UNIVERSIDADE FEDERAL DE SANTA CATARINA

Desenvolvendo um Middleware para a Integração de Servidores OLAP e Servidores de Mapa na Web

Carlos Eduardo da Costa

José Filipe Neis

Florianópolis – SC

2007 / 1

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

Desenvolvendo um Middleware para a Integração de Servidores OLAP e Servidores de Mapa na Web

Carlos Eduardo da Costa

José Filipe Neis

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Florianópolis – SC

2007 / 1

Carlos Eduardo da Costa

José Filipe Neis

Desenvolvendo um Middleware para a Integração de Servidores OLAP e Servidores de Mapa na Web

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Orientador (a): Renato Fileto

Banca examinadora

Roberto Carlos dos Santos Pacheco

Ronaldo dos Santos Mello

Índice

Resumo

A utilização de consultas OLAP (Processamento Analítico On-Line) em conjunto com técnicas de data warehousing para análise de informações tornou-se bem difundida nos últimos anos no mundo dos negócios. Como grande parte das informações nestas consultas referem-se direta ou indiretamente ao aspecto geográfico, surge uma oportunidade de enriquecer as análises agregando ao resultado sua visualização em mapas. Neste contexto, nos últimos anos surgiram algumas propostas relacionadas à integração de ferramentas OLAP com Sistemas de Informação Geográfica (SIG). No entanto, há muitas barreiras para a integração dessas duas tecnologias já razoavelmente consolidadas. Atualmente, o número de propostas conceituais de integração supera em muito o de propostas práticas. Os resultados alcançados por este trabalho incluem a proposta de uma arquitetura baseada no uso de componentes e padrões abertos para a integração OLAP x SIG, além de componentes desenvolvidos especificamente para a realização desta integração. Além disso, é também apresentado um componente desenvolvido neste trabalho para a geração de mapas dinamicamente a partir de dados provenientes de um servidor OLAP. Este trabalho difere-se dos demais trabalhos relacionados por propor uma arquitetura facilmente adaptável ao atual cenário de ferramentas OLAP, não requisitando mudanças bruscas na arquitetura e/ou na representação dos dados OLAP.

Abstract

The use of OLAP (On-Line Analytical Processing) with data warehousing techniques for information analysis has become widespread in the last few years for business applications. As a great part of the information used on these queries refer to directly or indirectly to geographical data, one can enrich the analysis by aggregating maps to the query results. In the last few years, some proposals about the integration of OLAP with Geographical Information Systems (GIS) have arisen. However, there are many challenges to integrate these two consolidated technologies. Nowadays, the number of conceptual proposals is much larger than practical ones. The results achieved by this work include the proposal of an architecture based on components and open standards for the integration of OLAP and GIS, with components developed specifically for integration purpose. It also presents a component developed in this work to dynamically create maps from data retrieved from the OLAP server. This work differs from related ones by presenting an architecture adapted to the current OLAP tools scenario, avoiding considerable changes on the current OLAP architecture or data representation.

1. INTRODUÇÃO

1.1 Apresentação

Atualmente a utilização ferramentas OLAP (Processamento Analítico On-Line) no ambiente de negócios é tida como consenso no que se refere a sistemas de suporte à decisão. A possibilidade de agregar a visualização destas análises em mapas pode trazer um interessante diferencial e enriquecimento das possibilidades de análise de informação. Porém, esta integração entre ferramentas OLAP e SIG (Sistemas de Informação Geográfica), encontra diversas barreiras ao longo do processo, caracterizando-se como um desafio de pesquisa e desenvolvimento.

Entre as barreiras encontradas neste processo de integração de tecnologias, destaca-se o fato que, apesar de cerca de 80% dos dados de um OLAP possuírem um componente de localização geográfica, esses dados estão usualmente representados de forma alfanumérica. Outro grande obstáculo é que tanto OLAP quanto SIG são tecnologias já consolidadas no mercado.

1.2 Motivação

Em virtude das barreiras supracitadas, o número de modelos conceituais propostos na bibliografia é desproporcionalmente grande, quando comparado ao de propostas envolvendo implementação. Enquanto modelos conceituais representam, por seus objetivos, uma estrutura mais flexível, a falta de modelos lógicos que suportem os mesmos torna sua utilização no curto prazo inviável.

Esta falta de propostas práticas motivou o desenvolvimento de uma solução que busque integrar estas diferentes tecnologias utilizando padrões adotados pelo mercado.

1.3 Objetivos

A arquitetura proposta neste trabalho busca viabilizar a visualização de dados multidimensionais em conjunto com dados geográficos de maneira transparente ao usuário. Para isso, é necessário ter-se em mente a abstração da complexidade naturalmente imposta pelo ambiente, já que tratam-se de duas fontes de dados distintas, tanto no que diz respeito ao conteúdo quanto ao tipo de dados que armazenam.

Propomos uma extensão de uma arquitetura OLAP típica, acrescentando componentes para armazenamento, manipulação e apresentação de dados geográficos. Além disso, é introduzido um componente (*Middleware*) responsável pela integração entre as tecnologias OLAP e SIG, peça chave na proposta da arquitetura e foco de desenvolvimento deste trabalho.

1.4 Justificação

O desenvolvimento de um *middleware* compondo uma arquitetura baseada em padrões utilizados pelo mercado gera uma estrutura na qual módulos de diferentes fornecedores poderão se acoplar, desde que sigam as mesmas especificações de interfaces de programação. Novos trabalhos poderão surgir a partir desta iniciativa e suprir as necessidades do mercado, que busca mais soluções práticas de integração de OLAP X SIG.

1.5 Organização dos capítulos

Este trabalho encontra-se organizado em seis capítulos. Os demais capítulos estão dispostos da seguinte forma. O capítulo 2 apresenta os conceitos fundamentais presentes neste trabalho, os quais incluem desde o processo de construção de uma aplicação OLAP até tecnologias e padrões utilizados em projetos SIG. O capítulo 3 aborda a dificuldade de integração das tecnologias OLAP e SIG, expondo as barreiras impostas pelos padrões atuais. Neste capítulo são citados também os trabalhos relacionados. O capítulo 4 apresenta a arquitetura proposta, com todo o seu fluxo de processamento das informações. O capítulo 5 descreve detalhes do nosso projeto, através de um exemplo prático onde é possível observar as tecnologias utilizadas

Finalmente, o capítulo 6 apresenta as conclusões e sugestões de trabalhos futuros.

2. FUNDAMENTOS

2.1 Sistemas de Apoio à Decisão

A utilização de sistemas de apoio à decisão tornou-se peça chave para a obtenção da tão disputada vantagem competitiva no mundo dos negócios. Prova disto é a ascensão de data warehousing e processamento analítico on-line (OLAP) - técnicas essenciais do suporte à decisão - e foco da indústria de banco de dados nos últimos anos [Coliat, G. 1996.].

2.1.1 Data warehouse

Um data warehouse [Inmon,, 1996] é uma base de dados projetada e implementada especificamente para o apoio à tomada de decisão. O processo de data warehousing – desenvolvimento e uso de data warehouses - envolve um conjunto de tecnologias que visam permitir a integração e a análise de informação de diversas fontes de dados, possibilitando a tomada de decisões baseada em análises de informações consolidadas, de maneira mais rápida e segura. Ao contrário dos sistemas transacionais – que têm como principal preocupação a execução de tarefas isoladas do fluxo de negócio – os data warehouses devem manter dados históricos, o que os torna naturalmente maiores com o tempo [Chaudhuri, 1997].

A utilização de data warehouses é muito maior na busca de padrões de distribuição de dados do que na análise de registros específicos. Suas consultas, portanto, tendem a ser mais complexas e demoradas que as consultas dos sistemas transacionais, haja vista a quantidade de dados e agrupamentos que as

consultas em data warehouses envolvem. Por este motivo, data warehouses são normalmente mantidos em bancos de dados exclusivos e utilizam modelos multidimensionais para facilitar as consultas [Chaudhuri,1997].

2.1.2 Modelo Dimensional de Dados

Profissionais da área de banco de dados buscam sanar grande parte dos problemas encontrados na armazenagem de grandes quantidades de dados ao longo dos anos utilizando modelos dimensionais. Apesar de seu projeto e utilização serem mais complexos, o modelo dimensional possui desempenho superior para consultar grandes quantidades de dados, se comparado ao modelo entidade-relacionamento (ER).

O modelo dimensional organiza as informações de medidas (e.g., quantidade e valor das vendas de uma rede de lojas, quantidade e notas dos alunos de uma instituição) segundo diversas dimensões (e.g., tempo, espaço geográfico e características dos produtos ou pessoas envolvidos nas análises). Cada uma dessas dimensões possui uma hierarquia que permite agrupamento das medidas analisadas (e.g., soma, média, mínimo e máximo das vendas, do número ou das notas dos alunos) em diversos níveis de agregação. Na dimensão tempo, por exemplo, os dados podem ser agrupados por ano, semestre, mês e na dimensão espaço segundo a hierarquia de partições político-administrativas (país, região, estado, região do estado, município, distrito).

Um esquema dimensional pode ser expresso no modelo relacional utilizando modelagem no estilo estrela ou floco de neve. Nesses esquemas as medidas são colocadas em tabelas fato, que ficam no centro, ligadas a tabelas de

dimensões, contendo os dados descritivos dessas dimensões. Na modelagem estrela, as tabelas de dimensão ligam-se às tabelas fato através de uma única junção (join) [Kimball,1996]. Neste caso, dados descritivos de diferentes níveis de agregação são modelados como atributos da tabela implementando a dimensão. Isso resulta em redundâncias nessas tabelas, que ficam não normalizadas. Na modelagem floco de neve, por outro lado, os dados descritivos de cada dimensão podem ser colocados em tabelas encadeadas por relacionamentos (e.g., tabela de País, relacionada à tabela Região, que por sua vez relaciona-se com Estado e assim por diante).

A Figura 1 ilustra um esquema dimensional no formato estrela para a análise de informações dos candidatos e provas do vestibular da Universidade Federal de Santa Catarina (UFSC). A tabela fato desse esquema contém as medidas e as chaves estrangeiras para as demais tabelas de dimensões. No modelo dimensional, todas as medidas relativas ao contexto analisado devem ficar na tabela de fato. As tabelas de dimensão, por sua vez, são responsáveis por armazenar as descrições textuais do negócio. As dimensões sexo, uf (unidade da federação de origem ou se é estrangeiro) e faixa de renda referem-se a características dos candidatos. A dimensão curso indica a área, o centro e o curso para o qual o candidato está prestando vestibular. A dimensão língua refere-se à escolha do candidato para a prova de idioma estrangeiro. A dimensão evento descreve os diferentes concursos vestibulares existentes no DW. A tabela de fato possui duas medidas: quantidade de inscritos e quantidade de aprovados. Com estas medidas é possível, através de uma ferramenta OLAP, realizar agregações

e cálculos diferenciados através dos diferentes atributos disponíveis nas dimensões, descritos anteriormente.

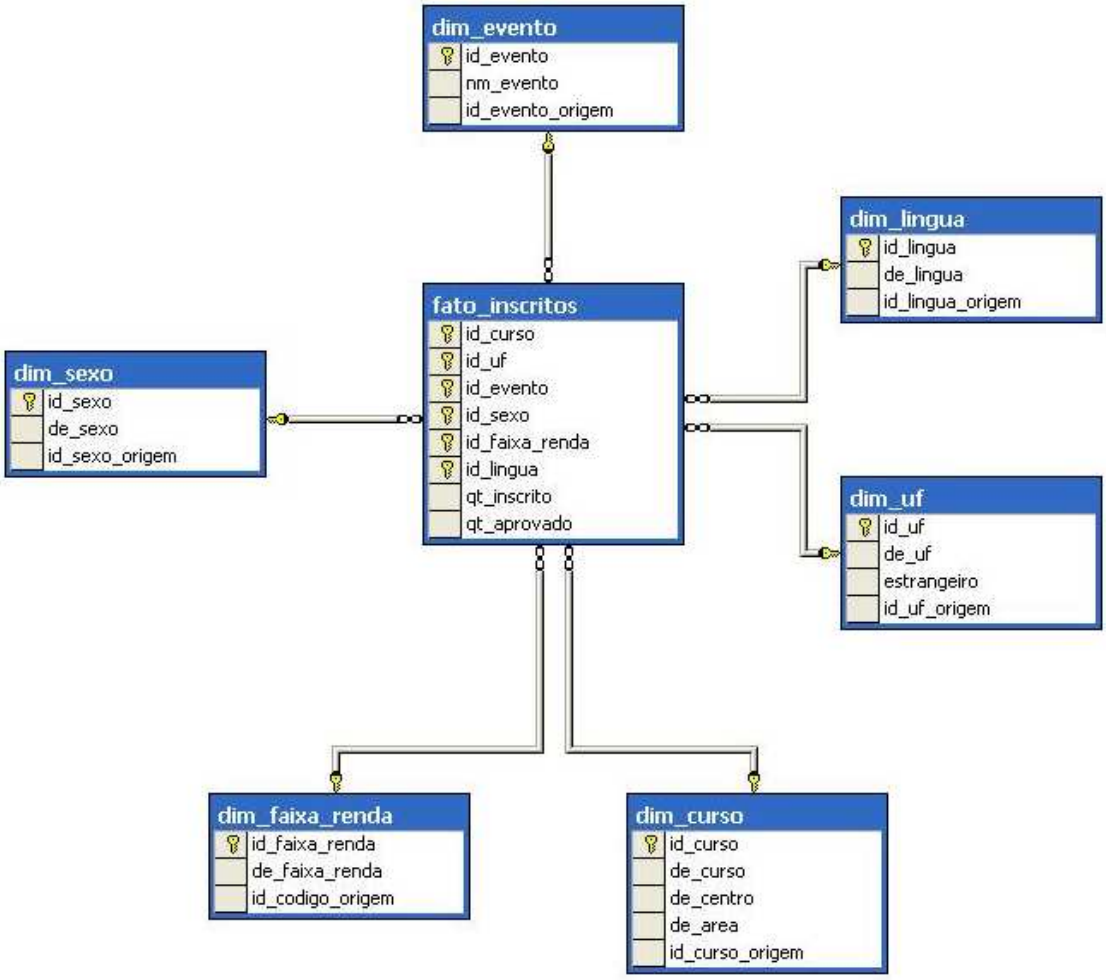

Figura 1: Esquema dimensional vestibular da UFSC

2.1.3 Processamento Analítico On-Line (OLAP)

OLAP, sigla em inglês para processamento analítico on-line, é uma categoria de processamento cujo objetivo é prover diversas visões dos dados de um data warehouse, em diferentes níveis de detalhamento e diferentes focos, buscando refletir a dimensionalidade dos mesmos do ponto de vista do usuário

[OLAP Council, 1997]. Uma ferramenta OLAP deve possibilitar o acesso eficiente a essas diferentes visões de dados agregados.

Segundo [Colliat, 1996], uma ferramenta deve atender a um conjunto de critérios para que possa ser considerada OLAP. Com base nestes critérios, é possível listar as principais características de ferramentas OLAP:

- **Nível Base Sumarizado** – O nível base dos dados de uma ferramenta OLAP deve ser constituído de cálculos sobre os dados de entrada (e.g., número de alunos aprovados por ano em determinado estado). Medidas de desempenho e variação das ocorrências de valores também podem ser efetuadas sobre os dados.
- **Dados Históricos e Projeções** – Ferramentas OLAP podem e devem, na grande maioria dos casos, possuir dados históricos e atuais, além de projeções. Devido a esta característica, a sumarização dos dados mediante as agregações desejadas deve resultar em uma massa de dados de médio a grande porte.
- **Visualizações Multidimensionais** – Por trabalharem sobre o modelo dimensional de dados, ferramentas OLAP devem possibilitar a agregação dos dados nos diversos níveis das hierarquias de dados descritivos das diferentes dimensões (e.g., aprovados por estado, por região, por faixa etária, etc.). Além disso, as ferramentas OLAP possibilitam navegar na hierarquia dos dados (*roll-up e drill-down*).

- **Constante Atualização e Flexibilidade** – Por trabalharem com modelos de negócio em constante atualização, ferramentas OLAP devem permitir mudanças constantes em suas estruturas e análises. Para isso, também é necessário que consultas ad-hock sejam processadas em tempos hábeis (na casa dos segundos).

A utilização de sistemas OLAP acontece de maneira complementar ao uso de data warehouses. Enquanto data warehouses armazenam e gerenciam os dados, ferramentas OLAP são capazes de transformar os mesmos em informações estratégicas para o negócio, abstraindo do usuário a complexidade de consultas e da representação física dos dados [OLAP Council, 1997].

2.1.4 Arquitetura OLAP

O papel de um servidor OLAP consiste no processamento dos dados armazenados em um modelo dimensional baseado em consultas efetuadas por um usuário. Desta maneira, pode-se dizer que a utilização de um servidor OLAP está condicionada à existência de uma arquitetura de apoio à decisão, complementada por um data warehouse e uma ferramenta para criação e visualização do resultado das consultas.

A Figura 2 ilustra uma arquitetura típica para a implementação de aplicações OLAP, composta por três camadas principais: camada de apresentação, camada de processamento e camada de dados. Cada uma destas camadas possui um ou mais componentes responsáveis por prover suas funcionalidades. Os componentes de cada camada ou mesmo a decomposição de uma camada em outras podem variar dependendo da solução OLAP adotada.

A **camada de interface** é responsável por disponibilizar ao usuário a informação processada pelas demais camadas, bem como prover maneiras do mesmo informar ao sistema quais informações deseja visualizar, em que foco e em qual granularidade. As informações retornadas pela camada de processamento podem ser exibidas de diversas formas: desde uma imagem de um gráfico até uma tabela dinâmica para exibição de dados multidimensionais.

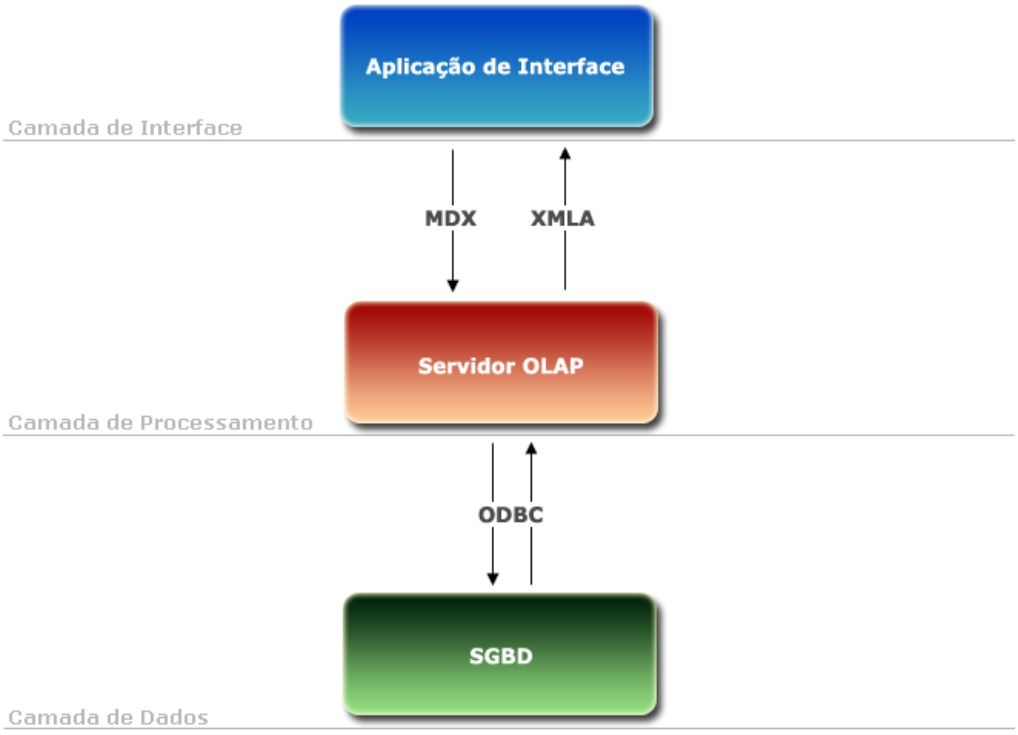

Figura 2. Arquitetura OLAP Típica

A **camada de processamento** é a camada que hospeda o servidor OLAP. Esta camada tem como principal função mediar a exibição ao usuário dos dados

armazenados, efetuando o processamento das consultas submetidas na camada de interface. O processamento dessas consultas exige a interpretação das mesmas e, muitas vezes, a utilização de um conjunto de agregações ou acesso a agregações previamente computadas na fase de pré-processamento, para viabilizar a execução em tempo hábil.

A **camada de dados** é composta basicamente por um sistema gerenciador de banco de dados (SGBD). Apesar dos dados serem vistos no modelo dimensional, seu armazenamento em meio físico pode ser realizado de maneiras diferenciadas. Enquanto MOLAP (*multidimensional OLAP*) armazena os dados em estruturas otimizadas para acesso multidimensional, ROLAP (*relational OLAP*) transforma cada registro da tabela de fatos em uma linha. Existe ainda um terceiro modelo, o HOLAP (*hybrid OLAP*), que busca otimizar o acesso aos dados, mesclando características MOLAP e ROLAP [Chaudhuri,1997].

Apesar de não existirem padrões formais que descrevam os padrões e protocolos a serem utilizados na comunicação entre os componentes da arquitetura de um sistema OLAP, os principais fornecedores de soluções OLAP vêm utilizando um conjunto comum de protocolos nos últimos anos, constituindo um padrão informal de mercado.

O ponto chave da integração numa arquitetura OLAP típica está no fato das camadas de interface e processamento compartilharem um “protocolo” comum para definição do que o usuário deseja visualizar (MDX) e outro para o retorno dos dados encontrados na camada de dados e processados pela camada de processamento OLAP (XMLA). O acesso à camada de dados, por sua vez, é

facilitado pela utilização de drivers ODBC, padrão de conexão entre sistemas de gerenciamento de bancos de dados amplamente utilizado.

A próxima seção descreve em detalhes a linguagem MDX de consulta a servidores OLAP e o padrão XMLA utilizado para retornar descrições dos cubos de dados e dos resultados de consultas gerando visões dos cubos de dados para o módulo de interface.

2.1.5 Padrões para conexão a servidores OLAP

MDX

MDX [Microsoft Corporation, 2007], acrônimo para ***m****ulti****d****imensional e****x****pressions*, é uma linguagem que foi criada pela Microsoft em 1998 com o intuito de estabelecer um padrão que simplificasse o acesso a fontes de dados multidimensionais. Com o passar do tempo, várias empresas adotaram a sintaxe como padrão para suas aplicações, tendência que continua em vigor atualmente.

A sintaxe da linguagem MDX assemelha-se bastante à linguagem SQL padrão. Seus resultados e capacidades, porém, são diferentes em diversos sentidos. A maior e provavelmente mais importante diferença é a capacidade da linguagem MDX consultar fontes de dados multidimensionais. Enquanto a linguagem SQL foi construída para trabalhar com duas dimensões (linhas e colunas), a linguagem MDX permite trabalhar com uma, duas, três ou mais dimensões. Por este motivo, a visualização dos resultados obtidos com a execução de uma consulta MDX nem sempre é tão facilmente interpretável.

A linguagem MDX, assim como a SQL, é composta por uma estrutura básica de cláusulas. Em uma consulta MDX podemos identificar as seguintes estruturas:

- **SELECT** – cláusula responsável por definir os valores das dimensões que constituirão as arestas da visão do cubo multidimensional a ser retornado como resposta.
- **FROM** – cláusula responsável por definir a fonte de dados multidimensional a ser utilizada.
- **WHERE** – cláusula opcional. Quando utilizada, define qual dimensão ou membro será utilizado como foco na consulta (*slice*).

Sintaxe básica MDX:

```
SELECT <set> ON COLUMNS,
       <set> ON ROWS
       FROM [<cube_name>]
```

Figura 3. Sintaxe básica MDX

A Figura 4 mostra uma consulta em MDX que retorna o total de vendas e custos durante os anos de 1997 e 1998 individualmente para todas as lojas dos Estados Unidos da América, incluindo todos os produtos vendidos. A informação é disposta em uma grade (tabela) bi-dimensional, com as vendas e os custos

(chamados “measures” neste DW) nas linhas e os anos (1997 e 1998) nas colunas. [XMLA.org, 2007].

Figura 4. Estrutura detalhada de uma consulta MDX

XMLA

Apresentado pela primeira vez em meados de 2001, *XML for Analysys*, ou somente XMLA [XMLA.org, 2007] é um padrão que permite aplicações cliente comunicarem-se com fontes de dados multidimensionais.

A comunicação é feita através de mensagens, utilizando padrões web – HTTP, SOAP, e XML – o que motiva desenvolvedores que pretendem acessar fontes de dados através da internet a adotarem tal padrão. XMLA pode ser encarada como a especificação para um conjunto de interfaces de mensagens XML que utilizam SOAP para definir iteração de acesso de dados entre uma aplicação cliente e um provedor analítico conectados através da internet. A linguagem de consulta utilizada é o MDX, já explicada anteriormente, que pode ser considerada a linguagem de consulta multidimensional mais utilizada atualmente.

Com relação à aceitação deste padrão, podemos ressaltar que o que diferencia o XMLA das tentativas anteriores de padronização é que o primeiro já

tem suporte de grandes empresas como Hyperion, Microsoft, SAP e SAS. A adoção do XMLA por um número cada vez maior de empresas traz ganhos inestimáveis tanto do ponto de vista técnico quanto do ponto de vista de negócio. Do ponto de vista técnico, ganha-se um padrão aberto de API de acesso aos principais servidores OLAP disponíveis no mercado ou mediante licenças públicas. Com isso, os demais provedores e consumidores passam a ser encorajados a adotarem o modelo. Do ponto de visto de negócio, clientes e fornecedores podem diminuir custos já que os servidores OLAP tornam-se cambiáveis mantendo uma interface de acesso única.

O padrão XMLA define dois métodos gerais de acesso:

Discover – método responsável por obter informações de metadados de um web service que provê acesso às funcionalidades e ao conteúdo de um servidor OLAP. Com o uso de listas de propriedades e uma interface genérica, permite que uma aplicação cliente acesse as informações descritivas de uma fonte de dados sem ter que reescrever nenhum código.

- **Execute** – Com base nas informações obtidas pelo método Discover, o cliente pode montar consultas ao conteúdo de cubos OLAP em uma fonte de dados específica. O comando execute permite a execução dessas consultas, retornando os resultados em XML, segundo o esquema definido pelo XMLA e dados descritivos do cubo e da visualização acessados.

Tipos de dados utilizados no XMLA:

- **Boolean**: utiliza padrão XML de tipos de dado booleanos.
- **Decimal**: utiliza padrão XML de tipos de dado decimais.

- **Integer**: utiliza padrão XML de tipos de dado inteiro.
- **EnumString**: utiliza padrão XML de tipos de dado String. Este tipo define um conjunto de constantes para um determinado numerador (enum). O valor específico para cada constante é especificado com a definição do numerador.
- **MDDataSet**: Daremos uma atenção importante para este tipo de dado devido `a sua importância para este trabalho. Trata-se de um formato usado para trazer dados multidimensionais. Representa dados OLAP em XML. Um MDDataSet possui três seções principais:
    1. **OLAPInfo**: Define a estrutura do resultado, listando os cubos, arestas e células;
    2. **Axes**: Contém os dados das arestas definidas na seção OLAPInfo;
    3. **CellData**: Contém os dados das células definidas na seção OLAPInfo;

A Figura 5 ilustra a estrutura de um documento XMLA enviado como resposta a uma consulta a um cubo de dados. Este documento é associado a uma definição de esquema e logo em seguida aparecem as seções OlapInfo, Axes e CellData, descritas em mais detalhes a seguir.

```
<ExecuteResponse xmlns="urn:schemas-microsoft-com:xml-analysis">
 <return>
  <root>
   <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!-- The XML schema definition of the result comes here -->
    <!--xsi:noNamespaceSchemaLocation="./exemplo.xsd"-->
    <include schemaLocation="./exemplo.xsd"/>
   </xsd:schema>
   <OlapInfo>
    <!--  Dimension information comes here  -->
   </OlapInfo>
   <Axes>
    <!--  Axis information for the MDDataSet result axes comes here  -->
   </Axes>
   <CellData>
    <!--  Cell data values and properties come here  -->
   </CellData>
  </root>
 </return>
</ExecuteResponse>
```

Figura 5. Estrutura de um documento XMLA

OLAPInfo

A seção OLAPInfo contém três elementos:

- **CubeInfo**: contém uma coleção de elementos descritivos do cubo, incluindo informações sobre as suas arestas e células em elementos aninhados do tipo AxesInfo e CellInfo.
- **AxesInfo**: Inclui as definições de arestas do cubo em elementos em um conjunto de elementos <AxisInfo> que têm como alias um ordinal, name="Axis0", por exemplo. As hierarquias são listadas com suas propriedades (<HierarchyInfo>). A Figura 6 exemplifica a estrutura de um elemento AxisInfo.

- **CellInfo**: O elemento CellInfo contém a definição das propriedades das células, dentro de um cubo específico, como exemplificado na Figura 7. Esta estrutura permite que as células também tenham propriedades adicionais. As propriedades listas neste exemplo são Value, FmtValue e a propriedade adaptada FormatString.

```
<OlapInfo>
  <AxesInfo>
    <AxisInfo name="Axis0">
      <HierarchyInfo name="Measures">
        <UName name="[Measures].[MEMBER_UNIQUE_NAME]"></UName>
        <Caption name="[Measures].[MEMBER_CAPTION]"></Caption>
        <LName name="[Measures].[LEVEL_UNIQUE_NAME]"></LName>
        <LNum name="[Measures].[LEVEL_NUMBER]"></LNum>
        <DisplayInfo name="[Measures].[DISPLAY_INFO]"></DisplayInfo>
      </HierarchyInfo>
    </AxisInfo>
      <AxisInfo name="Axis1">
        <HierarchyInfo name="Store">
          <UName name="[Store].[MEMBER_UNIQUE_NAME]"></UName>
          <Caption name="[Store].[MEMBER_CAPTION]"></Caption>
          <LName name="[Store].[LEVEL_UNIQUE_NAME]"></LName>
          <LNum name="[Store].[LEVEL_NUMBER]"></LNum>
          <DisplayInfo name="[Store].[DISPLAY_INFO]"></DisplayInfo>
          <Store_x0020_SQFT name="[Store].[Store Name].[Store SQFT]"></Store_x0020_SQFT>
        </HierarchyInfo>
        <HierarchyInfo name="Time">
          <UName name="[Time].[MEMBER_UNIQUE_NAME]"></UName>
          <Caption name="[Time].[MEMBER_CAPTION]"></Caption>
          <LName name="[Time].[LEVEL_UNIQUE_NAME]"></LName>
          <LNum name="[Time].[LEVEL_NUMBER]"></LNum>
          <DisplayInfo name="[Time].[DISPLAY_INFO]"></DisplayInfo>
        </HierarchyInfo>
      </AxisInfo>
```

Figura 6. Estrutura de um XMLA - AxesInfo

```
<OlapInfo>
...
        <CellInfo>
            <Value name="VALUE"></Value>
            <FmtValue name="FORMATTED_VALUE"></FmtValue>
          <FormatString name="FORMAT_STRING"></FormatString>
        </CellInfo>
```

Figura 7. Estrutura de um XMLA - CellInfo

Axes

A seção Axes possui um conjunto de itens denominados Axis que são listados na ordem em que aparecem no dataset, iniciando por zero. A propriedade AxisFormat determina como os elementos especificados pelos Axis serão apresentados. Os provedores de XMLA precisam suportar os seguintes valores para a propriedade AxisFormat:

- **ClusterFormat:** Neste formato as arestas são representadas como conjuntos.
- **TupleFormat:** As arestas são representadas como conjuntos de tuplas.
- **CustomFormat:** Permite que o provedor gere as arestas em qualquer combinação válida citada anteriormente.

CellData

A seção CellData contém os valores de cada célula. O atributo CellOrdinal indica a posição de cada célula e é numerado de 0 a n-1, para n células. Aqui são preenchidos os valores das propriedades especificadas na sessão CellInfo.

```
<CellData>
  <Cell CellOrdinal="0">
    <Value xsi:type="xsd:double">16890</Value>
    <FmtValue>16,890.00</FmtValue>
    <FormatString>Standard</FormatString>
  </Cell>
  <Cell CellOrdinal="1">
    <Value xsi:type="xsd:int">50</Value>
    <FmtValue>50</FmtValue>
    <FormatString>Standard</FormatString>
  </Cell>
  <Cell CellOrdinal="2">
    <Value xsi:type="xsd:double">36175.2</Value>
    <FmtValue>$36,175.20</FmtValue>
    <FormatString>Currency</FormatString>
  </Cell>
</CellData>
```

Figura 8. Estrutura de um XMLA - CellData

2.2 Sistemas de Informação Geográfica (SIG)

Sistemas de Informação Geográfica (SIG) são sistemas computacionais capazes de manipular dados espaciais referenciados à superfície da Terra. De maneira mais detalhada, SIG são sistemas capazes de integrar, armazenar, editar, analisar, compartilhar e exibir dados geográficos.

Apesar de seu aparecimento ter acontecido primeiro nos órgãos públicos e para gerenciamento de recursos naturais no início da década de 80, o contínuo barateamento de equipamentos e aumento da capacidade de processamento gráfico dos mesmos tem trazido os benefícios proporcionados pelos SIGs para o meio corporativo [Inmon, 1996]. Podem ser citados como alguns dos principais usos de SIGs:

- **Planejamento Estratégico** – a utilização de informações relativas à localização de residências em relação a pontos de venda de determinada empresa, por exemplo, pode influenciar na decisão de abertura de uma nova filial.
- **Projetos Ambientais** – a análise de quais áreas de um território estão sofrendo maior degradação florestal, por exemplo, pode ser facilitada em muito com a utilização de um SIG.
- **Indústrias em Rede** – setores de energia elétrica, telefonia, água, esgoto e gás, entre outros, formam uma gama específica, já que essas redes são naturalmente tratadas de forma geográfica.

2.2.1 Interconexão de SIGs

Uma das dificuldades enfrentadas por Sistemas de Informações Geográficas (SIG) é a troca de informações entre sistemas distintos [Frozza e Mello, 1996]. A complexidade e a riqueza dos modelos de dados geográficos faz com que diferentes SIGs empreguem diferentes formas para representar a realidade geográfica. A ampliação e a diversificação do uso de SIGs nas organizações fez surgir a necessidade de intercâmbio de informações georeferenciadas existentes em fontes autônomas e heterogêneas [Zhang *et al.*, 2004].

O intercâmbio de informações entre SIGs é particularmente importante devido aos custos de levantamento e tratamento de dados geográficos, economia

que pode ser obtida mediante a integração e reuso de dados heterogêneos e aos potenciais ganhos de análise de informações com cruzamentos de dados de diversas fontes.

Esta seção descreve os principais esforços voltados para a definição de padrões para interconexão de SIGs sendo propostos atualmente. O objetivo deste trabalho é usar construções desses formatos e protocolos como ponto de partida para o desenvolvimento formatos de dados integrados e APIs para a interoperabilidade entre sistemas OLAP e SIG.

OpenGIS®

OpenGIS é uma marca registrada do Open Geospatial Consortium, Inc (OGC), consórcio industrial internacional composto por 339 empresas, agências governamentais e universidades envolvidas com o desenvolvimento, utilização e interoperabilidade de SIG. O OGC possui como principal objetivo a padronização no desenvolvimento e utilização de aplicações geográficas nos mais variados meios, através de especificações de interfaces públicas [OpenGIS, 2007].

As especificações OpenGIS® – ou OpenGIS® Specifications, como são citadas internacionalmente – são documentos técnicos que descrevem interfaces ou padrões de representação de dados intercambiados entre SIGs. Essas especificações buscam, como objetivo final, a resolução de problemas relacionados à interoperabilidade de sistemas geográficos. Em um cenário ideal, dois SIGs desenvolvidos por empresas diferentes utilizando as especificações OpenGIS® podem ser capazes de trabalhar de maneira integrada como componentes, sem necessidade de esforço adicional [OpenGIS, 2007].

GML

Geography Markup Language (GML), ou linguagem de marcação geográfica em português, uma das principais especificações OpenGIS, é uma gramática XML escrita em XML Schema que visa a modelagem, transporte e armazenamento de informação geográfica [Kimball,1996]. Por basear-se em XML, a GML permite que informações geográficas sejam enviadas através da web utilizando protocolos como HTTP e formatos de mensagens padronizados, tais com envelopes SOAP.

Por ser uma especificação do OpenGIS®, a GML baseia-se nos conceitos da OpenGIS Abstract Specification. Isto consiste, basicamente, na utilização do conceito de feições (*features* em inglês) e sistema de referência baseado em coordenadas, entre outros objetos, para a descrição de dados geográficos [OpenGIS, 2007].

Uma *feature* pode ser entendida como a abstração de um fenômeno do mundo real, referenciando um ponto da Terra. Desta maneira, a representação do mundo real pode ser entendida como um conjunto de *features*. Uma *feature* é composta por uma ou mais propriedades, cada qual definida por uma tupla da forma **{nome, tipo valor}** [OpenGIS, 2007].

```
<gml:Point gml:id="p21" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
   <gml:pos dimension="2">45.67 88.56</gml:pos>
</gml:Point>
```

Figura 9: Exemplo de representação de um ponto em GML

Segundo a própria [OpenGIS, 2007], a especificação GML define a sintaxe de um XML Schema, com mecanismos e convenções para:

- Prover um framework aberto e independente de fornecedor para definição de esquemas e objetos geoespaciais;
- Possibilitar aplicações que suportem capacidades descritivas do framework GML;
- Suportar a descrição de esquemas de aplicação geoespacial para domínios específicos e comunidades de informação;
- Permitir a criação e a manutenção de aplicações que utilizem esquemas e conjuntos de dados geográficos conectados;
- Suportar o armazenamento e o transporte de esquemas de aplicação e conjuntos de dados geográficos;
- Aumentar a capacidade das organizações compartilharem esquemas de aplicações geográficas e a informação representada de acordo com esses esquemas.

Web Service Common (WSC)

A *OpenGIS® Web Service Common Implementation Specification (WSC)* é um dos mais recentes documentos do OGC. A especificação tem como principais objetivos formalizar muitos dos aspectos comuns entre as *OGC Web Service (OWS) Interface Implementation Specifications*, como mensagens de requisição e retorno, parâmetros e codificação das mesmas. Atualmente, fazem parte das

OWS Specifications a *Web Map Service (WMS)*, *Web Feature Service (WFS)* e *Web Coverage Service (WCS)*.

Web Map Service (WMS)

A *OpenGIS® Web Map Service Implementation Specification* (WMS) define uma interface de programa de aplicação (API) que, além de servir de base para outras especificações, busca definir padrões para a criação e a disponibilização de visualizações compostas pela sobreposição de mapas providos simultaneamente por diversas fontes remotas e heterogêneas.

Na prática, a WMS permite que um usuário combine diferentes mapas de diferentes servidores, distribuídos na Web, em uma única visualização, e obtenha informações sobre possíveis *features* de cada um, através de uma única API. Do ponto de vista de negócio, isso permite que empresas apostem na solução que julgarem mais apropriada, já que implementando os clientes e os servidores seguindo as especificações eles tornam-se totalmente acopláveis.

A WMS define três métodos em sua especificação, descritos a seguir.

- **GetMap** define como solicitar e retornar mapas, seja como uma imagem ou um conjunto de *features*.
- **GetFeatureInfo** define como pedir e prover informações sobre o conteúdo de determinado mapa como, por exemplo, o valor de uma *feature* em determinado local.
- **GetCapabilities** retorna a descrição de quais tipos de mapas um determinado servidor pode prover.

Web Map Context (WMC)

A *OpenGIS® Web Map Context Implementation Specification* (WMC) funciona como uma extensão da WMS, permitindo que a visualização conjunta de vários mapas, provenientes de servidores distintos distribuídos na Web, seja salva e possa ser carregada posteriormente. O "contexto", descrevendo a composição de níveis de informação de mapas de diferentes servidores, é salvo em um arquivo XML, que pode ser carregado posteriormente pelo usuário que criou a visualização ou mesmo por outros usuários.

A utilização da WMC pode ser útil em diversas situações, entre elas a reutilização de uma visualização de mapas comum a um determinado grupo de pessoas, a manutenção da visão criada por um usuário entre uma seção e outra, ou mesmo para catalogação e posterior pesquisa de visualizações interessantes.

Web Feature Service (WFS)

A *OpenGIS® Web Feature Service Implementation Specification* (WFS) permite a um cliente requisitar e atualizar dados geoespaciais de múltiplos servidores. Basicamente, a implementação da especificação permite a um usuário realizar a criação, remoção e atualização de dados geográficos, ativação de trancas (*locks*) e consulta em diferentes repositórios de dados geográficos utilizando GML via web.

Web Coverage Service (WCS)

A *OpenGIS® Web Coverage Service Implementation Specification* (WCS) pode ser vista como uma especificação semelhante tanto à WMS quanto à WFS.

Seu objetivo é prover informações de cobertura – informação geoespacial variável no tempo e no espaço – através da web.

Como a WMS e a WFS, a WCS também permite ao usuário solicitar determinadas porções de informação armazenadas pelo servidor, através de constantes espaciais e outros parâmetros. Ao contrário da WMS, porém, as informações retornadas por um *Web Coverage Service* não são estáticas, e possuem uma rica sintaxe e semântica que devem ser interpretadas no cliente para exibição.

A WCS define, como a WMS, três métodos em sua especificação, descritos a seguir.

- **GetCapabilities** define quais coberturas podem ser solicitadas pelo cliente.
- **DescribeCoverage** descreve em profundidade uma ou mais coberturas disponibilizadas por determinado servidor.
- **GetCoverage** retorna uma cobertura, ou seja, valores e propriedades que definem a cobertura de um conjunto de coordenadas geográficas.

A Figura 10 sintetiza as características das especificações descritas acima, de modo a facilitar a sintetizar o entendimento de seus papéis complementares.

Especificação	WMS	WFS	WCS
Objetivo	Combinar diferentes mapas em uma única visualização.	Requisitar e atualizar dados geoespaciais de múltiplos servidores	Prover informações de cobertura – informação geoespacial variável no tempo e no espaço
Métodos Implementados	GetMap, GetFeatureInfo e GetCapabilities	N/A	GetCapabilities, DescribeCoverage e GetCoverage
Vantagens	Soluções acopláveis e visualizações compostas.	Operações de DDL e DML em diferentes repositórios	Informações detalhadas de uma cobertura
Tipo de Retorno	Estático	Estático	Dinâmico. Rica sintaxe e semântica que devem ser interpretadas

Figura 10. Tabela comparativa das principais especifcações da OGC.

3. INTEGRAÇÃO OLAP X SIG

3.1. Barreiras na Integração

Há algum tempo a utilização ferramentas OLAP no ambiente de negócios é tida como consenso no que se refere a sistemas de suporte à decisão. Rapidamente, o diferencial competitivo provido por tal tecnologia fez com que passasse de novidade a *commoditie,* isto é, os componentes de uma arquitetura OLAP podem ser intercambiados graças à utilização de padrões de conexão como MDX e XMLA. A integração de ferramentas OLAP a sistemas de informações geográficas (SIG), entretanto, pode ser caracterizada como um tema recente e um desafio de pesquisa e desenvolvimento.

A utilização de OLAP em conjunto com técnicas de gerenciamento, intercâmbio e visualização de informações geográficas tem grande potencial para enriquecer as análises de informações georreferenciadas, por meio da apresentação das informações de tabelas dinâmicas ou gráficos também na forma de mapas. [Penna et al.] defende que a utilização de ferramentas OLAP integradas a SIG pode significar um salto na eficiência e eficácia da análise de dados multidimensionais. Segundo [Rao et al., 2003], 80% dos dados presentes em sistemas transacionais estão ligados, direta ou indiretamente, a informações geográficas. Os dados de servidores OLAP, porém, encontram-se armazenados de forma alfanumérica, enquanto os dados de SIGs muitas vezes estão dispersos em diversos servidores, que usualmente utilizam formatos proprietários, pois os padrões na área de GIS ainda não estão bem definidos e estabelecidos, o que

impõe uma barreira à implementação de sistemas integrando recursos de OLAP e GIS.

Um dos grandes desafios na implementação de tal integração reside no fato de tanto OLAP quanto SIG já serem tecnologias consolidadas. Por esse motivo, o número de modelos conceituais propostos na bibliografia é desproporcional se comparado as propostas lógicas de implementação. Enquanto modelos conceituais representam, por seus objetivos, uma estrutura mais flexível, a falta de modelos lógicos que suportem os mesmos torna sua utilização no curto prazo inviável.

Distribuidores de bancos de dados têm suprido a necessidade de utilização de SGBDs para fins geográficos através da concessão de extensões aos produtos já existentes. Porém a integração entre as duas tecnologias não pode ocorrer através da simples extensão dos sistemas OLAP atuais. A manipulação de dados espaciais por data warehouses convencionais, segundo o autor, exige a revisão dos conceitos do próprio OLAP, já que influencia conceitos formais, lógicos e físicos na manipulação dos dados.

Considerando que a atual arquitetura OLAP retorna os dados de negócio através do formato XMLA, tem-se uma lacuna para o preenchimento de dados geográficos. Apesar de ainda não consolidada, a pesquisa para a integração de OLAP e GIS está fortemente direcionada à modificação do modelo dimensional convencional, a fim de capacitá-lo para a manipulação de dados geográficos – conceito referenciado como SOLAP (*Spatial OLAP*). Esta proposta busca agregar aos dados de negócio informações geográficas através do próprio data warehouse.

3.2. Trabalhos Relacionados

A utilização de ferramentas OLAP integradas a sistemas de informações geográficas (SIG) pode ser considerada um tema recente. Existem vários trabalhos relacionados a data warehouses espaciais, como [Penna et al.], que procura dar bastante ênfase aos conceitos e características desta integração. O trabalho descreve os benefícios da integração OLAP X SIG para as organizações e discute as principais questões relativas a data warehouses espaciais. Propõe um ciclo de projeto análogo ao proposto por Kimball [Kimball, 1996], visando sustentar metodologicamente o desenvolvimento e a implementação de um DW espacial. Este ciclo tem suas etapas muito semelhantes às do processo de data warehousing convencional. [Penna et al] se preocupa principalmente com os conceitos e a metodologia para implantação de um DW espacial, sem se aprofundar nos detalhes de modelagem e implementação. A questão de como estas duas tecnologias (OLAP e SIG) poderiam trocar informações também fica em segundo plano.

[Ferreira, 2002] propõe um framework conceitual para suportar a integração entre GIS e OLAP. Como se trata de um modelo conceitual, considera-se que seja independente de qualquer tecnologia ou ferramenta para sua implementação. Esse framework é baseado em uma arquitetura de três camadas, onde a camada de dados é responsável por prover os dados para análise. Ela é composta pelos dados pré-existentes nas aplicações GIS e nas aplicações OLAP. A segunda camada é a camada de mediação (*middleware*), que tem como função fazer a mediação entre as aplicações e a camada de dados. Finalmente, na camada de

aplicação, encontramos as aplicações responsáveis pelas funcionalidades de visualização, similares às existentes em ferramentas GIS e OLAP individualmente.

A segunda camada proposta por [Ferreira, 2002], mediação, pode ser considerada a mais importante, no que se refere à relação com o nosso trabalho. A camada de mediação proposta tem duas funções: (I) armazenar a definição e a correlação entre as diferentes aplicações que compõem a camada de dados e serviços de acesso aos mesmos e (II) realizar o processamento de consultas submetidas através da camada de aplicação.

A cada consulta feita na interface da camada de aplicação, no componente OLAP ou SIG, é disparado um evento que verifica se existe uma "correspondência de ação" a ser feita no outro componente da aplicação. Essa checagem é feita pela camada de mediação, através do acesso à base de integração, procurando na tabela "Elo" as estruturas OLAP e SIG que estão associadas. Dependendo da estrutura 'destino' associada, uma ação específica deve ser tomada.

A diferença marcante entre esta proposta e a que propomos está no fato que focamos em tecnologias abertas e de ampla utilização no mercado, buscando assim, suportar a integração de quaisquer ferramentas, ou seja, procurou-se desenvolver uma arquitetura genérica para integração de ferramentas SIG e OLAP. Nossa base de metadados, ao contrário da citada anteriormente, se baseia em XML a fim de facilitar a troca de dados entre as ferramentas.

Outro trabalho importante que segue a mesma linha e serve como referência para outros é o projeto GOLAPA (*Geographic On-Line Analytical Processing Architecture*) [Silva et al, 2006]. O trabalho apresenta uma arquitetura proposta pelo grupo de pesquisa do Centro de Informática da Universidade

Federal de Pernambuco (UFPE) e tem como objetivo a integração de ferramentas SIG e OLAP utilizando tecnologias abertas. A arquitetura GOLAPA é formada por cinco camadas, sendo cada camada formada por vários componentes. Essa arquitetura, diferente de outras propostas, contempla as fases de extração, transformação e carga de dados em um DW geográfico. Além disso, o formato utilizado para solicitar a geração de um mapa é realizado através de um novo formato, definido pelo projeto, o GMLA (*GML for Analysis*). Nesta abordagem, dados geográficos e analíticos são condensados em um único formato, provendo todas informações necessárias ao componente responsável pela geração do mapa.

Apesar de o GOLAPA ser uma proposta muito semelhante à nossa, a utilização deste novo formato é a grande diferença entre os dois trabalhos. O GMLA foi criado para tentar suprir a necessidade de transportar dados heterogêneos em um mesmo padrão. Procuramos utilizar o XMLA e MDX puros por razão de compatibilidade com outros sistemas desenvolvidos utilizando padrões OpenGIS.

Como ressaltado anteriormente, a definição e/ou utilização de um formato que consiga mesclar informações de negócio com informações geográficas, apesar de oferecer um ganho conceitual inquestionável, tem que transpor a barreira da compatibilidade com a estrutura atual de um OLAP. Grandes customizações nos servidores de interface são necessárias, o que inviabiliza a entrega de resultados práticos aplicáveis num curto prazo – motivo pelo qual se optou pela utilização do XMLA associado à base de metadados.

4. A ARQUITETURA PROPOSTA PARA INTEGRAÇÃO OLAP X SIG

A arquitetura proposta por este trabalho busca viabilizar a visualização de dados multidimensionais em conjunto a dados geográficos de maneira transparente ao usuário. Para isso, é necessário ter-se em mente a abstração da complexidade naturalmente imposta pelo ambiente, já que tratam-se de duas fontes de dados distintas, tanto no que diz respeito ao conteúdo quanto ao tipo de dados que armazenam.

A arquitetura sugerida pode ser entendida como uma extensão de uma arquitetura OLAP típica, acrescentando componentes para armazenamento, manipulação e apresentação de dados geográficos. Além disso, é introduzido um componente responsável para integração entre as tecnologias OLAP e SIG, peça chave na proposta da arquitetura e foco de desenvolvimento deste trabalho.

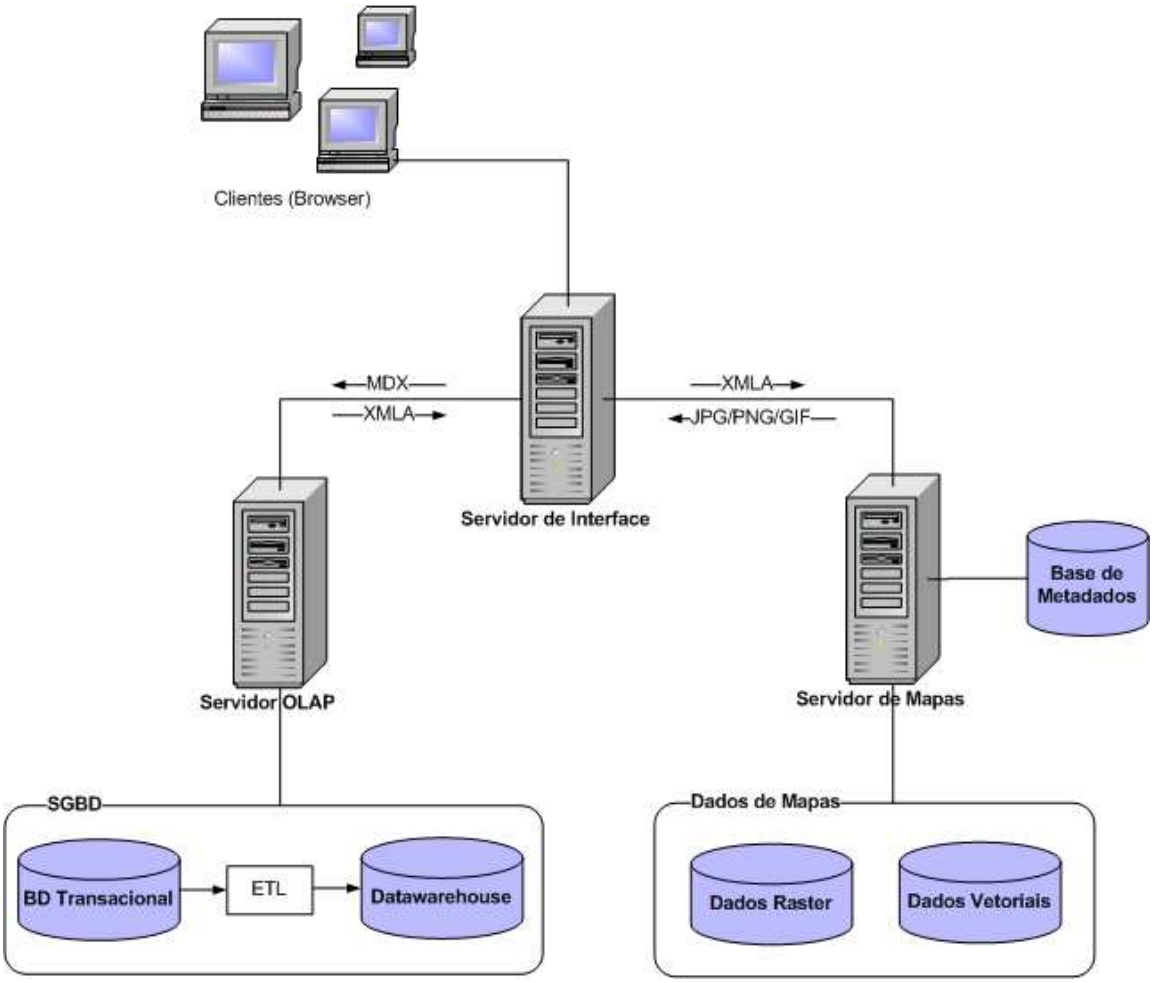

Figura 11. Arquitetura Proposta para Integração OLAP x SIG

Na Figura 11, a arquitetura OLAP típica pode ser identificada nos três componentes alinhados à esquerda: SGBD (incluindo o banco de dados transacional e o data warehouse obtido do primeiro mediante um processo de ETC - Extração, Transformação e Carga de dados), Servidor OLAP e Servidor de Interface OLAP sobre a Web. Nestes componentes podemos identificar claramente as camadas de dados (inferior), processamento (intermediária) e interface (superior), respectivamente. A comunicação entre esses três componentes é feita utilizando os mesmos protocolos que numa arquitetura OLAP

típica, já descritos na Seção 2 deste trabalho, isto é, MDX para especificação de consultas OLAP e XMLA para retornar as descrições e visões correspondentes dos cubos de dados.

Na parte inferior da Figura 11, um servidor SIG representa o componente responsável pelo armazenamento e manipulação dos dados geográficos. O papel do servidor SIG na arquitetura é fundamental, já que todas as características puramente geográficas são de sua responsabilidade. Fazendo um paralelo entre as atribuições do servidor SIG e a arquitetura OLAP, é possível afirmar que ele executa papel de análogo ao do servidor OLAP, isto é, na camada intermediária da arquitetura, porém da maneira necessária ao tratamento de dados geográficos.

Completando a arquitetura, o Servidor de Mapas centraliza o processo de integração entre as tecnologias OLAP e SIG. Ele é o único componente da arquitetura completamente desenvolvido no âmbito deste trabalho, desde sua especificação até a sua implementação. O Servidor de Mapas é o componente central da arquitetura, sendo o mediador (*middleware* centralizador) da comunicação entre a aplicação de interface e o servidor SIG. A implementação do *middleware* de integração OLAP X GIS envolve também algumas manutenções e extensões nos demais módulos da arquitetura, para possibilitar o intercâmbio dos dados através das APIs dos servidores de mapas.

A comunicação entre o Servidor de Interface e o Servidor de Mapas é feita através do encaminhamento, pelo Servidor de Interface, do XMLA recebido do Servidor OLAP para o Servidor de Mapas. O XMLA, por ser um formato definido para intercâmbio de dados multidimensionais, e não geográficos, possui em sua estrutura informações relacionadas puramente à análise numérica: valores e

descrições, com seus respectivos níveis hierárquicos analisados (se for o caso). O georeferenciamento dos dados OLAP no XMLA é implícito, através da referência a entidades geográficas (tais como, estados, municípios e distritos) pelos seus nomes na descrição de algumas das dimensões do modelo dimensional (por exemplo, o candidato reside em um local definido pelos nomes de uma seqüência de entidades geográficas na hierarquia). Para viabilizar a geração do mapa, então, é necessário que as informações relacionadas aos dados geográficos (cores, legendas e referências aos próprios mapas – como os polígonos representando as entidades geográficas) sejam agregadas na visualização das informações de um XMLA.

A Base de Metadados mantém o mapeamento das informações do DW com as do SIG. Ela contém um as especificações de mapeamento de entidades geográficas referenciadas no DW e retornadas pelo servidor OLAP na forma alfanumérica em documentos XMLA (e.g., nomes de estados e municípios), para suas respectivas extensões geográficas, necessárias para a exibição da informação em mapas. Os metadados de mapeamento são armazenados em um arquivo XML, o que garante a flexibilidade necessária à expansão da arquitetura proposta. A Figura 12 ilustra através de exemplos a estrutura das entradas da Base de Mapeamento.

Para localizar a extensão geográfica de determinado item a ser plotado, é realizada uma consulta sobre a Base de Metadados. Tal consulta utiliza como chave de busca o identificador alfanumérico da entidade geográfica referenciado no arquivo XMLA. Como este identificador faz parte da estrutura obrigatória do XMLA, é possível identificar os membros das dimensões com componentes

geográficos referenciados no documento XMLA e determinar qual componente de mapa deve ser utilizado para plotar determinado membro de uma dimensão. A cor de preenchimento do componente no mapa vai depender da medida associada ao mesmo no documento XMLA retornado pelo servidor OLAP para responder uma consulta postada pelo usuário.

```
− <GeoStruct>
    <UniqueName>[UF].[RS]</UniqueName>
    <ClassExpression>RS</ClassExpression>
    <FileName>uf</FileName>
    <ClassItem>uf</ClassItem>
    <Color>255;184;149</Color>
    <OutlineColor>0;0;0</OutlineColor>
  </GeoStruct>
− <GeoStruct>
    <UniqueName>[UF].[SC]</UniqueName>
    <ClassExpression>SC</ClassExpression>
    <FileName>uf</FileName>
    <ClassItem>uf</ClassItem>
    <Color>255;184;149</Color>
    <OutlineColor>0;0;0</OutlineColor>
  </GeoStruct>
```

Figura 12. Exemplo de estrutura da Base de Metadados

A decisão pela utilização da Base de Metadados pode ser bastante questionável do ponto de vista conceitual. Ao propor uma base auxiliar propõe-se, obrigatoriamente, que os dados desta base sejam sempre mapeados pelo Servidor de Mapas a suas respectivas dimensões. Isto pode ser um empecilho, por exemplo, ao considerarmos a utilização de um serviço web genérico para geração de mapas. Mais fácil seria a utilização de um formato que contivesse, ao mesmo tempo, informações de negócio (numéricas) e geográficas.

A definição e/ou utilização de um formato que contemple tanto informações de negócio quanto informações geográficas, por outro lado, apesar de oferecer um ganho conceitual inquestionável, esbarra na questão de compatibilidade ao analisarmos sua implantação no atual cenário OLAP. Sua implementação exigiria grandes customizações nos Servidores de Interface hoje disponíveis no mercado, o que inviabilizaria a entrega de resultados práticos aplicáveis num curto prazo – motivo pelo qual se optou pela utilização do XMLA associado à Base de Metadados.

O objetivo deste trabalho é que, associadas às informações geográficas aos dados extraídos do XMLA, o Servidor de Mapas permita o retorno do mapa nos diversos formatos de imagem utilizados na Web (png, gif, jpeg) e também representações dos mapas em linguagens de marcação e outras representações compactas e versáteis, que possam ser interpretadas por navegadores Web e plug-ins, tais como *Scalable Vector Graphics* (SVG). O objetivo dessas representações, que poderão ser implementadas futuramente, é possibilitar certa dinâmica na manipulação dos mapas (e.g., *zoom in*, *zoom out*, adição e remoção de níveis de informação), eficientemente, nos clientes da aplicação.

Fluxo de Processamento

Descritos os componentes, podemos apresentar o fluxo básico de processamento dos dados na arquitetura proposta, o qual consiste dos seguintes passos:

1. O usuário interage com a aplicação de interface, selecionando o agrupamento de determinadas medidas disponíveis no cubo OLAP,

possivelmente referenciando hierarquias de dados geográficos em alguma dimensão do cubo (e.g., número de aprovados dentro de um estado, para cada município daquele estado);

2. A aplicação de interface solicita ao servidor OLAP o agrupamento solicitado, através de um comando MDX. O servidor OLAP, por sua vez, realiza as consultas necessárias, possivelmente utilizando o SGBD, para obter os agrupamentos e aplicar os filtros solicitados, retornando os uma visão do cubo em XMLA;
3. Ao receber o documento XMLA resultante da consulta, a aplicação de interface solicita ao Servidor de Mapas a geração dos mapas, enviando ao mesmo o arquivo XMLA recebido do servidor OLAP;
4. Recebendo o arquivo XMLA, o Servidor de Mapas realiza a iteração nodo a nodo na estrutura prevista, extraindo informações como a aresta do cubo em que encontram-se as informações geográficas, os valores consultas e, o mais importante, a estrutura hierárquica em que as dimensões geográficas estão dispostas;
5. Com os dados extraídos do XMLA na forma de uma classe, o Servidor de Mapas é capaz de identificar quais dimensões geográficas estão sendo consultadas e qual sua granularidade. Com base nestas informações, é realizada uma nova iteração, desta vez pelos objetos criados a partir do XMLA, para associação das informações geográficas. Nesta iteração, o Servidor de Mapas realiza diversas consultas à base de metadados, agregando às

estruturas anteriormente criadas seus atributos geográficos (cor padrão, arquivo de mapas, identificador geográfico, etc);

6. Com as informações retornadas pela base de metadados, o Servidor de Mapas gera, com base nos arquivos de mapas disponibilizados, as diversas camadas necessárias para a correta interpretação visual – através de cores e legendas – sempre com base nos parâmetros configurados pelo usuário;
7. Gerado o mapa em memória, o Servidor de Mapas realiza a conversão do mesmo para um formato de imagem, realizando o retorno desta à aplicação de interface;
8. A aplicação de interface, de posse do arquivo XMLA e do mapa em formato de imagem, é capaz de exibir ambos, agregando valor à análise dos dados apresentados.

A arquitetura e o fluxo aqui sugeridos dão ênfase à utilização de ferramentas e padrões abertos. Espera-se, com isso, definir não só uma proposta extensível e maleável para utilização em uma gama diversificada de projetos, mas também aproveitar a evolução de cada um dos componentes para viabilizar, com o tempo, a maturidade da arquitetura e de suas implementações.

5. ESTUDO DE CASO

5.1 A Aplicação

O estudo de caso deste trabalho busca agregar à análise dos dados retornados por uma consulta OLAP a visualização dos dados geográficos em forma de mapa. Assim, o objetivo principal resume-se à geração de um mapa baseada nas informações retornadas pela consulta OLAP.

Dentro do fluxo proposto para a realização da integração entre OLAP e SIG descrito anteriormente, o papel da aplicação desenvolvida encaixa-se na intermediação entre o Servidor de Interface, Base de Metadados e os componentes dos mapas. Diferencia-se de um Servidor de Mapas convencional por desempenhar não somente a função de gerar determinado mapa, mas também customizá-lo dinamicamente de acordo com as informações providas pelo OLAP.

Levando em consideração o ponto de vista do usuário, a aplicação desenvolvida é capaz de gerar mapas baseados nas informações retornadas pelo servidor OLAP (formato XMLA), agregando uma série de funcionalidades dinamicamente, entre elas:

- Geração de Contornos Contextualizados: durante o processamento dos dados provindos da arquitetura OLAP, a aplicação realiza o mapeamento de cada nível de análise, desenhando somente os contornos necessários a cada uma delas. Assim, por exemplo, se um determinado estado estiver sendo analisado no nível municipal e outro somente no nível

estadual, os contornos municipais serão gerados somente para um dos estados, gerando um mapa mais limpo e direcionado à análise;

- Configuração de Expressões de Legendas: considerando que em algumas análises a visualização das legendas de cada unidade pode ser fundamental e que, em outras, tal visualização pode resultar num mapa pouco compreensível, o *middleware* foi projetado para que as expressões apresentadas nas legendas sejam configuráveis. Desta maneira, o mapa pode ser configurado para rotular cada unidade com seu texto e valor associados, somente um desses atributos ou sem nenhum deles;
- Coloração Gradual das Unidades de Análise: como funcionalidade principal na agregação de valor à análise, a coloração gradual das unidades de análise baseia-se em valores iniciais e finais de coloração, criando uma escala de cores para representação proporcional dos valores das medidas dentro da faixa resultante de uma consulta OLAP. Esta funcionalidade é importante na análise de consultas com muitas unidades de análise geográficas, permitindo identificar áreas valores baixos, elevados ou semelhantes da medida com facilidade (e.g., número de candidatos por região, representado por cores entre branco (nenhum candidato) e preto (número máximo de candidatos entre as regiões)).

Por tratar-se da primeira implementação do *middleware*, a versão apresentada neste trabalho possui algumas limitações de utilização, que poderão ser tratadas mais detalhadamente em trabalhos futuros. Neste contexto, vale

ressaltar que a aplicação desenvolvida visa o retorno de mapas em um único formato de imagem. Além disso, em casos de análises que cruzem mais de uma medida com as dimensões geográficas (e.g., número de aprovados e reprovados agrupados por UF e município), somente será gerado o mapa da primeira medida retornada. Por questões lógicas, esta situação exigiria mais de um mapa ou alguma lógica adicional de interpolação de imagens para viabilizar a coloração das regiões de acordo com os valores das duas medidas.

Para melhor entendimento da aplicação, serão apresentados exemplos práticos que explicitam as funcionalidades implementadas, relacionando consultas OLAP aos mapas gerados pela aplicação.

A Figura 13 apresenta uma consulta à DW de concursos vestibulares da Universidade Federal de Santa Catarina pelo número de aprovados e desistentes, dos estados da região Sul do Brasil, detalhando esses dados para algumas macro-regiões do estado de Santa Catarina.

		APROVADOS	DESISTENTES
SC	SERRANA	99	15
	SUL CATARINENSE	0	20
PR	-	66	35
RS	-	33	70

Figura 13. Número de aprovados e desistentes da região Sul do Brasil com detalhes para Santa Catarina

Ao enviar a requisição destes dados ao servidor OLAP, será gerado um XMLA de retorno com valores do tipo MDDataSet, utilizado para expressar dados multidimensionais.

```
- <Axis name="Axis0">
  - <Tuples>
    - <Tuple>
      - <Member Hierarchy="Aprovados">
          <UName>[Measures].[Aprovados]</UName>
          <Caption>Aprovados</Caption>
          <LName>Aprovado</LName>
          <LNum>0</LNum>
        </Member>
      </Tuple>
    - <Tuple>
      - <Member Hierarchy="Measures">
          <UName>[Measures].[Desistentes]</UName>
          <Caption>Desistentes</Caption>
          <LName>Desistentes</LName>
          <LNum>0</LNum>
        </Member>
      </Tuple>
    </Tuples>
  </Axis>
```

Figura 14. Definição das colunas retornadas no XMLA

```
- <Axis name="Axis1">
  - <Tuples>
    - <Tuple>
      - <Member Hierarchy="UF">
          <UName>[UF].[PR]</UName>
          <Caption>PR</Caption>
          <LName>[UF]</LName>
          <LNum>2</LNum>
          <DisplayInfo>131077</DisplayInfo>
        </Member>
      </Tuple>
    - <Tuple>
      - <Member Hierarchy="UF">
          <UName>[UF].[SC]</UName>
          <Caption>SC</Caption>
          <LName>[SC]</LName>
          <LNum>2</LNum>
          <DisplayInfo>131077</DisplayInfo>
        </Member>
      - <Member Hierarchy="Regiao">
          <UName>[Regiao].[Serrana]</UName>
          <Caption>Serrana</Caption>
          <LName>[Regiao]</LName>
          <LNum>1</LNum>
          <DisplayInfo>131075</DisplayInfo>
        </Member>
      </Tuple>
    - <Tuple>
      - <Member Hierarchy="UF">
          <UName>[UF].[SC]</UName>
          <Caption>SC</Caption>
          <LName>[SC]</LName>
          <LNum>2</LNum>
          <DisplayInfo>131077</DisplayInfo>
        </Member>
      - <Member Hierarchy="Municipio">
          <UName>[Regiao].[Sul Catarinense]</UName>
          <Caption>Sul Catarinense</Caption>
          <LName>[Regiao]</LName>
          <LNum>1</LNum>
          <DisplayInfo>131075</DisplayInfo>
        </Member>
      </Tuple>
    - <Tuple>
      - <Member Hierarchy="UF">
          <UName>[UF].[RS]</UName>
          <Caption>RS</Caption>
          <LName>[UF]</LName>
          <LNum>2</LNum>
          <DisplayInfo>131077</DisplayInfo>
        </Member>
      </Tuple>
    </Tuples>
  </Axis>
```

Figura 15. Definição das linhas retornadas no XMLA

As figuras 14 (colunas) e 15 (linhas) ilustram como estaria definida a seção Axes, explicada anteriormente, do XMLA retornado. Fica claro neste exemplo que apenas o estado de Santa Catarina será dividido em regiões. Os valores de cada elemento estarão dispostos na seção CellData como veremos na figura a seguir.

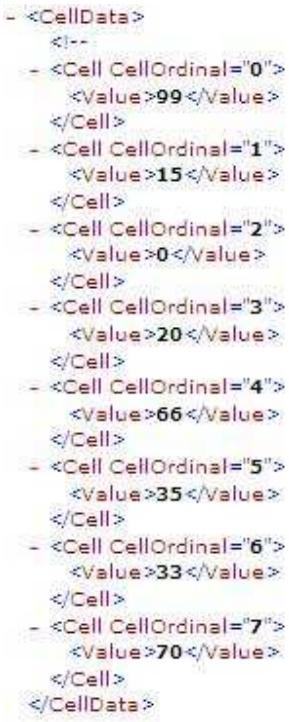

Figura 16. Definição dos valores retornados no XMLA - CellData

Em posse deste arquivo com dados devidamente preenchidos, o Servidor de Mapas realiza a iteração em cada nodo, extraindo informações das arestas, seus valores consultas e a estrutura hierárquica em que as dimensões geográficas se encontram.

Estas informações são então armazenadas em uma estrutura composta por objetos de uma classe denominada *GeoStruct* que será melhor explicada

posteriormente. Nesta fase, somente os atributos relacionados aos dados de negócio destes objetos serão preenchidos.

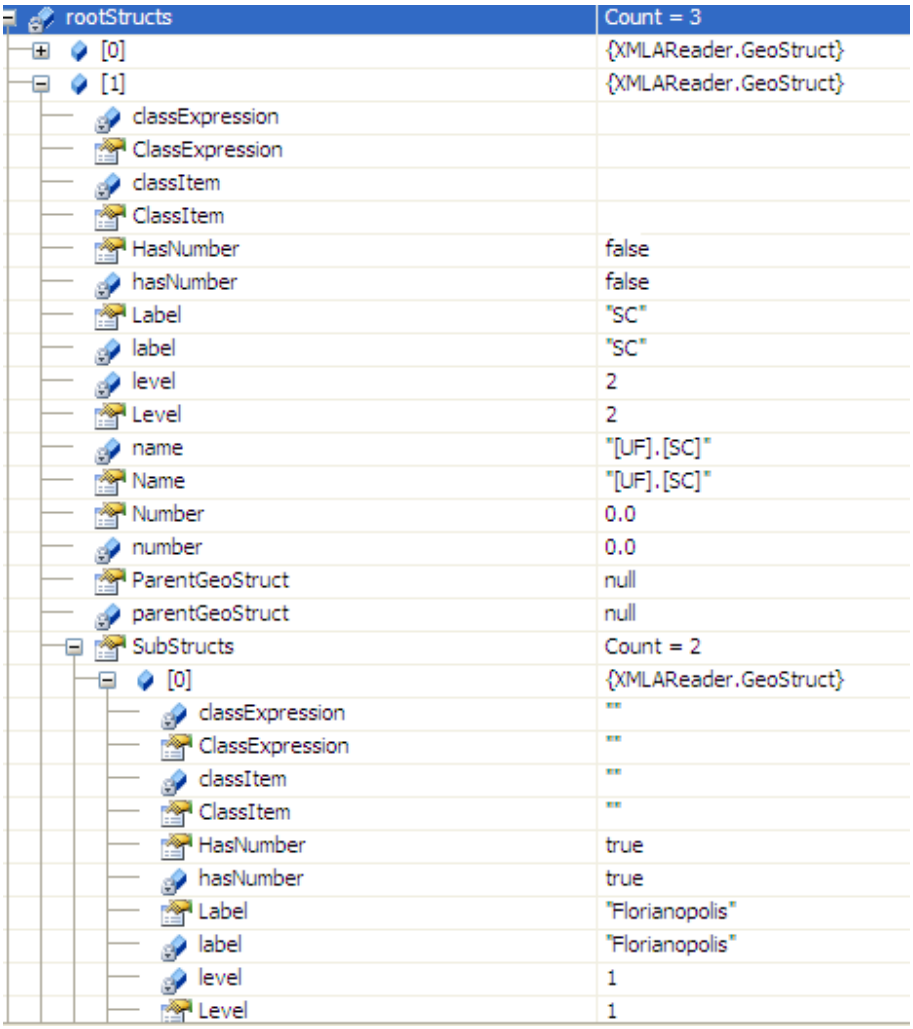

Figura 16. Objeto *GeoStruct* com seus atributos preenchidos.

A figura 16 mostra uma lista com três objetos *GeoStructs* com seus atributos devidamente preenchidos. Podemos observar a estrutura hierárquica no objeto de índice zero ([UF].[SC]), seu atributo *SubStructs* possui 2 objetos filhos (municípios) ilustrando a recursividade do modelo. Um nodo pode ter n nodos do mesmo tipo.

O próximo passo é agregar os valores geográficos aos atributos restantes. Nesse ponto é realizada uma nova iteração envolvendo a base de metadados para associação das informações geográficas e atualização dos objetos já criados. (Exemplo de uma base de metadados anteriormente ilustrada na figura 12).

Com base nos arquivos de mapas disponibilizados e em posse desta lista de objetos com dados de negócio e dados geográficos retornados pela base de metadados, o Servidor de Mapas gera as camadas necessárias para gerar o mapa. Gerado o mapa em memória, o Servidor de Mapas realiza a conversão do mesmo para um formato de imagem, trazendo-o para a interface.

E por fim, é gerado um mapa como ilustrado na figura 17.

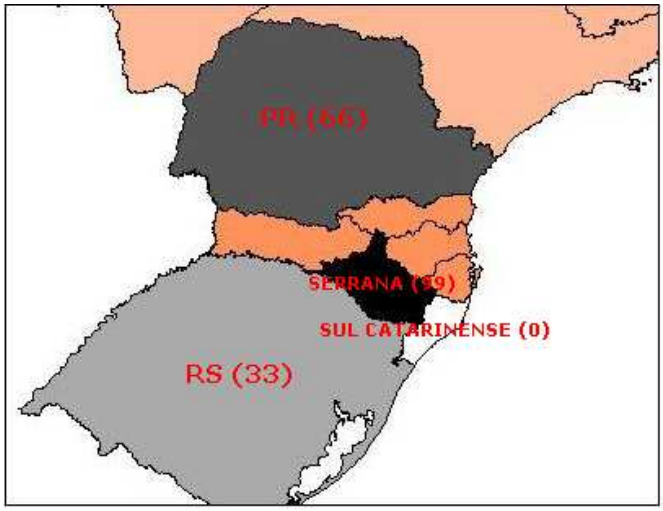

Figura 17. Mapa baseado no número de aprovados da região Sul

É possível, com este exemplo, visualizar que os contornos do nível de macro-região foram traçados somente para o estado de Santa Catarina. Para os demais estados os dados não estavam detalhados neste nível. Além disso, foram configuradas legendas tanto para a descrição dimensional quanto para o valor de cada item de análise. Por último, percebe-se a coloração diferenciada de cada

região presente na consulta OLAP, configuradas numa escala de branco para preto, o que permite a diferenciação imediata das unidades com maior valor relativo.

5.2 As tecnologias utilizadas na implementação

Para implementação da aplicação de integração descrita anteriormente, foi utilizado um conjunto de tecnologias, mesclando ferramentas abertas, de mercado e componentes desenvolvidos especificamente para este trabalho. Esta seção visa detalhar tecnicamente cada um dos componentes utilizados para o desenvolvimento da aplicação implementada. Neste trabalho, não são detalhados produtos relacionados à arquitetura OLAP, já que os mesmos podem ser alterados de maneira transparente ao restante da arquitetura.

A Figura 18 explicita, nos mesmos moldes de apresentação da arquitetura proposta, os componentes utilizados e desenvolvidos para implementação do *middleware* de integração OLAP x SIG. Cada um dos itens apresentados na Figura 18 é descrito na seqüência.

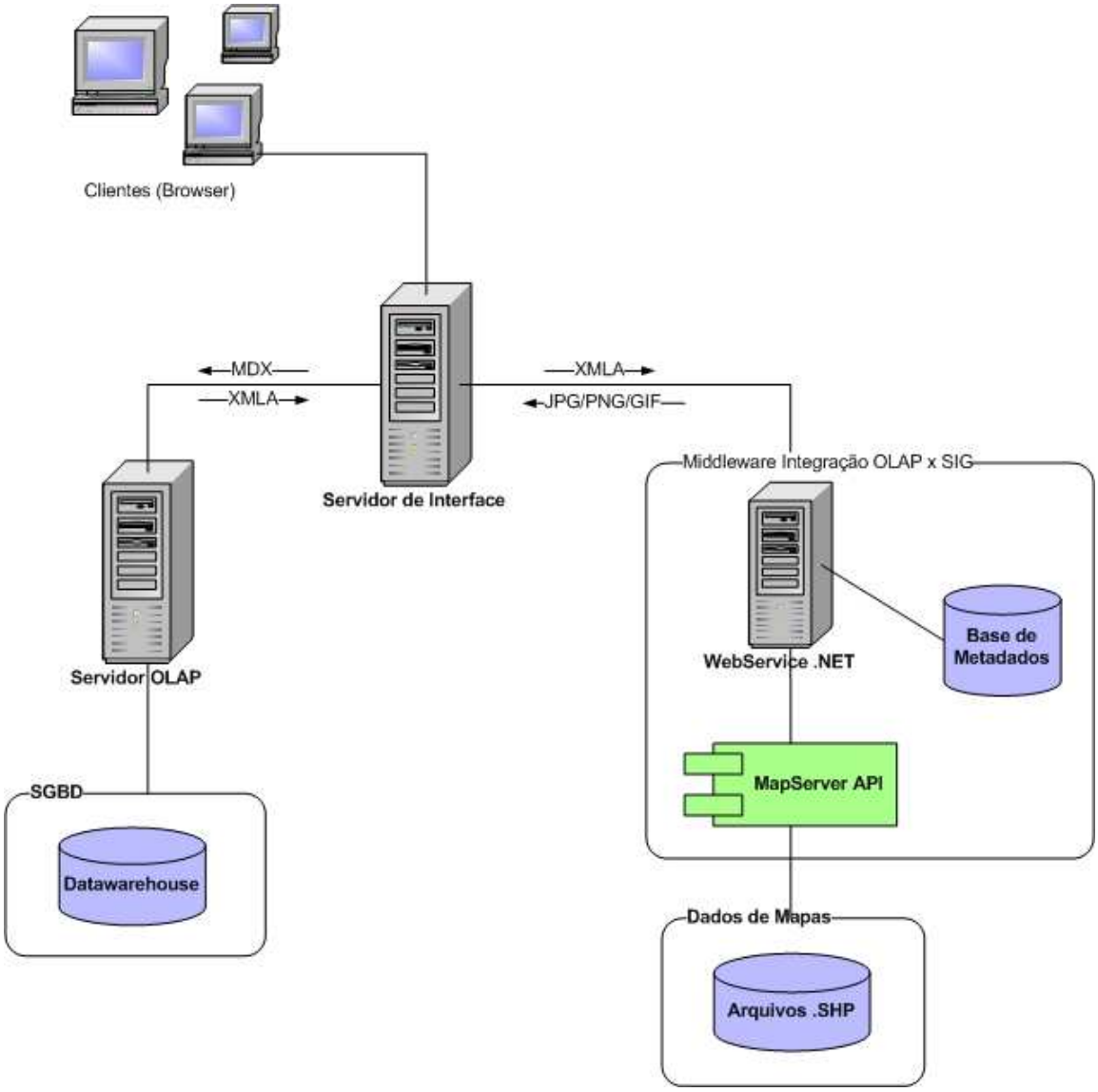

Figura 18. Arquitetura Implementada para Integração OLAP x SIG

Modelo Multidimensional

Apesar de não ser função deste trabalho descrever o SGBD utilizado – por tratar-se de um componente cambiável, como dito anteriormente – o modelo multidimensional contido no mesmo é de fundamental importância na aplicação proposta, haja vista a necessidade de entendimento do modelo para compreensão

das análises possíveis e, por conseqüência, dos requisitos necessários à geração dinâmica de mapas relacionados às mesmas.

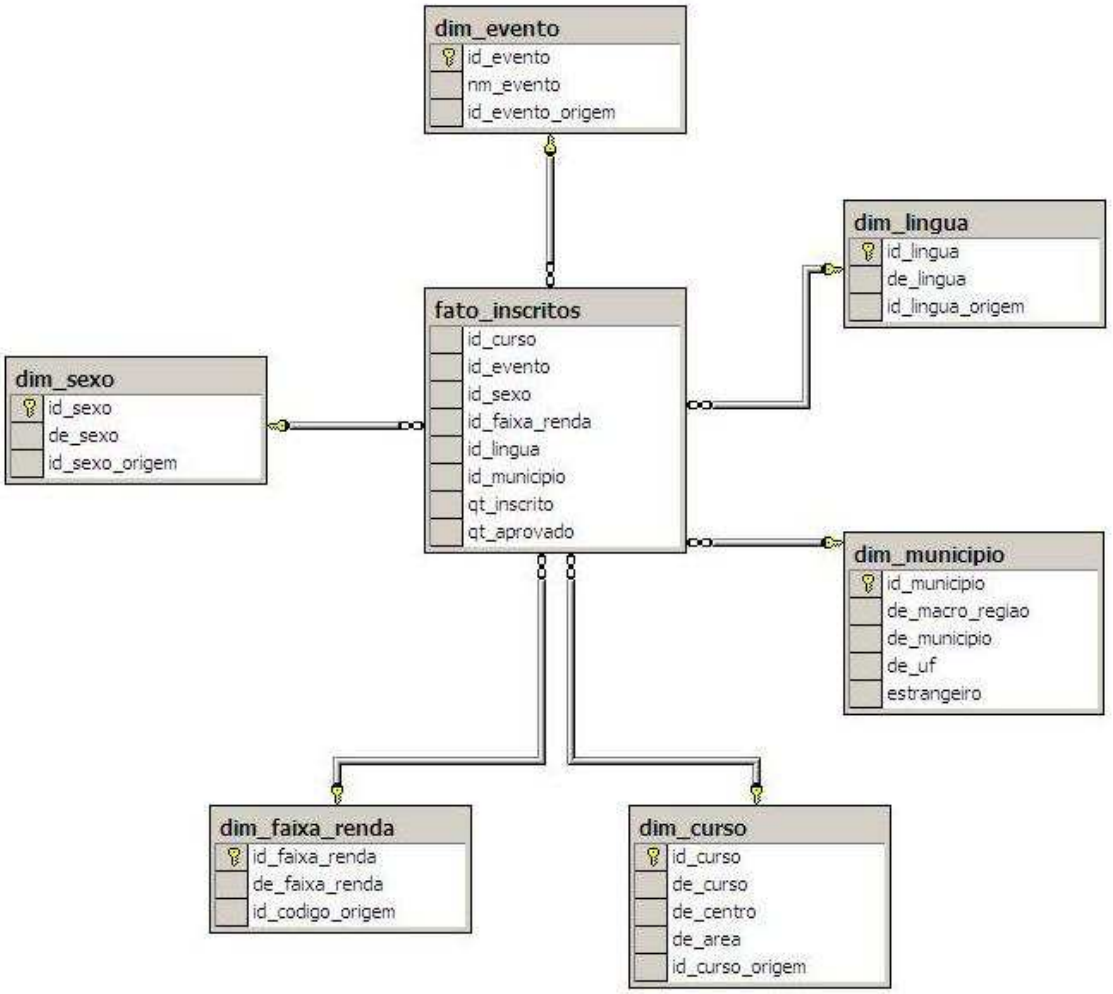

Figura 19. Modelo Multidimensional Utilizado no Desenvolvimento

A Figura 19 apresenta o modelo multidimensional utilizado na elaboração das consultas OLAP e geração de mapas. As dimensões sexo, uf e faixa de renda referem-se a características dos candidatos. Vale ressaltar que a dimensão uf será a dimensão que permitirá a geração de análises com dados geográficos (estado, macro-região e município). A dimensão curso indica a área, o centro e o curso

para o qual o candidato está prestando vestibular. A dimensão língua refere-se à escolha do candidato para a prova de idioma estrangeiro. A dimensão evento descreve os diferentes concursos vestibulares existentes no DW. A tabela de fato possui duas medidas: quantidade de inscritos e quantidade de aprovados.

Middleware de Integração OLAP x SIG

Como componente central da arquitetura proposta, por ser o responsável pela conciliação de dados de negócios e dados geográficos, o *middleware* de integração OLAP x SIG – aplicação criada durante o desenvolvimento deste trabalho – acumula uma série de funções, dentre elas:

- XMLA Parser: dentro do fluxo proposto, a decomposição do arquivo XMLA recebido do Servidor de Interface para uma estrutura própria (que permita a agregação de dados geográficos) é item essencial ao processamento interno;
- Busca de Dados Geográficos: após decomposição do XMLA, cabe ao middleware a função de, com os dados OLAP em mãos, consultar a Base de Metadados e selecionar, a cada item de análise, os componentes de representação geográfica (e.g., polígonos relativos a partições territoriais) necessários para posterior geração do mapa;
- Servidor de Mapas: apesar de não ser o responsável direto pela leitura dos componentes de mapa, o *middleware* utiliza uma biblioteca para acesso à funções do MapServer (ver seção específica) para desenho dos mapas.

O *middleware* realiza a iteração dentro dos nodos do XMLA, identificando a estrutura hierárquica da consulta OLAP realizada e criando uma estrutura própria de dados – a classe *GeoStruct*. Esta classe desempenha papel fundamental no processamento da aplicação, tanto na decomposição do XMLA, como na agregação de dados geográficos e desenho final do mapa com suas variações.

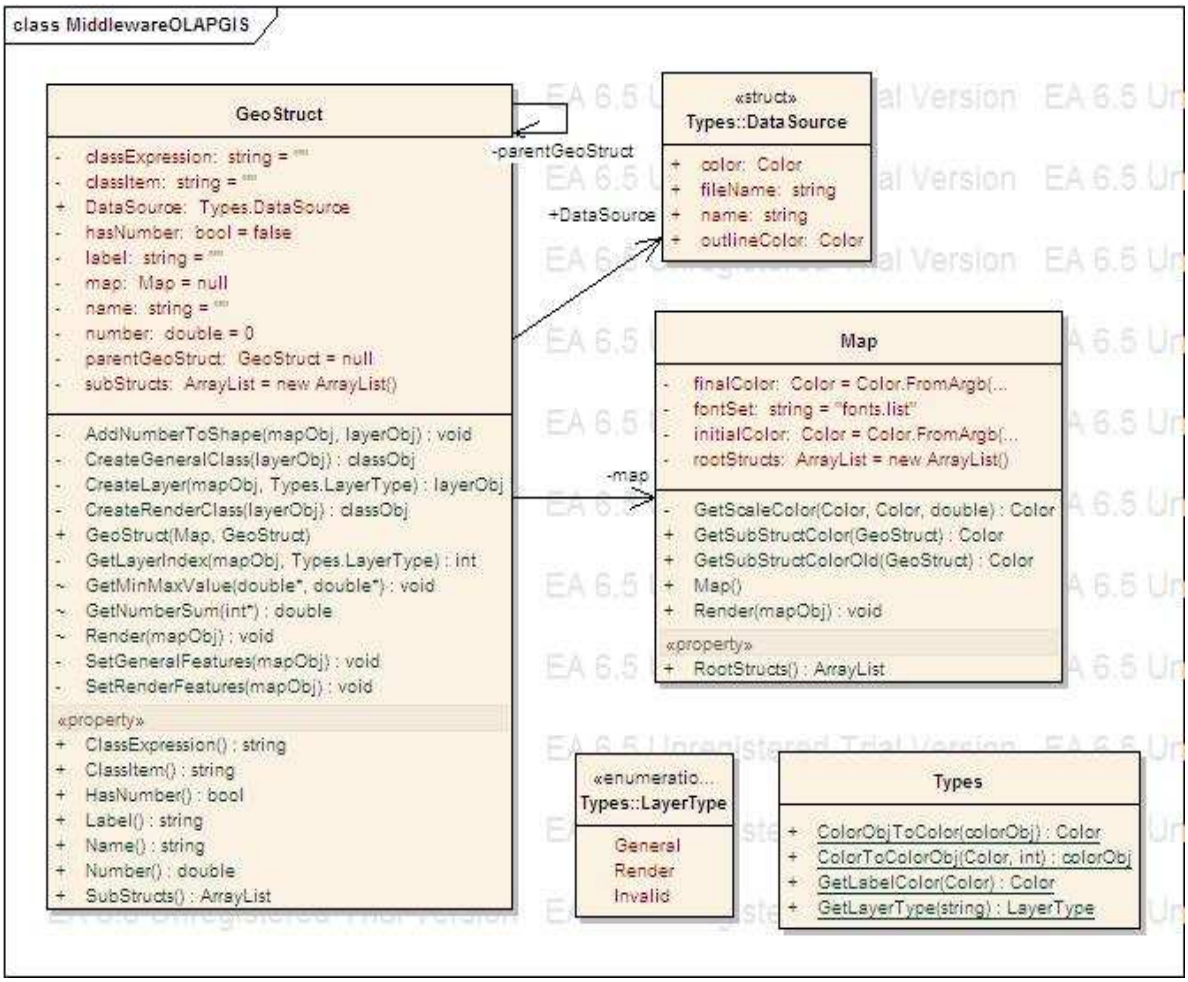

Figura 20. Diagrama de Classes do Middleware

É importante ressaltar que, na fase de decomposição do XMLA, somente os atributos relacionados aos dados de negócio – disponíveis no XMLA – da classe *GeoStruct* são preenchidos. São eles:

- **name**: é o texto que será utilizando como legenda e também servirá como chave para identificação das informações geográficas;
- **number**: no caso de um nodo folha, contém o valor da medida associada ao mesmo;
- **hasNumber**: identifica um nodo folha, para que não haja confusão entre nodos sem valor e nodos com valor zerado;
- **subStructs**: conjunto de nodos filhos de determinado nodo. Somente é preenchido no caso de derivações de uma unidade de análise (e.g., um estado subdividido em macro-regiões);
- **parentGeoStruct**: recebido como parâmetro no construtor da classe, somente é preenchido no caso de derivações de uma unidade de análise (e.g., um município que subdivide um estado);
- **map**: recebido como parâmetro no construtor da classe, representa o mapa que conterá a estrutura, permitindo o acesso a configurações gerais e cálculos baseados também no valor de outros nodos (e.g., para calcular a cor de determinado item é necessário saber a soma total de valores).

Preenchidas as informações obtidas da decomposição do arquivo XMLA, cabe ao *middleware* preencher os demais atributos usando a Base de Metadados. Nesta situação, a aplicação realiza a iteração na estrutura da classe *GeoStruct*, utilizando os próprios objetos dessa estrutura no preenchimento de seus atributos geográficos. Cada objeto realiza a pesquisa de seus dados na base geográfica,

utilizando para busca a descrição obtida do arquivo XMLA e preenchendo os atributos relacionados a dados geográficos. São eles: *classItem*, *classExpression* e *dataSource* (*fileName*, *color* e *outlineColor*). Por serem atributos relacionados à lógica de geração de mapas, não convém descrevermos um a um, já que sua utilidade será melhor explicada em sessão específica (ver MapServer).

Com informações das duas naturezas (de negócio e geográficas) associadas em um mesmo objeto, é possível iniciar o processo de geração do mapa. Neste fluxo, a aplicação realiza novamente uma iteração pelas estruturas já populadas, criando as camadas e diferenciações necessárias. Como nem todos os itens de determinado contorno podem conter valores (e.g., somente alguns estados de um país podem ser referenciados), a aplicação diferencia dois tipos de camadas:

- **Camadas Gerais**: Não são valorados, ficando como camada de fundo, já que contém o contorno de todos os itens;
- **Camadas *Render*:** Referem-se a itens valorados no resultado da consulta OLAP. São as camadas em que são desenhados os valores, legendas e cores. São desenhados sobre as camadas gerais.

Na lógica da aplicação, cada item, ao gerar seu contorno, cor e legenda, verifica se as camadas de fundo necessárias já estão criadas, caso contrário o fazem. Com a lógica apresentada até aqui, a criação da legenda pode ser efetuada usando um atributo na camada *Render* relacionada. Para a questão da coloração, o algoritmo do *middleware* utiliza-se da estrutura de lista encadeada em que é criada a classe *GeoStruct*, calculando a cor representativa do valor de cada

entidade geográfica de acordo com a proporção desse valor na faixa de valores total retornada pela consulta OLAP, gerando assim um efeito de gradação dos valores na escala de cores.

Até a conclusão deste trabalho, o *middleware* foi apresentado como uma aplicação web. A configuração dos diferentes parâmetros citados durante sua descrição (escala de cores a ser plotada, formato da legenda, arquivo XMLA de origem, etc) são configurados através de um arquivo localizado no servidor web que hospeda a aplicação. Apesar de ainda não apresentar algumas características e funcionalidades – tratadas com mais detalhes nas sugestões de trabalhos futuros (ver *Conclusão*) – todos os itens citados ao longo do trabalho já estão implementados.

O *middleware* foi desenvolvido utilizando a linguagem C#.NET, uma tecnologia Microsoft que compõe a plataforma .NET. A escolha da linguagem foi realizada levando em consideração, basicamente, a familiaridade dos autores com a mesma. Como a idéia é que a aplicação seja disponibilizada na forma de um serviço web, sua linguagem de implementação ocupa papel não tão importante do ponto de vista da arquitetura proposta.

MapServer API

Aplicações SIG têm como objetivo final a geração de mapas. Estes mapas, por sua vez, são gerados a partir de uma série de coordenadas e/ou imagens, que podem estar dispostos em uma gama de diferentes fontes de dados (e.g., coordenadas armazenadas em SGBDs ou em arquivos, imagens armazenadas em SGBDs, superfícies com padrões descritos através de cálculos matemáticos,

etc.). A necessidade da aplicação desenvolvida neste trabalho, entretanto, ia além: os mapas deveriam ser gerados baseados em determinadas fontes de dados – como numa aplicação convencional – porém contextualizados as consultas OLAP enviadas através do formato XMLA. Desta maneira, a geração dos mapas ocorre de maneira dinâmica, relacionando os traçados necessários as diferenças dimensões retornadas pelo XMLA.

Na implementação do servidor de mapas, optou-se pela utilização do MapServer, um ambiente de desenvolvimento de aplicações geográficas voltado para a web. A escolha foi altamente influenciada pelo fato da ferramenta disponibilizar a possibilidade de acesso as suas funcionalidades através de uma API, provendo a flexibilidade necessária à arquitetura proposta.

A *MapServer API* é composta por um conjunto de classes que permitem ao desenvolvedor criar mapas com qualquer funcionalidade existente no modo de interação convencional com a ferramenta: através de uma aplicação CGI parametrizável via arquivo. Os conceitos e a hierarquia utilizados na API são os mesmos do arquivo de configuração, sendo o entendimento de suas principais estruturas fundamental para a compreensão da lógica de implementação descrita anteriormente. A Figura 21 sintetiza a descrição das principais estruturas do MapServer.

Estrutura	Descrição	Principais Atributos
Map	Representa o mapa a ser criado.	SIZE (tamanho do mapa em pixels) EXTENT (pontos extremos do mapa) SHAREPATH (local onde ficam os arquivos de mapa)
Layer	Dentro de um mapa, representa as diferentes camadas criadas.	NAME (Nome do layer) DATA (Arquivo de mapa utilizado) STATUS (Indica se o layer está ativo) TYPE (Linha, ponto, polígono, etc) CLASSITEM (Define o atributo usado para diferenciar Classes)
Class	Dentro de um layer, indica a maneira com que determinada forma será desenhada.	NAME (Nome da classe) EXPRESSION (Relaciona o grupo dentro do CLASSITEM do Layer) TEXT (Substitui a legenda)
Style	Dentro de um Class, indica as cores com que o mesmo será desenhado.	COLOR (Cor de preenchimento) OUTLINECOLOR (Cor do contorno)
Label	Dentro de um Class, indica como suas legendas serão desenhadas.	COLOR (Cor da fonte) FONT (Tipo da fonte) SIZE (Tamanho da fonte) POSITION (Posição dentro da forma)

Figura 21. Principais Estruturas do MapServer

Para realizar o desenho dos mapas, o MapServer acessa arquivos do tipo *ESRI Shape. Arquivos ESRI Shape* (SHP), que armazenam os dados geográficos em formato vetorial. O formato SHP foi criado pelo *Enviromental System Research Institute* (ESRI) para armazenar pontos, polígonos, linhas e outros objetos simples. O formato prevê três tipos de arquivo:

- **Arquivo Principal (.shp)**: armazena as formas vetorialmente descritas;
- **Arquivo Dados (.dbf)**: possui atributos que descrevem as diferentes formas existentes no arquivo principal;
- **Arquivo Índice (.shx)**: realiza a indexação das descrições existentes no arquivo de dados e das formas do arquivo principal.

O MapServer suporta também outras formas de armazenamento dos dados relacionados aos mapas - como banco de dados, por exemplo. Para a implementação da aplicação deste trabalho, foram levadas em consideração a velocidade de consulta além da facilidade de utilização dos arquivos no formato SHP.

6. CONCLUSÃO

A utilização de consultas OLAP sobre dados multidimensionais para análise de dados e geração de informação estratégica é tida hoje como uma necessidade no mundo dos negócios. Como muitas das informações utilizadas nestas consultas são relacionadas a dados geográficos (e.g., países, estados, municípios), sua visualização em formato de mapa aumenta a eficiência e a eficácia das análises.

Diante disto, surgiram nos últimos anos diversas propostas para realizar a integração entre OLAP e SIG, oferecendo geração de mapas dinâmica, baseada nos dados consultados via OLAP. Grande parte das soluções encontradas, porém, são extensões a produtos OLAP já existentes ou projetos cuja adoção pelo mercado tende a ser mais lenta, por redefinirem arquiteturas hoje já consolidadas (e.g., OLAP).

Neste contexto, foi definida uma arquitetura aberta que combina componentes de mercado usando um *middleware* baseado em padrões de sistemas abertos das áreas de OLAP e SIG para realizar a ligação entre as duas tecnologias. A arquitetura foi proposta tendo em mente a compatibilidade com formatos e protocolos tidos hoje como padrão pelo mercado, o que torna sua implementação em arquiteturas atualmente consideradas típicas e viáveis na prática.

Na implementação relativa a este trabalho, foi desenvolvido um *middleware* capaz de, utilizando um arquivo XMLA e uma base de metadados, gerar mapas relacionados aos dados gerados por consultas OLAP, definindo dinamicamente a

coloração de cada uma das entidades geográficas analisadas, com base em uma faixa pré-configurada, permitindo assim que faixa de valores das medidas sejam facilmente distinguidas no mapa, acrescentando o valor esperado à análise.

A contribuição deste trabalho à integração entre as tecnologias OLAP e SIG reside principalmente na proposta de uma arquitetura adaptável às existentes hoje no mercado e que permite acoplar facilmente componentes disponibilizados atualmente por diferentes fornecedores. Ao contrário de outros trabalhos discutidos, o foco do middleware apresentado é prático, resultando numa aplicação que, com algumas modificações, citadas a seguir como possibilidades de trabalhos futuros, pode ser adaptada de maneira relativamente simples aos atuais cenários OLAP encontrados no mercado.

Como este trabalho trata da proposta de uma arquitetura que possui um grau alto tanto de complexidade quanto de inovação em parte de seus componentes, uma série de trabalhos futuros pode refinar e prover à arquitetura a maturidade e versatilidade necessárias para sua consolidação. Neste ponto, é muito importante efetuar a modificação de um servidor de interface OLAP para adaptação ao fluxo adicional proposto de envio de XMLA e retorno de imagem de mapa para apresentação ao usuário. Cabe ainda, a adaptação da aplicação criada para a forma de um serviço web e compatível com as especificações OGC. Estudos sobre o a interação dinâmica do cliente com mapas retornados (e.g., utilizando linguagens de marcação como SVG e plug-ins) também são fundamentais. Vale, ainda, ressaltar a importância da adoção ou definição de um formato que integre informações de negócio e geográficas na comunicação entre o servidor de interface e o servidor de mapas, tendo em vista a provável

consolidação de tal abordagem no longo prazo.

REFERÊNCIAS

1. Bimonte, S., Tchounikine, A., Miguel, M. (2005) **Towards a Spatial Multidimensional Model**. DOLAP'05, Bremen, Alemanha, Novembro, 2005.

2. Chaudhuri, S., Dayal, U. (1997) **An overview of data warehousing and OLAP technology**. SIGMOD Record, Vol. 26, No. 1, New York, NY, USA. pp 65 a 74, Março, 1997.

3. Chaudhuri, S., Dayal, U. (1997) **Data warehousing and OLAP for decision support**. Proceedings of the 1997 ACM SIGMOD international conference on Management of data, Tucson, AZ, USA. pp 507 a 508.

4. Coliat, G. (1996) **OLAP, Relational, and Multidimensional Database Systems**. SIGMOD Record,AC, USA. Vol. 25, No. 3, Setembro, 1996

5. Ferreira, A.C.F.(2002), **Um Modelo para Suporte à Integração de Análises Multidimensionais e Espaciais**. Rio de Janeiro: UFRJ/IM/NCE. Abril,2002.

6. Forsman, S. (1997) OLAP Council White Paper.

7. Frozza, A. A, Mello, R. S. (1996) **Um Método para Determinar a Equivalência Semântica entre Esquemas GML**. GEOINFO, 1996

8. Harinarayan, V., Rajaraman, A., Ulman, J. D. (1996) **Implementing Data Cubes Efficiently**. Proceedings of the 1996 ACM SIGMOD international conference on Management of data, Vol. 26, No. 2, Montreal, Quebec, Canada. pp 205 a 216, Junho, 1996.

9. Hyde, J. Pentaho **Analysis Services: Architecture**. Disponível em <http://mondrian.pentaho.org/documentation/architecture.php>. Acesso em 14 março 2007.

10. Inmon, W. H. (1996) **Building the Data Warehouse**. (ed) John Wiley & Sons, Inc., New York, NY, USA.

11. Kimball, R. (1996) **The Data Warehouse Toolkit : Practical Techniques for Building Dimensional Data Warehouses**. (ed) John Wiley & Sons, Inc., New York, NY, USA.

12. Microsoft Corporation. **MDX**. Disponível em < http://msdn2.microsoft.com/en-us/library/aa216767(SQL.80).aspx >. Acesso em 14 março 2007.

13. OpenGIS. **Opengis Specfications**. Disponível em < www.opengeospatial.org/standards >. Acesso em 3 de junho de 2007.

14. Penna, R. A. C., Reis, A. S., Cavalcanti, M. **Informações Geográficas Integradas à Inteligência do Negócio**.

15. Rao, F. et al (2003) **Spatial Hierarchy and OLAP-Favored Search in Spatial Data Warehouse**. DOLAP'03, Lousiana, EUA, Novembro, 2003.

16. Sampaio, M., Sousa, A., Baptista, C. (2006) **Towards a Logical Multidimensional Model for Spatial Data Warehousing and OLAP**. DOLAP'06, Virginia, EUA, Novembro, 2006.

17. Simba Technologies Inc. **XML for Analysis**. Disponível em <http://www.xmlforanalysis.com>. Acesso em 14 março 2007.

18. Silva, J., Times,V. C., Salgado, A. C (2006). **An Open Source and Web Based Framework for Geographic and Multidimensional Processing.**

19. Stefanovic, N. (1997) **Design and Implementation of On-Line Analytical Processing (OLAP) of Spatial Data**. (Universidade de Belgrado, Sérvia, Setembro, 1997.

20. XMLA.org. **XMLA Frequently Asked Questions**. Disponível em < http://www.xmla.org/faq.asp >. Acesso em 14 março 2007.

21. Moss, L., Atre, S. (2003) **Businnes intelligence roadmap: the complete project lifecycle for decision support applications**. (ed) Addison-Wesley Professional, Boston, MA, USA.