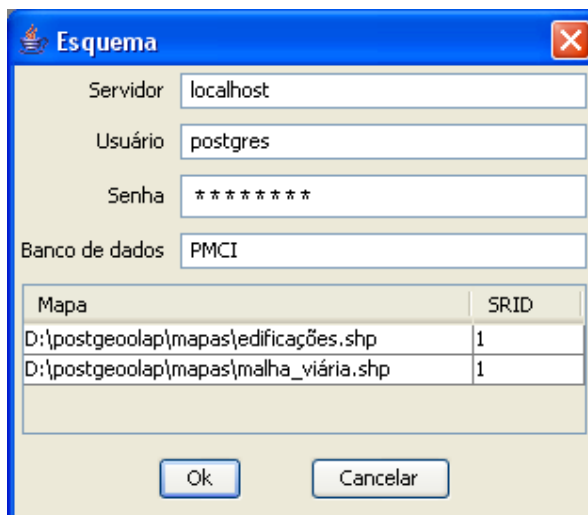


Figura 56. Configuração do esquema.

Após definido o esquema, pode-se realizar a conexão com o *data warehouse* no PostgreSQL, conforme mostrado na Figura 57.

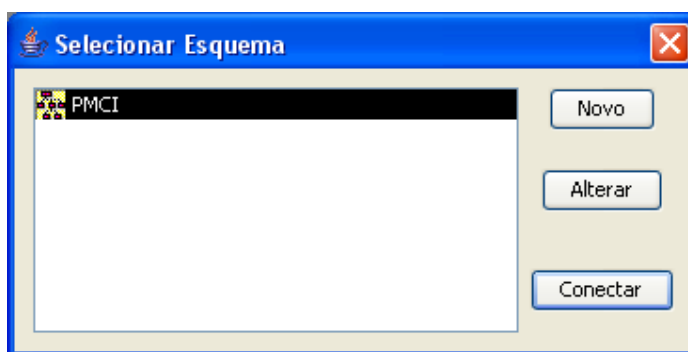


Figura 57. Efetuando conexão com o banco de dados.

5.3.2 Criação do cubo

Após a obtenção da conexão com o banco de dados, o próximo passo é a criação do cubo e a definição da tabela-fato e dos itens numéricos, ou seja, dos atributos nos quais serão aplicadas operações de agregação no processamento do cubo. A Figura 58 mostra a GUI de criação do cubo. Nesta interface é solicitado do usuário que digite a quantidade mínima de linhas para criação de agregações, ou seja, no processamento do cubo apenas serão geradas agregações caso a quantidade de linhas seja igual ou maior do que o número fornecido.

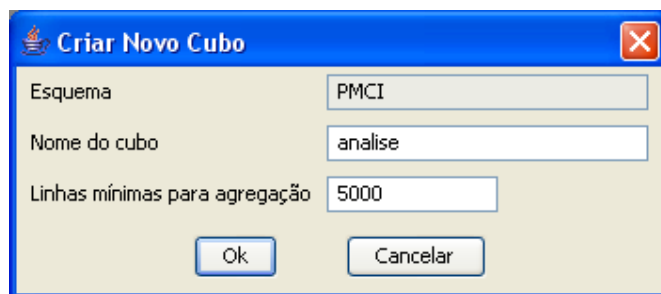


Figura 58. Criação de um novo cubo.

Com o cubo criado, o fluxo da GUI leva à interface para a seleção da tabela-fato e de seus atributos agregáveis, conforme a Figura 59.

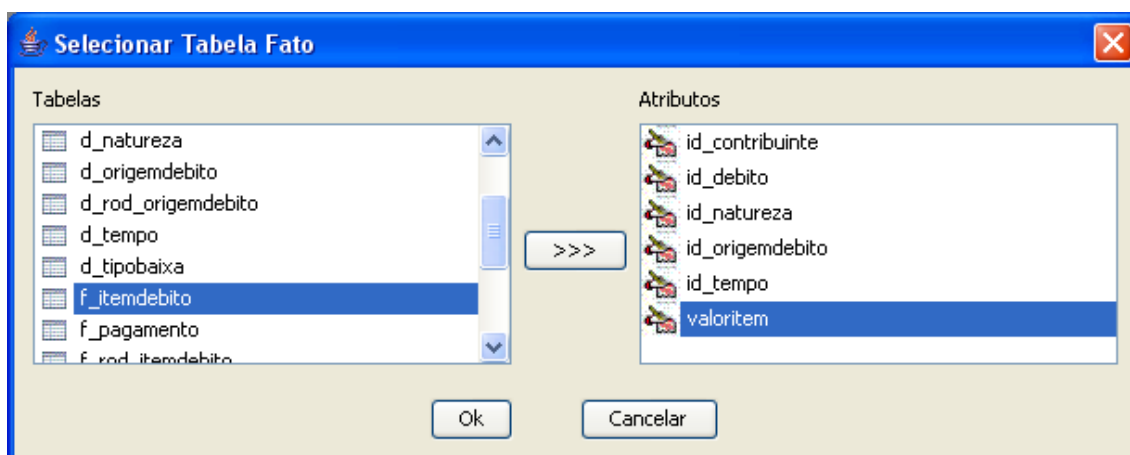


Figura 59. Seleção da tabela-fato e dos atributos agregáveis.

Com a tabela-fato `f_itemdebito` selecionada, aparecem ao lado esquerdo os campos da tabela, para a seleção daquele(s) que será(o) o(s) atributo(s) agregável(is). No caso em questão, apenas o atributo `valoritem` é selecionado. Após definida a tabela-fato, passa-se à seleção das dimensões para o cubo, conforme a Figura 60.

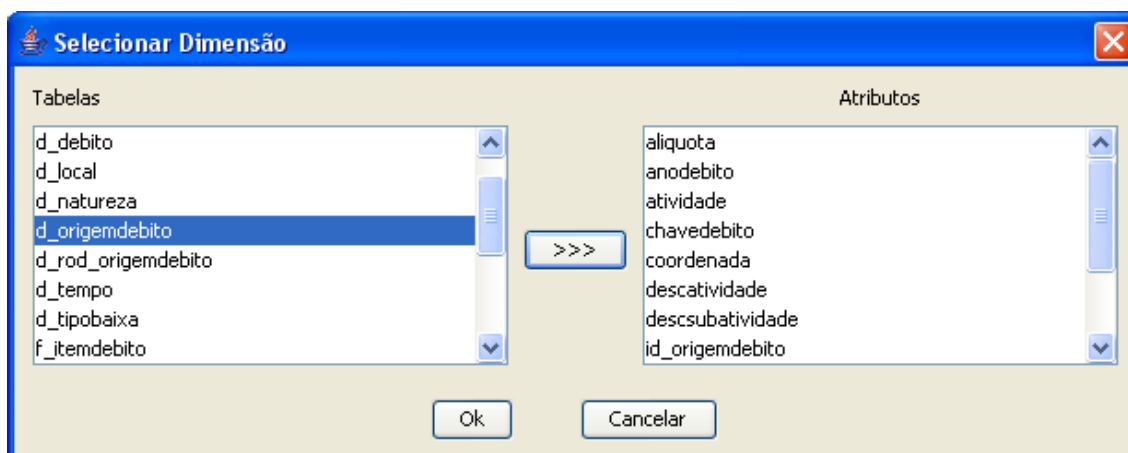


Figura 60. Seleção de uma dimensão.

A seleção das dimensões é bastante simples, bastando selecionar a dimensão desejada e clicar *Ok*. Também é possível ver os atributos da dimensão, clicando no botão no centro do formulário. Após a escolha da dimensão, devem-se atribuir os níveis hierárquicos a cada um dos atributos da dimensão. É preciso também indicar o atributo padrão para cada nível hierárquico, de modo que sirva de fio condutor à análise.

Nome	Nível Hierárquico	Padrão
aliquota	9	<input type="checkbox"/>
anodebito	9	<input type="checkbox"/>
atividade	8	<input checked="" type="checkbox"/>
chavedebito	9	<input type="checkbox"/>
coordenada	9	<input type="checkbox"/>
descatividade	8	<input type="checkbox"/>
descsubatividade	7	<input type="checkbox"/>
id_origemdebito	9	<input type="checkbox"/>
numerodebito	9	<input checked="" type="checkbox"/>
subatividade	7	<input checked="" type="checkbox"/>
tipo	9	<input type="checkbox"/>
tipodesc	9	<input type="checkbox"/>

Figura 61. Seleção da hierarquia para a dimensão OrigemDebito.

A Figura 61 mostra a atribuição dos níveis hierárquicos para a dimensão OrigemDebito. A hierarquia foi definida com números inteiros de 9 para o nível mais agregado e decrescendo-se até o nível menos agregado. Para a dimensão corrente, OrigemDebito, a hierarquia foi dada pelo identificador do débito (atributo `numerodebito`), atribuindo-se o nível 9; pela sub-atividade (atributo `subatividade`), no nível 8; e pela atividade (atributo `atividade`), no nível 7.

Na tabela abaixo, informe o nível hierárquico de cada atributo nesta dimensão, de forma que os de mais alto nível recebam valores menores e os de menor granularidade recebam valores maiores, até um máximo de [9]. Vários atributos podem compartilhar o mesmo nível em uma dimensão. Ex: NomeLoja [9], CNPJLoja [9], BairroLoja [8], CidadeLoja [7].

Nome	Nível Hierárquico	Padrão
anocarne	9	<input type="checkbox"/>
anodebito	9	<input type="checkbox"/>
dataivento	9	<input type="checkbox"/>
datacalculo	9	<input type="checkbox"/>
dataemissao	9	<input type="checkbox"/>
data lancdivat	9	<input type="checkbox"/>
id_debito	9	<input type="checkbox"/>
numeroautoinfr	9	<input type="checkbox"/>
numerocarne	9	<input type="checkbox"/>
numerodebito	9	<input checked="" type="checkbox"/>
parcelas	9	<input type="checkbox"/>
setor	9	<input type="checkbox"/>
siglaorgao	9	<input type="checkbox"/>
situacao	9	<input type="checkbox"/>
situacaodesc	9	<input type="checkbox"/>
tipo	8	<input checked="" type="checkbox"/>
tipodesc	8	<input type="checkbox"/>
tipovencdesc	9	<input type="checkbox"/>
tipovencimento	9	<input type="checkbox"/>
usuario	9	<input type="checkbox"/>
valor1parcela	9	<input type="checkbox"/>
valorparcela	9	<input type="checkbox"/>
valortotal	9	<input type="checkbox"/>

Ok Cancelar

Figura 62. Seleção da hierarquia para a dimensão Debito.

A Figura 62 mostra a seleção dos níveis hierárquicos para a dimensão Debito. A hierarquia foi dada pelo identificador do débito (atributo `numerodebito`), atribuindo-se o nível 9 e pelo tipo do débito (atributo `tipo`), no nível 8.

Na tabela abaixo, informe o nível hierárquico de cada atributo nesta dimensão, de forma que os de mais alto nível recebam valores menores e os de menor granularidade recebam valores maiores, até um máximo de [9]. Vários atributos podem compartilhar o mesmo nível em uma dimensão. Ex: NomeLoja [9], CNPJLoja [9], BairroLoja [8], CidadeLoja [7].

Nome	Nível Hierárquico	Padrão
descricao	9	<input checked="" type="checkbox"/>
id_natureza	9	<input type="checkbox"/>
natureza	9	<input type="checkbox"/>

Ok Cancelar

Figura 63. Seleção da hierarquia para a dimensão Natureza.

A Figura 63 mostra a seleção dos níveis hierárquicos para a dimensão Natureza. A hierarquia foi dada pelo identificador do débito (atributo `numerodebito`), atribuindo-se o nível 9 e pelo tipo do débito (atributo `tipo`), no nível 8.

Na tabela abaixo, informe o nível hierárquico de cada atributo nesta dimensão, de forma que os de mais alto nível recebam valores menores e os de menor granularidade recebam valores maiores, até um máximo de [9]. Vários atributos podem compartilhar o mesmo nível em uma dimensão. Ex: NomeLoja [9], CNPJLoja [9], BairroLoja [8], CidadeLoja [7].

Nome	Nível Hierárquico	Padrão
ano	5	<input checked="" type="checkbox"/>
bimestre	9	<input type="checkbox"/>
data	9	<input type="checkbox"/>
dia	9	<input checked="" type="checkbox"/>
diasemana	9	<input type="checkbox"/>
feriado	9	<input type="checkbox"/>
id_tempo	9	<input type="checkbox"/>
mes	8	<input type="checkbox"/>
mesano	8	<input checked="" type="checkbox"/>
quadrimestre	9	<input type="checkbox"/>
semestre	6	<input checked="" type="checkbox"/>
trimestre	7	<input checked="" type="checkbox"/>

Ok Cancelar

Figura 64. Seleção da hierarquia para a dimensão Tempo.

A Figura 64 mostra a seleção dos níveis hierárquicos para a dimensão Tempo. A hierarquia foi dada pelo dia do mês (atributo `dia`), atribuindo-se o nível 9; pelo nome do mês (atributo `mesano`), no nível 8; pelo trimestre (atributo `trimestre`), no nível 7; pelo semestre

(atributo `semestre`), no nível 6; e, finalmente, o ano (atributo `ano`), no nível 5. É importante notar que os atributos `bimestre` e `quadrimestre` não estão sendo considerados no que se refere a níveis hierárquicos, pois como há no modelo lógico do PostGeoOlap a limitação de apenas uma hierarquia por dimensão, optou-se por considerar trimestre e semestre em detrimento de bimestre e quadrimestre. Em um sistema sem esta limitação, outra hierarquia, além da descrita acima, poderia ser especificada, com `dia→mês→bimestre→quadrimestre→ano`.

5.4 ANÁLISE DOS DADOS

Após criadas as dimensões e processado o cubo, já é possível proceder à análise dos dados, de modo a demonstrar com exemplos práticos as propostas desta dissertação.

Para que seja possível a visualização, é necessário que haja, no mínimo, duas dimensões selecionadas para visualização. Inicialmente, foram escolhidas as dimensões `Natureza` e `Tempo`, conforme mostra a Figura 65.



	CONSERVACAO DE VIAS E LOGRADOUROS...	IMPOSTO PREDIAL	IMPOSTO TERRITORIAL	TAXA DE EXPEDIENTE
1990		12207.0	1136.0	
1991		17195.0	3676.0	
1992		28557.0	8542.0	
1993		39868.0	12409.0	
1994	2705.0	17672.0	4452.0	592.0
1995	493.0	1386.0		30.0
1996		35.0	80.0	
1997	48.0		133.0	14.0
1998			91.0	

Figura 65. Grade multidimensional apresentando `Natureza` x `Tempo`.

É importante notar que os dados são apresentados na forma da visualização multidimensional, ou seja, os valores das dimensões são apresentados nos eixos e os valores das medidas apresentados na tabela central. Os espaços vazios referem-se a dados inexistentes, indicando que não há débitos para uma aquela natureza naquele ano.

Para o estado inicial, têm-se como definições de esquemas para os eixos as Equações 18 e 19.

$$Row_S = \{ [Tempo], [ano] \} \quad (18)$$

$$Column_S = \{ [Natureza], [descricao] \} \quad (19)$$

A partir deste ponto, é possível, por exemplo, obter um *drill-down* em todas as instâncias da dimensão *Tempo*, passando a obter uma interface como a mostrada na Figura 66 e uma equação para o eixo vertical conforme a Equação 20.

<input type="checkbox"/> Mostrar Resultados no Mapa		<input type="button" value="Executar"/>		<input type="button" value="Nova Consulta"/>	
		CONSERVACAO DE VIAS E LOGRADOU...	IMPOSTO PREDIAL	IMPOSTO TERRITORIAL	TAXA DE EXPEDIENTE
1990	1		12115.0	1136.0	
	2		92.0		
1991	1		17076.0	3676.0	
	2		119.0		
1992	1		28434.0	8542.0	
	2		123.0		
1993	1		39792.0	12409.0	
	2		76.0		
1994	1	2705.0	17632.0	4452.0	592.0
	2		40.0		
1995	1	488.0	1372.0		29.0
	2	5.0	14.0		1.0
1996	2		35.0	80.0	
1997	2	48.0		133.0	14.0
1998	1			91.0	

Figura 66. Eixo vertical após operação de *drill-down*.

$$Row_S = \{ [Tempo], [ano, semestre] \} \quad (20)$$

Assim, para cada instância do atributo *ano*, são mostrados grupos de instâncias do atributo *semestre*. Nos anos de 1996, 1997 e 1998 há apenas um semestre subordinado a si. Ocorre que para os semestres faltantes não houve quaisquer incidências de dados, de modo que as representações visuais das instâncias que incorreram na falta de dados foram suprimidas.

Após um *roll-up* na dimensão *Tempo*, que fará com que a grade retorne ao estado que possuía na Figura 65 e os esquemas retornem ao estado mostrado nas Equações 18 e 19, faz-se um novo *drill-down*, mas desta vez apenas sobre a instância “1994”, conforme mostrado na Figura 67.

CONSERVACAO DE VIAS E LOGRADOU...		IMPOSTO PREDIAL	IMPOSTO TERRITORIAL	TAXA DE EXPEDIENTE
1990		12207.0	1136.0	
1991		17195.0	3676.0	
1992		28557.0	8542.0	
1993		39868.0	12409.0	
1994	1	2705.0	17632.0	4452.0
	2	40.0		592.0
1995		35.0	80.0	
1996	48.0		133.0	14.0
1997			91.0	

Figura 67. Aplicação de *drill-down* apenas na instância “1994”.

Após esta operação, permanecem os esquemas dos eixos conforme as Equações 18 e 19. Porém, é instanciado um `PointSchema` associado à instância “1994”, dado na Equação 21, enquanto o esquema geral para o eixo continua como mostrado na Equação 18.

$$PS_{ano=1994} = \{ [Tempo], [ano, semestre] \} \quad (21)$$

Caso o analista do negócio deseje analisar mais de perto outra instância, por exemplo, “1991”, tem-se de imediato o desejado, conforme a Figura 68.

CONSERVACAO DE VIAS E LOGRADOU...		IMPOSTO PREDIAL	IMPOSTO TERRITORIAL	TAXA DE EXPEDIENTE
1990		12207.0	1136.0	
1991	1	17076.0	3676.0	
	2	119.0		
1992		39868.0	12409.0	
1993	2705.0	17632.0	4452.0	592.0
1994	1	40.0		
	2	35.0	80.0	
1995	48.0		133.0	14.0
1996			91.0	

Figura 68. Aplicação de *drill-down* apenas na instância “1991”.

É possível, ainda, aplicar operações de rotação sobre a grade multidimensional, invertendo as dimensões em relação aos eixos. A migração de uma dimensão de um eixo visual a outro tem um custo de meramente redesenhar a tela gráfica, pois se trata da movimentação da dimensão do *axis schema* onde se encontra para o esquema do outro eixo, sem qualquer acesso a bancos de dados, constituindo-se apenas de um rearranjo na estrutura de visualização. O resultado desta operação é mostrado na Figura 69.

	1990	1991		1992	1993	1994		1995
		1	2			1	2	
CONSERVA...					2705.0			48.0
IMPOSTO P...	12207.0	17076.0	119.0	39868.0	17632.0	40.0	35.0	
IMPOSTO T...	1136.0	3676.0		12409.0	4452.0		80.0	133.0
TAXA DE EX...					592.0			14.0

Figura 69. Aplicação da operação de rotação.

Até agora, as visualizações envolveram apenas uma dimensão em cada eixo. Uma aplicação OLAP real envolve, via de regra, mais de uma dimensão por eixo visual. Deste modo, incluiu-se na visualização a dimensão *Debito*. O resultado pode ser visto na Figura 70. A dimensão *Debito* foi colocada no eixo horizontal, subordinada à dimensão *Tempo*.

	1990		1991				1992		1993		1994			
	2	3	1		2		2	3	2	3	1		2	
			2	3	2	3					2	3	2	3
CONSERVA...										2671.0	2671.0	34.0		
IMPOSTO P...	12145.0	62.0	16917.0	159.0	119.0	28349.0	208.0	39508.0	360.0	17460.0	17420.0	212.0	40.0	35.0
IMPOSTO T...	1123.0	13.0	3663.0	13.0		8525.0	17.0	12409.0		4452.0	4452.0			80.0
TAXA DE EX...										583.0	583.0	9.0		

Figura 70. Incluindo a dimensão *Debito* na visualização.

Esta configuração possui os esquemas de eixo dados na Equação 23 e 24 e o *point schema* para os grupos de instâncias encabeçados por “1991” e “1994” dado pela Equação 25.

$$Row_S = \{ [Natureza], [descricao] \} \quad (22)$$

$$Column_S = \{ [Tempo \times Debito], [ano] \times [tipo] \} \quad (23)$$

$$PS_{ano=1991, ano=1994} = \{ [Tempo \times Debito], [ano, semestre] \times [tipo] \} \quad (24)$$

Caso se realizem operações de *roll-up* sobre as instâncias que previamente haviam sofrido *drilling*, e uma rotação entre as dimensões *Tempo* e *Natureza*, o *point schema* é eliminado e a grade retorna à estrutura mostrada na Figura 71.

	CONSERVACAO DE VIAS E LOGRADOUROS P...		IMPOSTO PREDIAL		IMPOSTO TERRITORIAL		TA
	2	3	2	3	2	3	2
1990			12145.0	62.0	1123.0	13.0	
1991			17036.0	159.0	3663.0	13.0	
1992			28349.0	208.0	8525.0	17.0	
1993			39508.0	360.0	12409.0		
1994	2671.0	34.0	17460.0	212.0	4452.0		583.0
1995	493.0		1386.0				30.0
1996			35.0		80.0		
1997	48.0				133.0		14.0
1998					91.0		

Figura 71. Dimensões *Debito*, *Tempo* e *Natureza* sendo visualizadas.

Finalmente, podem-se visualizar os resultados geográficos cabíveis para determinadas instâncias. Por exemplo, é possível escolher uma ou mais células para que os pontos referentes a estas sejam visualizados. Aqui, a abordagem é diversa daquela proposta por Colonese (2004), onde a visualização geográfica se dava pela seleção de atributos, que, caso fossem geográficos, acarretariam na plotagem de seu conteúdo – primitivas geométricas do OpenGIS – no *toolkit* de visualização. Aqui, como se trata de dados sumarizados, a seleção de um ou mais pontos causará a plotagem no visualizador do JUMP do conteúdo dos atributos geográficos individuais envolvidos na sumarização, caso exista, conforme mostra a Figura 72. Pode-se facultar ao usuário escolher qual atributo será o rótulo do atributo geográfico. No caso em questão, é o atributo *numerodebito*.

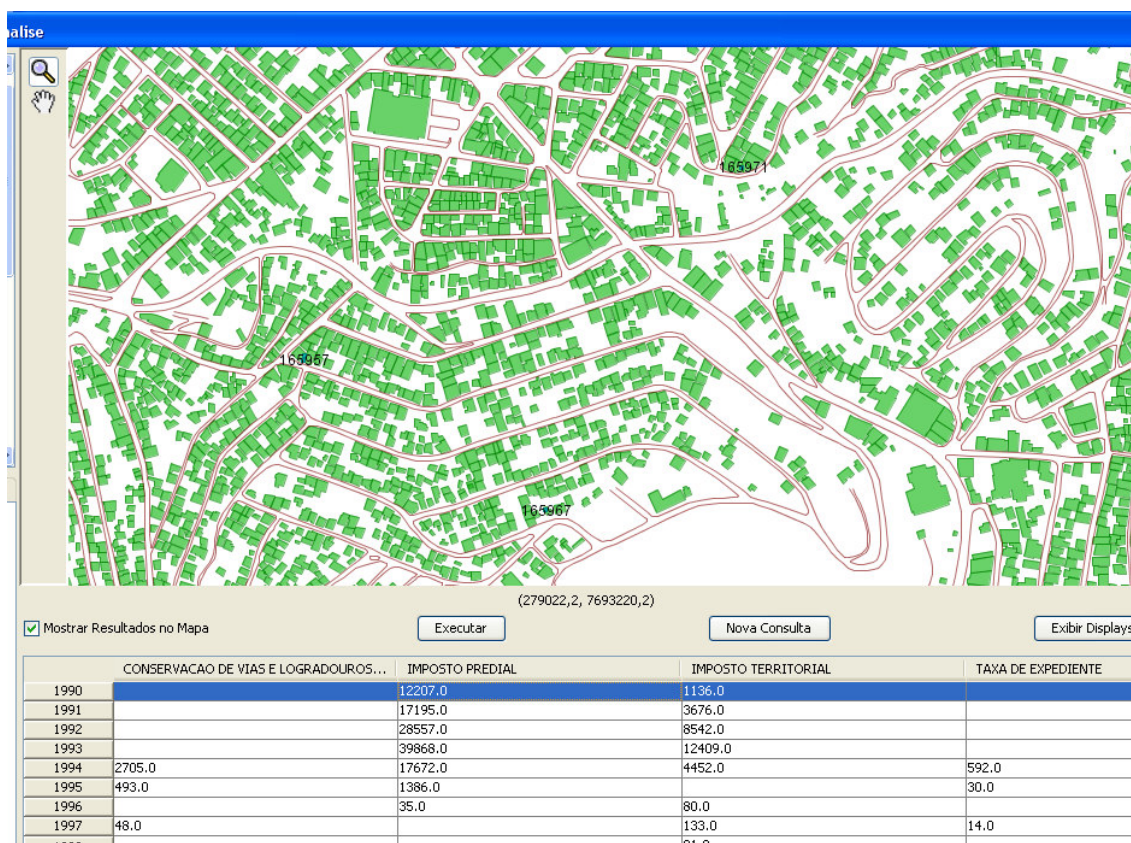


Figura 72. Visualização geográfica no PostGeoOlap.

6 CONCLUSÃO

Cada vez mais cresce, em instituições de diversos tamanhos e finalidades, a necessidade de se extrair, da enorme quantidade de dados gerados pelos sistemas transacionais, informações que possam auxiliar os analistas de negócios destas instituições a melhor geri-las e, assim, obter melhores condições na concorrência com seus pares – no caso de empresas – ou, no caso de instituições públicas, prestar serviços de modo mais abrangente e com maior qualidade. Neste cenário, as tecnologias OLAP e SIG são de fundamental importância para análises que levem em conta, além dos dados propriamente ditos, o componente espacial presente em seu ramo de atividade.

No presente trabalho, abordou-se principalmente o estudo de um modelo de visualização multidimensional para dados tabulares. Apesar de a visualização ser um tópico de fundamental importância em sistemas de suporte à decisão, o tema tem sido pouco explorado. Optou-se, aqui, por estender o CPM (Cube Presentation Model), por considerá-lo uma proposta bastante exequível e funcional. A proposta logra criar uma arquitetura de *software* que permite operar com dados dimensionais para visualização, independentemente de considerações tecnológicas. O uso extensivo de metadados é um elemento fundamental de flexibilização da arquitetura.

Ofereceu-se, neste trabalho, uma arquitetura para que o modelo de visualização seja independente de modelos lógicos, de modo a diminuir o acoplamento intercamadas e facilitar a manutenção. A independência entre os modelos lógico e de visualização constitui um saudável aspecto de projeto do trabalho de Maniatis *et al.* (2003), o qual considerou-se adequado manter

aqui. Estendeu-se, também, o modelo lógico do PostGeoOlap para o suporte a dimensionalidade.

O primeiro passo prático neste trabalho foi a reescrita da ferramenta PostGeoOlap em Java, de modo a torna-la cidadã de primeira classe no mundo do Software Livre. O resultado desta reescrita está funcional e disponível para *download* em <http://postgeoolap.projects.postgresql.org> e uma nova versão com todas as extensões aqui propostas encontra-se em protótipo, cujos *screenshots* puderam ser vistos nas figuras do estudo de caso, e breve também será disponibilizada. Cumpre informar que esta primeira versão foi aceita no PGFoundry⁴⁸, a incubadora de projetos do PostGreSQL.

Finalmente, cabe comentar sobre o PostGeoOlap como uma ferramenta em *software* livre. De acordo com Levesque (2004), um projeto de *software* livre costuma padecer dos seguintes problemas: pobre interface com o usuário, documentação insuficiente, ênfase em funcionalidades demasiado particulares e a elevação dos postulados do mundo *open source* a dogmas inquestionáveis. Tentou-se, na medida do que é possível a um projeto acadêmico, escapar a estes problemas. Em relação à interface, esta dissertação cuida principalmente deste assunto. No que diz respeito à documentação, o estudo de caso apresentado no Capítulo 5 pode dar uma boa orientação a usuários e os demais capítulos são úteis a desenvolvedores. Além disto, procurou-se escrever um código claro, simples e com uso de metadados de documentação no estilo Javadoc⁴⁹. Quanto a funcionalidades demasiado específicas, buscou-se aqui, ao contrário, a generalização. No que tange à sacralização dos postulados do *open-source*, procurou-se utilizar o conceito justamente como é: um meio eficiente e adequado aos dias atuais de democratização e liberdade de uso e disseminação da informação, nada mais nem menos que isto.

Caberá à comunidade – este autor incluso – discutir e implementar melhorias ligadas ao ajuste fino da ferramenta e à eliminação de eventuais *bugs*, bem como de evoluções propostas como trabalhos futuros e outras.

⁴⁸ <http://www.pgfoundry.org>, acesso em 10/03/2007.

⁴⁹ Javadoc é uma convenção, adotada pela maioria das IDEs para Java, que permite a geração de documentação de classes, métodos e propriedades no formato HTML (Hypertext Markup Language) a partir de comentários padronizados no código-fonte. Tem uso generalizado na plataforma Java, sendo utilizada para a geração tanto da documentação da implementações da Sun quanto da imensa maioria dos projetos de *software* livre.

6.1 CONTRIBUIÇÕES

A presente dissertação traz importantes contribuições à área de Sistemas de Informação e ao projeto GeoOlap, entre as quais podem-se citar:

- extensões ao modelo da ferramenta PostGeoOlap (COLONESE, 2004), para que esta possa tratar efetivamente os dados extraídos do *data warehouse* de uma forma analítica, integrando-os aos metadados e estendendo o paradigma analítico às leituras colhidas. Além disto, foram apresentadas melhorias como o suporte a múltiplos mapas, modelagem de atributos e dimensões para suporte a múltiplas dimensões por tabela e definição de hierarquia principal para uma dimensão;
- uma extensão ao *Cube Presentation Model* (CPM) (MANIATIS *et al.*, 2003), concedendo-lhe a capacidade de fornecer ao usuário uma visualização mais flexível do cubo, ao permitir que instâncias individuais de níveis hierárquicos possam sofrer operações de *drilling*;
- implementação em Java do modelo CPM – já adicionadas as extensões propostas nesta dissertação – sendo, ainda implementado como um *engine* de visualização OLAP;
- integração da implementação do CPM estendido à ferramenta PostGeoOlap, fazendo com que a ferramenta tenha suporte interativo a operações OLAP como pivoteamento e *drilling*;
- projeto de uma arquitetura orientada a objetos, extensível e genérica, do tipo *plug-in*, para o completo desacoplamento entre o *toolkit* gráfico utilizado e o modelo de visualização OLAP e entre este e o modelo OLAP subjacente. O projeto proposto, na forma de um *mini-framework*, é implementado em Java. Para validação da proposta, o projeto é instanciado de modo a criar um *plug-in* para o uso do CPM com o *toolkit* gráfico Swing e é criada outra instância para a comunicação com o modelo dimensional do PostGeoOlap;
- para a implementação de uma interface OLAP sobre o Swing foi necessário estender diversos componentes visuais do *toolkit*, pois as classes de componentes padrão estão muito distantes do mínimo necessário para uma GUI OLAP básica. O projeto destes componentes, sempre sob estrita observância da

orientação a objetos, consta na dissertação e seu código-fonte está disponível, já que todo o projeto é aderente aos preceitos do Software Livre;

- finalmente, os altos custos com soluções proprietárias para suporte à decisão, principalmente envolvendo uma conjunção de OLAP e SIG, tornam a ferramenta estendida PostGeoOlap uma alternativa bastante atrativa para pequenas e médias instituições públicas e privadas. O fato de o presente trabalho, com a migração para a plataforma Java, tornar a ferramenta efetivamente multiplataforma, além de utilizar somente componentes distribuídos sob licenças livres, contribui sobremaneira para a democratização do acesso a este tipo de tecnologia.

6.2 SUGESTÕES PARA TRABALHOS FUTUROS

O presente trabalho enseja diversos tópicos para pesquisas futuras no projeto GeoOlap. Como a presente dissertação tem como produto prático imediato uma ferramenta que permite análise OLAP com dados espaciais, há diversas sugestões de funcionalidades que possam vir a ser incorporadas à ferramenta, tanto por projetos acadêmicos quanto por iniciativas da comunidade de Software Livre. É importante notar que não se pretende aqui arrolar sugestões gerais de melhoramentos na ferramenta PostGeoOlap ou no seu núcleo, mas apenas sugestões de trabalhos no que tange a visualizações OLAP e geográfica, daí a ausência de tópicos relacionados a ETL ou desempenho:

- extensão da porção geográfica da ferramenta, com a possibilidade de adição de atributos geográficos nas medidas, permitindo a aplicação de funções de agregação sobre estes atributos, conforme o conceito de SOLAP (RIVEST, BÉDARD e MARCHAND, 2001);
- implementação da capacidade de definição de múltiplas hierarquias para cada uma das dimensões. Isto acarretará em modificações no algoritmo de geração do cubo, descrito em Colonese (2004), pois atributos poderão eventualmente possuir mais de um nível hierárquico em diferentes hierarquias;
- implementação da operação OLAP *drill-across*, permitindo navegação intercubos;

- implementação da operação OLAP *drill-through*, possibilitando a que a ferramenta seja capaz de buscar dados em outros repositórios que não o *data warehouse* subjacente;
- melhoramentos – sempre necessários – na interatividade da tabela de análise de dados, inclusive com consideração do uso de outras tecnologias que não o Swing, como, por exemplo, o SWT⁵⁰ (Standard Widget Toolkit);
- implementação de técnicas de visualização, como Table Lens (RAO e CARD, 1994; PIROLI e RAO, 1996), para suporte automatizado ao destaque de áreas de interesse dentro de uma tabela OLAP, conforme descrito por Maniatis *et al.* (2003[a]);
- extensão do modelo de forma a incluir metadados geográficos para tratamento de aspectos ligados à visualização dos objetos, permitindo que cada usuário adapte características visuais dos atributos geográficos de acordo com a semiologia de sua área;
- extensão da arquitetura de metadados para o suporte a formas customizadas de ordenação. Por exemplo, um atributo `nome_do_mês` não pode ser ordenado alfabeticamente, mas a partir do atributo `número_do_mês`;
- modelagem das análises realizadas pelo usuário, de modo que possam ser persistidas e o usuário não precise recomeçar do zero toda vez que utilizar a ferramenta. Uma proposta de modelagem de operações típicas do mundo relacional sobre objetos é encontrada no trabalho de Barros e Manhães (2006) e pode ser útil para esta tarefa;
- implementação da visualização do PostGeoOlap para outras plataformas, como web – com o auxílio de tecnologias como AJAX (GARRETT, 2005) – ou dispositivos móveis (MANIATIS, 2004). Como o modelo de visualização OLAP proposto e projetado nesta dissertação é independente de tecnologias e mesmo de *toolkits* gráficos, bastaria apenas implementar a visualização propriamente dita e implementar as interfaces de comunicação para as tecnologias em particular.

⁵⁰ <http://www.eclipse.org/swt>, acesso em 13/03/2007.

REFERÊNCIAS BIBLIOGRÁFICAS

- ABELLÓ, Alberto; SAMOS, José; SALTOR, Fèlix. “On Relationships Offering New Drill-Across Possibilities”. In: Proceedings of 5th International Workshop on Data Warehousing and OLAP (DOLAP 2002). McLean/EUA: ACM Press, 2002.
- ABELLÓ, Alberto; SAMOS, José; SALTOR, Fèlix. “YAM²: a Multidimensional Conceptual Model Extending UML”. In: Information Sciences. Elsevier, 2005.
- ALMEIDA, Eduardo. Estudo de Viabilidade de uma Plataforma de Baixo Custo para Data Warehouse. Dissertação de Mestrado. Curitiba: Universidade Federal do Paraná, 2004.
- BARBIERI, Carlos. BI – Business Intelligence: Modelagem e Tecnologia. Rio de Janeiro: Axcel Books, 2001.
- BARROS, Carlos J. B.; MANHÃES, Rodrigo S. Um Framework Estrutural Multicamadas para Desenvolvimento de Aplicações Corporativas. Monografia de Pós-Graduação. Campos dos Goytacazes: Centro Federal de Educação Tecnológica de Campos, 2006.
- BERNARD, Mark; CARTER, James B. “Alan Moore and the Graphic Novel: Confronting the Fourth Dimension”. In: ImageTextT. Volume 1, nº 2. 2004.
- BHARGAVA, Hemant K.; POWER, Daniel J. Decision Support Systems and Web Technologies: A Status Report. Material preparado para a AMCIS 2001, Americas Conference on Information Systems. Disponível em <http://dssresources.com/papers/dsstrackoverview.pdf>, acesso em 12/03/2007. Boston: AMCIS, 2001.

- BIMONTE, S.; TCHOUNIKINE, A.; MIQUEL, M. "Towards a Spatial Multidimensional Model". In: Proceedings of 8th ACM International Workshop on Data Warehousing and OLAP. Nova York: ACM Press, 2005.
- BRACHA, Gilad. Generics in the Java Programming Language. Disponível em <http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf>, acesso em 11/03/2007. Sun Microsystems, 2004.
- BRANCO, Marcelo. Software Livre. Disponível em http://www.vecam.org/edm/article.php3?id_article=55, acesso em 09/08/2005. Enjeux de Mots, 2005.
- BUSCHMANN, Frank *et al.* Pattern-Oriented Software Architecture: Volume 1 – A System of Patterns. Nova York: John Wiley & Sons, 1996.
- CÂMARA, Gilberto; MONTEIRO, Antônio M. V. "Introdução". In: CÂMARA, Gilberto; DAVIS, Clodoveu; MONTEIRO, Antônio M. Introdução à Ciência da Geoinformação. Disponível em <http://www.dpi.inpe.br/gilberto/livro/introd>, acesso em 25/02/2007. São José dos Campos: DPI/INPE, 2001.
- CÂMARA, Gilberto; DAVIS, Clodoveu. "Conceitos Básicos da Ciência da Geoinformação". In: CÂMARA, Gilberto; DAVIS, Clodoveu; MONTEIRO, Antônio M. V. Introdução à Ciência da Geoinformação. Disponível em <http://www.dpi.inpe.br/gilberto/livro/introd>, acesso em 25/02/2007. São José dos Campos: DPI/INPE, 2001.
- CASTELLS, Manuel. A Era da Informação: Economia, Sociedade e Cultura. Volume 1: A Sociedade em Rede. São Paulo: Paz e Terra, 1999.
- CHOI, Jong-Deok *et al.* "Escape Analysis for Java". In: Proceedings of the 14th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'99). Nova York: ACM Press, 1999.
- CHOMSKY, Noam. O Lucro ou as Pessoas?: Neoliberalismo e Ordem Global. Rio de Janeiro: Bertrand Brasil, 2002.
- CODD, Edgar F. "A Relational Model for Large Shared Data Banks". In: Communications of ACM, Vol. 13, Número 6, págs. 377-387. Association for Computer Machinery, 1970.
- CODD, Edgar F. *et al.* Providing OLAP (On-Line Analytical Processing) to User Analyst: An IT Mandate. Disponível em http://dev.hyperion.com/resource_library/white_papers/providing_olap_to_user_analysts.pdf, acesso em 25/02/2007. E. F. Codd Associates, 1993.
- COLONESE, Giovanni. Uma Ferramenta para Integração de Sistemas de Bancos de Dados Analíticos e Geográficos. Dissertação de Mestrado. Campos dos Goytacazes: Universidade Candido Mendes, 2004.

- COLONESE, Giovanni *et al.* “PostGeoOlap: An Open-Source Tool for Decision Support”. In: Anais do II Simpósio Brasileiro de Sistemas de Informação. Florianópolis: SBSI, 2005.
- COLONESE, Giovanni *et al.* “Uma Ferramenta de Baixo Custo para o Desenvolvimento de Sistemas de Suporte à Decisão”. In: Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional. Goiânia: SOBRAPO, 2006.
- COME, Gilberto. Contribuição ao Estudo da Implementação de Data Warehousing: Um Caso no Setor de Telecomunicações. Dissertação de Mestrado. São Paulo: Universidade de São Paulo, 2001.
- DATE, C. J. Introdução a Sistemas de Bancos de Dados. Tradução da 7ª edição. Rio de Janeiro: Campus, 2000.
- DATTA, Anindya; THOMAS, Helen. “The Cube Data Model: a Conceptual Model and Algebra for On-Line Analytical Processing in Data Warehouses”. In: Decision Support Systems. Nº 27. Elsevier, 1999.
- DINIZ, Pedro; RINARD, Martin. “Lock Coarsening: Eliminating Lock Overhead in Automatically Parallelized Object-Based Programs”. In: Proceedings of 9th Workshop on Languages and Compilers for Parallel Computing. San Jose/EUA, 1996.
- DITTRICH, Jens-Peter; KOSSMANN, Donald; KREUTZ, Alexander. “Bridging the Gap between OLAP and SQL”. In: Proceedings of 31st Very Large Data Base (VLDB) Conference. Trondheim/Noruega: VLDB, 2005.
- DOEDERLEIN, Osvaldo. “O Projeto Eclipse: Arquitetura, Subprojetos, Planos e Ferramentas”. In: Java Magazine. Edição 23. Rio de Janeiro: DevMedia, 2005a.
- DOEDERLEIN, Osvaldo. “Performance Aplicada: Explorando Novas Técnicas de Otimização do HotSpot e Mustang”. In: Java Magazine. Edição 30. Rio de Janeiro: DevMedia, 2005b.
- ELMASRI, Ramez; NAVATHE, Shamkant. Sistemas de Banco de Dados: Fundamentos e Aplicações. Rio de Janeiro: LTC, 2002.
- FERREIRA, Ana Cristina F. Um Modelo para Suporte à Integração de Análises Multidimensionais e Espaciais. Dissertação de Mestrado. Rio de Janeiro: UFRJ/IM/NCE, 2001.
- FERREIRA, Ana Cristina F.; CAMPOS, M.L.; TANAKA, Asterio K. “An Architecture for Spatial and Dimensional Analysis Integration”. In: Proceedings of World Multiconference of Systemics, Cybernetics and Informatics. Vol. XIV, Computer Science Engineering, Part II, p. 392-395. Orlando: SCI, 2001.

- FIDALGO, R. N.; TIMES, V. C.; SOUZA, F. F. “GOLAPA: Uma Arquitetura Aberta e Extensível para Integração entre GIS e OLAP”. In: Anais do III Workshop Brasileiro de Geoinformática. Rio de Janeiro: GeoInfo, 2001.
- FORMAN, Ira; FORMAN, Nate. Java Reflection In Action. Greenwich: Manning, 2004.
- FOWLER, Martin. Inversion of Control Containers and the Dependency Injection Pattern. Disponível em <http://www.martinfowler.com/articles/injection.html>, último acesso em 10/03/2007. MartinFowler.com, 2004.
- FOWLER, Martin. UML Essencial: Um Breve Guia para a Linguagem-Padrão de Modelagem de Objetos. 3ª edição. Porto Alegre: Bookman, 2005.
- FREITAS, Marcelo T. M. Desenvolvimento de um Data Warehouse Espacial para o Setor Tributário Municipal Com Uso de Tecnologia Aberta. Dissertação de Mestrado. Campos dos Goytacazes: Universidade Candido Mendes, 2007.
- GALANTE, Alan C. Sistema de Suporte à Decisão no Planejamento Urbano Municipal: Um Estudo de Caso no Município de Macaé. Dissertação de Mestrado. Campos dos Goytacazes: Universidade Candido Mendes, 2004.
- GAMMA, Erich *et al.* Design Patterns: Elements of Reusable Object-Oriented Software. Nova York: Addison-Wesley, 1995.
- GARRETT, Jesse J. Ajax: A New Approach to Web Applications. AdaptivePath, 2005. Disponível em <http://www.adaptivepath.com/publications/essays/archives/000385.php>, acesso em 01/03/2007.
- GATES, Bill. A Estrada do Futuro. São Paulo: Companhia das Letras, 1995.
- GEBHARDT, M.; JARKE, M.; JACOBS, S. “A Toolkit for Negotiation Support Interfaces to Multidimensional Data”. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data. Tucson/EUA: ACM Press, 1997.
- GIOVINAZZO, William. Object-Oriented Data Warehouse Design: Building a Star Schema. Upper Saddle River/EUA: Prentice-Hall, 2000.
- HAN, Jiawei; STEFANOVIC, Nebojsa; KOPERSKI, Krzysztof. “Selective Materialization: An Efficient Method for Spatial Cube Construction”. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 1998.
- HÄTTENSCHWILLER, Pius. “Neuer Anwenderfreundliches Konzept der Entscheidungsunterstützung”. In: Gutes Entscheiden in Wirtschaft, Politik und Gesellschaft. Zurich: vdf Hochschulverlag AG, 1999.

- HELLERSTEIN, Joseph; NAUGHTON, Jeffrey; PFEFFER, Avi. "Generalized Search Trees for Database Systems". In: Proceedings of 21st International Conference on Very Large Databases (VLDB'95), págs. 562-573. Zurique: Morgan Kaufmann, 1995.
- HOLSAPPLE, Clyde W.; WHINSTON, Andrew B. Decision Support Systems: A Knowledge-Based Approach. St. Paul/EUA: West Publishing Company, 1996.
- INMON, William. Como Construir o Data Warehouse. Rio de Janeiro: Campus, 1997.
- KIMBALL, Ralph. The Data Warehouse Toolkit. Nova York: John Wiley & Sons, 1996.
- KIMBALL, Ralph. A Trio of Interesting Snowflakes: Beat Three Common Modeling Challenges with Extensions of the Dimensional Model. Disponível em http://www.intelligententerprise.com/010629/warehouse1_1.jhtml, acesso em 02/03/2007. Intelligent Enterprise Magazine, 2001.
- KIMBALL, Ralph. The Soul of the Data Warehouse, Part Two: The Drilling Across. Disponível em http://www.intelligententerprise.com/030405/606warehouse1_1.jhtml, acesso em 17/03/2007. Intelligent Enterprise Magazine, 2003.
- KIMBALL, Ralph; CASERTA, Joe. The Data Warehouse ETL Toolkit. Nova York: John Wiley and Sons, 2004.
- KIMBALL, Ralph; ROSS, Margy. The Data Warehouse Toolkit: Guia Completo para Modelagem Dimensional. Tradução da segunda edição. Rio de Janeiro: Campus, 2002.
- KOUBA, Zdeněk; MATOUŠEK, Kamil.; MIKŠOVSKÝ, Petr. "On Data Warehouse and GIS Integration". In: Proceedings of the 11th International Conference on Database and Expert Systems. Greenwich: DEXA, 2000.
- KURZ, Robert. O Colapso da Modernização: Da Derrocada do Socialismo de Caserna à Crise da Economia Mundial. São Paulo: Paz e Terra, 1992.
- KURZ, Robert. "L'Apoteosi del Denaro". In: La Fine della Politica a l'Apoteosi del Denaro. Roma: Manifesto Libri, 1997.
- LARMAN, Craig. Utilizando UML e Padrões. 2^a ed. Porto Alegre: Bookman, 2004.
- LECHTENBÖRGER, Jens; VOSSEN, Gottfried. "Multidimensional Normal Forms For Data Warehouse Design". In: Information Systems. Volume 28, Issue 5. Elsevier, 2003.
- LEVESQUE, Michelle. "Fundamental Issues with Open Source Software Development". In: First Monday. Vol. 9, Num. 4. Disponível em http://www.firstmonday.dk/issues/issue9_4/levesque/index.html, acesso em 02/03/2007. First Monday, 2004.
- LEWIS, J. P.; NEUMANN, Ulrich. Performance of Java versus C++. Disponível em <http://www.idiom.com/~zilla/Computer/javaCbenchmark.html>, acesso em 13/03/2007. Los

- Angeles: Computer Graphics and Immersive Technology Lab/University of Southern California, 2004.
- LISBOA FILHO, Jurgurta; IOCHPE, Cirano. "Specifying Analysis Patterns for Geographical Databases on the Basis of a Conceptual Framework". In: Proceedings of 7th ACM International Symposium on Advances in Geographic Information Systems. Nova York: ACM Press, 1999.
- MACHADO, Hugo B. Curso de Direito Tributário. 5^a ed. Rio de Janeiro: Forense, 1992.
- MALINOWSKI, Elzbieta; ZIMÁNYI, Esteban. "Representing Spatiality in a Conceptual Multidimensional Model". In: Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems. New York: ACM Press, 2004.
- MANHÃES, Rodrigo S. *et al.* "GeoOlap: An Integrated Approach for Decision Support". In: Proceedings of Second IFIP TC8 International Conference on Research and Practical Issues of Enterprise Information Systems. Pequim: Springer-Verlag, 2007.
- MANIATIS, Andreas. "OLAP Presentation Modeling with UML and XML". In: Proceedings of the 1st Balkan Conference in Informatics. Tessalônica: BCI, 2003.
- MANIATIS, Andreas *et al.* "CPM: A Cube Presentation Model for OLAP". In: Proceedings of 5th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2003). Praga: DaWaK, 2003.
- MANIATIS, Andreas *et al.* "Advanced Visualization for OLAP". In: Proceedings of 6th International Workshop on Data Warehousing and OLAP (DOLAP 2003). Nova Orleans: ACM Press, 2003(a).
- MANIATIS, Andreas. "The Case for Mobile OLAP". In: Proceedings of 9th International Conference on Extending Database Technology (EDBT 2004). Heraklion/Grécia: EDBT, 2004.
- MARINACCI, Joshua. Swing Has Failed. What Can We Do?. Disponível em http://weblogs.java.net/blog/joshy/archive/2003/11/swing_has_faile.html, acesso em 30/01/2007. Java.net, 2003.
- MAXWELL, Christine; GUTOWITZ, Howard. Data Mining Solutions and the Establishment of a Data Warehouse: Corporate Nirvana for the 21st Century? Disponível em http://www.firstmonday.dk/issues/issue2_5/maxwell/, acesso em 02/03/2007. First Monday: 1997.

- OPENGIS Consortium Inc. OpenGIS Simple Features Specification for SQL: Revision 1.1. OpenGIS Project Document 99-049. Disponível em <http://www.opengeospatial.org/docs/99-049.pdf>, acesso em 11/08/2005. OpenGIS Consortium, 1999.
- PAPADIAS, Dimitris *et al.* “Efficient OLAP Operations in Spatial Data Warehouses”. In: Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases, págs. 443-459. ACM Records, 2001.
- PARENTE, Daniel. Dividir para Conquistar ou Conquistar para Dividir?. Disponível em http://www.dwbrasil.com.br/html/artdw_20030620.html, acesso em 02/03/2007. DwBrasil, 2003.
- PESTANA, Gabriel; SILVA, Miguel M. “Multidimensional Modeling based on Spatial, Temporal and Spatio-Temporal Stereotypes”. In: Proceedings of 25th Annual ESRI International User Conference (UC). San Diego: ESRI, 2005.
- PIROLI, Peter; RAO, Ramana. “Table Lens as a Tool for Making Sense of Data”. In: Proceedings of the Advanced Visual Interfaces (AVI-96) Workshop. Gubbio/Itália: AVI, 1996.
- POWER, Daniel J. A Brief History of Decision Support Systems. Disponível em <http://DSSResources.COM/history/dsshhistory.html>, acesso em 30/01/2007. DSSResources.com, 2003.
- RAGO, Luzia M.; MOREIRA, Eduardo F. P. O Que é Taylorismo. Rio de Janeiro: Brasiliense, 1984.
- RAO, Ramana; CARD, Stuart K. “The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information”. In: Proceedings of the Conference on Human Factors in Computing Systems (ACM SIGCHI '94). ACM, 1994.
- RAO, Gururajan; TUROFF, Murray. “A Hypermedia-Based Group Decision Support System to Support Collaborative Medical Decision-Making”. In: Decision Support Systems. Volume 30, Issue 2. Elsevier, 2000.
- REINHOLTZ, Kirk. “Java Will Be Faster Than C++”. In: ACM SIGPLAN Notices. Volume 35, Issue 2. Nova York: ACM Press, 2000.
- RIVEST, Sonia; BÉDARD, Yvan; MARCHAND, Pierre. “Toward Better Support for Spatial Decision Making: Defining the Characteristics of Spatial On-Line Analytical Processing (SOLAP)”. In: Geomatica. Volume 55, nº 4. 2001.
- ROBINSON, Matthew; VOROBIEV, Pavel. Swing. 2ª ed. Greenwich: Manning, 2003.

- ROCHA, Cezar H. B. Geoprocessamento: Tecnologia Transdisciplinar. 2ª ed. Juiz de Fora: Ed. do Autor, 2002.
- SASSEN, Saskia. As Cidades na Economia Mundial. São Paulo: Studio Nobel, 1998.
- SHEKHAR, Shashi *et al.* “MapCube: A Visualization Tool for Spatial Data Warehouse”. In: MILLER, Harvey; HAN, Jiawei (orgs.). Geographic Data Mining and Knowledge Discovery. Taylor and Francis, 2001.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de Banco de Dados. São Paulo: Makron Books, 1999.
- SILVEIRA, Sérgio A. Inclusão Digital, Software Livre e Globalização Contra-Hegemônica. Disponível em http://www.softwarelivre.gov.br/softwarelivre/artigos/artigo_02, acesso em 08/03/2007. SoftwareLivre.gov.br, 2003.
- SOUZA, Bruno. “A Luta pelo Java Livre: Liberdade com Dentes Afiados”. In: Java Magazine. Edição 19. Rio de Janeiro: DevMedia, 2004.
- STALLMAN, Richard. Free But Shackled: the Java Trap. Disponível em <http://www.gnu.org/philosophy/java-trap.html>, acesso em 14/03/2007. Free Software Foundation, 2004.
- STEFANOVIC, Nebojsa. Design and Implementation of On-Line Analytical Processing (OLAP) of Spatial Data. Dissertação de Mestrado. Belgrado: Universidade de Belgrado, 1997.
- STROUSTRUP, Bjarne. The C++ Programming Language. 3ª ed. Nova York: Addison-Wesley, 1997.
- THOMSEN, Erik. OLAP Solutions: Building Multidimensional Information Systems. 2ª edição. Nova York: John Wiley & Sons, 2002.
- TRUJILLO, Juan *et al.* “Designing Data Warehouses with OO Conceptual Models”. In: IEEE Computer. p. 66-75. IEEE, 2001.
- VASSILIADIS, Panos; SELLIS, Timos. “A Survey on Logical Models for OLAP Databases”. In: ACM SIGMOD Record, Volume 28, Issue 4. Nova York: ACM Press, 1999.
- VASSILIADIS, Panos; SKIADOPOULOS, Spiros. “Modeling and Optimization Issues for Multidimensional Databases”. In: Proceedings of the 12th Conference on Advanced Information Systems Engineering (CAiSE-00). Estocolmo: CAiSE, 2000.
- WITTGENSTEIN, Ludwig. Tractatus Logico-Philosophicus. São Paulo: Edusp, 2001.