Ferramenta para integração de modelos multidimensionais

e

softwares de visualização de séries temporais

Gustavo Bianchi Maia

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO DE MESTRE
EM
CIÊNCIAS

Área de concentração: Ciência da computação
Orientador: Prof. Dr. João Eduardo Ferreira

-- São Paulo, Fevereiro de 2007 --

Sumário

Índice de Figuras

Resumo

Após o aparecimento dos bancos Analíticos e dos modelos Multidimensionais, que enfatizam o armazenamento sumarizado de uma informação consolidada, as empresas ganharam uma poderosa ferramenta de análise, porém tais sistemas ainda não foram capazes de extrair conhecimento, ou seja, gerar informações potencialmente úteis sem a necessidade de uma hipótese inicial por parte do usuário.

Dentre as inúmeras formas de análise, uma que tem se provado muito rica, por estar presente em quase todo tipo de sistema, é a análise temporal, ou seja, a busca de padrões em um banco de dados baseados em uma série temporal. Desta forma ciclos ou eventos que se repetem no tempo são identificados, medidos e analisados, ajudando os sistemas a não somente entender o passado (histórico da informação), como predizer o futuro (análises de estimativas).

Este trabalho apresenta uma ferramenta de domínio público para a integração de softwares de visualização de séries temporais e modelos multidimensionais, para prover não apenas a apresentação dos resultados de maneira mais clara e concisa, mas ainda utilizar todo o poder de análise de um ambiente analítico e da riqueza da informação adquirida pela análise temporal.

Abstract

After the appearance of the analytical databases and the multidimensional models that emphasize the summarized storage of consolidated data, the companies acquired a powerful tool for analysis, but those systems still can't extract knowledge, meaning they can not generate information potentially useful without the need of an initial hypothesis from the user.

Among all alternatives of analysis, one that proved to be very rich for existing in almost all systems is the temporal analysis, this is, the search for patterns in a database based on a time series. In this way cycles or events that repeat over the time are being identified, measured and analyzed, helping systems not just to understand the past (historical information), but to predict the future (estimative analysis).

This work presents a public domain tool to integrate time series visualization software and multidimensional models, to provide not only to help the visualization of the results in a more clear and precise way, but to use the whole power of analysis of an analytical environment and the wealth of the information acquired by the temporal analysis.

1) Introdução e Justificativa

Warehousing* é uma técnica utilizada para recuperação e integração de dados a partir de fontes distribuídas, autônomas, e possivelmente heterogêneas [ZGMHW94].

Estes dados são armazenados em um grande depósito denominado data warehouse**. Um data warehouse sumaria os dados que são organizados em dimensões, disponibilizando-os para consultas e análises através de aplicações OLAP (On-Line Analytical Processing) e sistemas de suporte à decisão [GM96].

A implementação de um data warehouse consiste basicamente da coleta, limpeza e armazenamento de informações provenientes de diversas fontes através de transferências periódicas de dados.

Após sua criação e primeiro carregamento dos dados, o data warehouse pode receber atualizações incrementais, que devem manter a sincronia com o banco operacional, transformando-o em um imenso banco de dados.

Como os dados são introduzidos em um data warehouse ao longo do tempo, é quase natural enxergarmos esta base como uma série de informações armazenadas no tempo. Tal ambiente é muito rico para análises de séries temporais, porém acrescenta alguns agravantes aos métodos comumente utilizados, como:

- O modelo multidimensional propicia a geração de um grande número de séries temporais;
- Modelos de visualização simples e às vezes pouco práticos para análises temporais;
- Dificuldade de integração entre ferramentas multidimensionais e sistemas especializados em análise das séries temporais.

---

* *O termo Warehousing não tem tradução para o português*

** *O termo Data warehouse, que pode ser traduzido para armazém de dados, é sempre encontrado na literatura escrita em português, na sua forma original, em inglês*

1.1) Justificativa

Conforme apresentado em [LKL04], o olho humano é considerado a mais poderosa ferramenta de data mining. Outros trabalhos também evidenciam isso, tal como [Kei02], [Shn96], [Tuf83]. Porém pouco trabalho foi desenvolvido no que se refere à visualização de séries temporais com grandes volumes de dados.

Sendo assim, um ambiente que consiga prover interfaces capazes de utilizar todo o potencial do olho humano, seja organizando melhor a informação, retirando o 'ruído' das informações, permitindo operações como seleção, filtragem, zoom (detalhamento), entre outras; terá um inestimado poder de análise.

As ferramentas de visualização de séries temporais encontradas normalmente utilizam como entrada um arquivo de dados contendo em uma única linha contínua, inúmeros dados referentes a amostras da informação sendo medida e coletados ao longo do tempo, conforme visualizado na figura 1.

>data1.dat
125 178 234 133 189 243 342 175 154 179 235 130 181 244 301 107 ...
Figura 1 – Exemplo de arquivo de entrada de dados por uma linha contínua.

Outras ferramentas utilizam diferentes arquivos de entrada dos dados, como por exemplo o descrito na figura 2, em a) uma linha adicional contendo o instante no qual aquele dado foi coletado e incluído; em b) as informações vêm em pares de 'instante da coleta' - 'informação coletada'. Neste exemplo um símbolo de "pipe" '|' foi utilizado como separador das informações, porém inúmeros formatos podem ser adotados.

a) *>data2.dat*

2005-11-08 11:33:01 | 2005-11-08 16:48:07 | 2005-11-09 10:33:21 | ...

R$75,00 | R$12,50 | R$300,00 | ...

b) *>data3.dat*

2005-11-08 11:33:01 R$75,00 | 2005-11-08 16:48:07 R$12,50 | 2005-11-09 10:33:21 R$300,00 | ... ...

***Figura 2** – a) Exemplo de arquivo contendo uma série temporal em uma linha, e as informações coletadas em outra linha.*

b) Exemplo de arquivo contendo em uma única linha, as informações do instante de coleta e a informação coletada.

A falta de um padrão para a criação deste arquivo de importação, e as incontestáveis vantagens da utilização de um banco de dados, levaram ao armazenamento e possível utilização de tabelas contendo dados armazenados ao longo do tempo como entrada para ferramentas de visualização. Tais informações são armazenadas conforme descrito na figura 3. É possível exportar a informação para arquivos de dados em diversos formatos, mesmo quando as ferramentas de visualização não possuam uma conexão direta com um banco de dados.

Tempo	Informação
8/11/2005 11:33	R$75,00
8/11/2005 16:48	R$12,50
9/11/2005 10:33	R$300,0
...	...

Figura 3 - Entrada da informação de uma série temporal utilizando uma tabela

Estas são as interfaces de entrada que a maioria das ferramentas de análise possui.

Aplicativos mais avançados permitem a comparação entre duas séries temporais, a entrada da informação é realizada individualmente, ou seja, cada série deve possuir seu próprio arquivo, ou sua própria tabela.

É fato que ao ser capaz de analisar duas séries temporais, um aplicativo é capaz de analisar N séries temporais, basta observá-las duas a duas. Porém este trabalho é árduo e requer um grande cuidado ao unir os resultados. Um exemplo de uma entrada de N séries temporais pode ser extraído da tabela vista na figura 4: temperaturas coletadas de algumas capitais brasileiras.

Capitais	1/2/2006	2/2/2006	3/2/2006	4/2/2006	5/2/2006	6/2/2006	...
São Paulo	24	23	25	26	25	24	...
Rio de Janeiro	28	29	29	30	31	30	...
Brasília	30	29	28	27	28	29	...
Belo Horizonte	28	26	25	25	26	28	...
...	...	...	...	...	...	...	...

Figura 4 – Exemplo de utilização de uma tabela contendo diversas séries temporais

Segundo [Kel94] uma das características que diferenciam um data warehouse dos demais sistemas é que os dados são associados à informação temporal e a períodos de tempo definidos, como fechamentos mensais ou baseados no ano fiscal. Em um modelo multidimensional, conforme [RadD96] é muito raro um modelo dimensional que não inclua a dimensão tempo como uma dimensão fundamental.

Desta forma, toda informação contida em um data warehouse estará relacionada ao tempo, pois cada linha armazenada na tabela fato conterá uma chave que a relaciona com a dimensão tempo. Vale ressaltar que um modelo multidimensional pode conter inúmeras dimensões de tempo. Como exemplo, uma loja que contenha dimensões como: data do pedido, data de entrega, data de fabricação do produto, data de validade do produto, entre outras.

Sendo assim pode-se identificar que cada métrica ao ser relacionada com uma dimensão tempo, resultará em uma série temporal a ser analisada. Uma consulta da medida

"qtd vendida", ao ser analisada junto à dimensão "data da venda", resultará em uma série temporal de quantidade vendida por data da venda, como pode ser visto na figura 5.

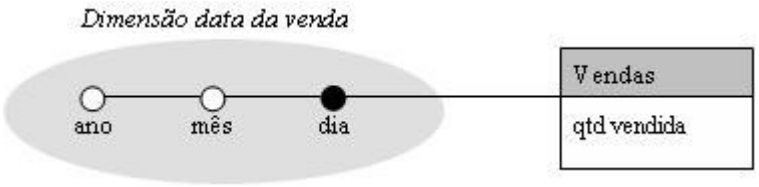


Figura 5 - Exemplo de uma consulta a um modelo multidimensional que resulta em uma série temporal

Para cada nova dimensão adicionada à consulta, e para cada elemento presente na hierarquia daquela dimensão, várias séries serão criadas. Isto resulta em um crescimento exponencial no número de séries temporais presentes em um data warehouse. Como apresentado na Figura 6, considere uma dimensão cliente, com N clientes, adicionada à consulta, isso resultará em 1 x N séries temporais (número de medidas utilizadas multiplicado pelo número de elementos da dimensão adicionada), se uma outra dimensão de produto, com P produtos, for adicionada, conseqüentemente teremos 1 x N x P séries temporais.

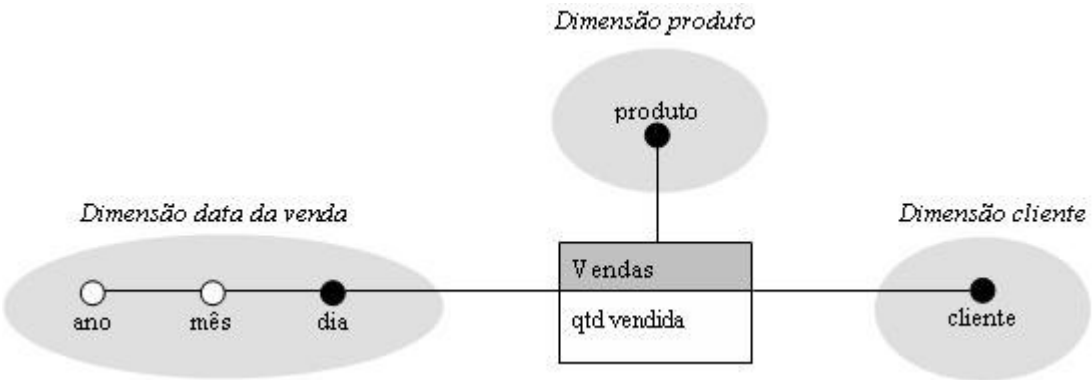


Figura 6 - Exemplo de uma consulta a um modelo multidimensional que resulta em inúmeras séries temporais

Com um número muito grande de séries temporais, com crescimento exponencial relativo ao número de elementos em cada dimensão, é quase impossível gerar as entradas individuais necessárias para a análise destas séries temporais pelas ferramentas de visualização de séries temporais apresentadas.

1.2) Objetivo

O objetivo deste trabalho é a construção de uma ferramenta para prover a visualização de séries temporais em modelos multidimensionais. Esta ferramenta disponibilizará várias interfaces para a identificação visual de séries temporais.

Com este objetivo atendido, os profissionais das áreas de gestão e tomada de decisão poderão utilizar um recurso adicional para a melhor identificação e utilização de modelos temporais.

2) Data warehouse

Segundo [AV98], um data warehouse existe para prover dados para análises relacionadas à gestão e tomada de decisões. No entanto esta função contrasta com o modelo transacional, que visa a normalização dos dados, armazenamento e recuperação da informação pontual (eventos individuais obtidos por sistemas OLTP – *On Line Transaction Processing*).

Um data warehouse é construído para responder questões que não estão limitadas às transações individuais, sendo capaz de responder perguntas como: "qual o valor dos produtos individuais de uma certa venda?" ou "qual o saldo da conta de um dado cliente?"; como também: "Quais os produtos que mais foram vendidos no mês passado?", "Qual a flutuação das contas de meus clientes devedores nos últimos seis meses?".

De acordo com [Kel94], um data warehouse pode ser definido em termos de seis características básicas que o diferenciam de outros sistemas corporativos. Os dados em um data warehouse são:

1. Separados dos sistemas transacionais e populados a partir destes;
2. Disponíveis, na sua totalidade, para a atividade de serem interrogados pelos usuários de negócio;
3. Integrados para ser uma base única e padrão para o modelo da empresa;
4. Associados à *informação temporal* e a períodos de tempo definidos, como fechamentos mensais ou baseados no ano fiscal;
5. Orientados por assunto, ou seja, organizado para descrever o ambiente do negócio;
6. Acessível aos usuários que tenham conhecimento limitado de sistemas computacionais ou estruturas de dados.

2.1) Arquitetura de um data warehouse

Na arquitetura proposta por [SMKK98] existe uma clara separação entre as quatro camadas de um data warehouse, conforme apresentado na figura 7, são elas:

1. Ferramentas de apresentação dos resultados;
2. Servidores OLAP (Molap / Rolap / Holap)
3. Banco de dados analíticos (data warehouse), e ferramentas de extração, limpeza e integração dos dados.
4. Banco de dados operacionais

O caminho de extração e disponibilização da informação é descrito como: partindo dos bancos operacionais, os dados são extraídos e validados pelos wrappers[*]. Os wrappers são programas de computador que fazem a comunicação entre o integrador do banco analítico e os bancos operacionais, refletindo no primeiro quaisquer mudanças que ocorram no segundo. A seguir os dados provenientes de várias fontes são integrados e finalmente armazenados no data warehouse. Depois disso eles podem ser consultados por um servidor OLAP que organiza os dados em estruturas multidimensionais propiciando acesso às ferramentas de datamining**, planilhas, geradores de relatórios, e ferramentas de análise. Os meta dados depositados no OLAP armazenam detalhes sobre a criação, gerenciamento e utilização do data warehouse.

---

[*] *O termo wrapper foi mantido como no original em inglês.*

** [*] *O termo datamining, que pode ser traduzido como mineração de dados, foi mantido como no original em inglês.*

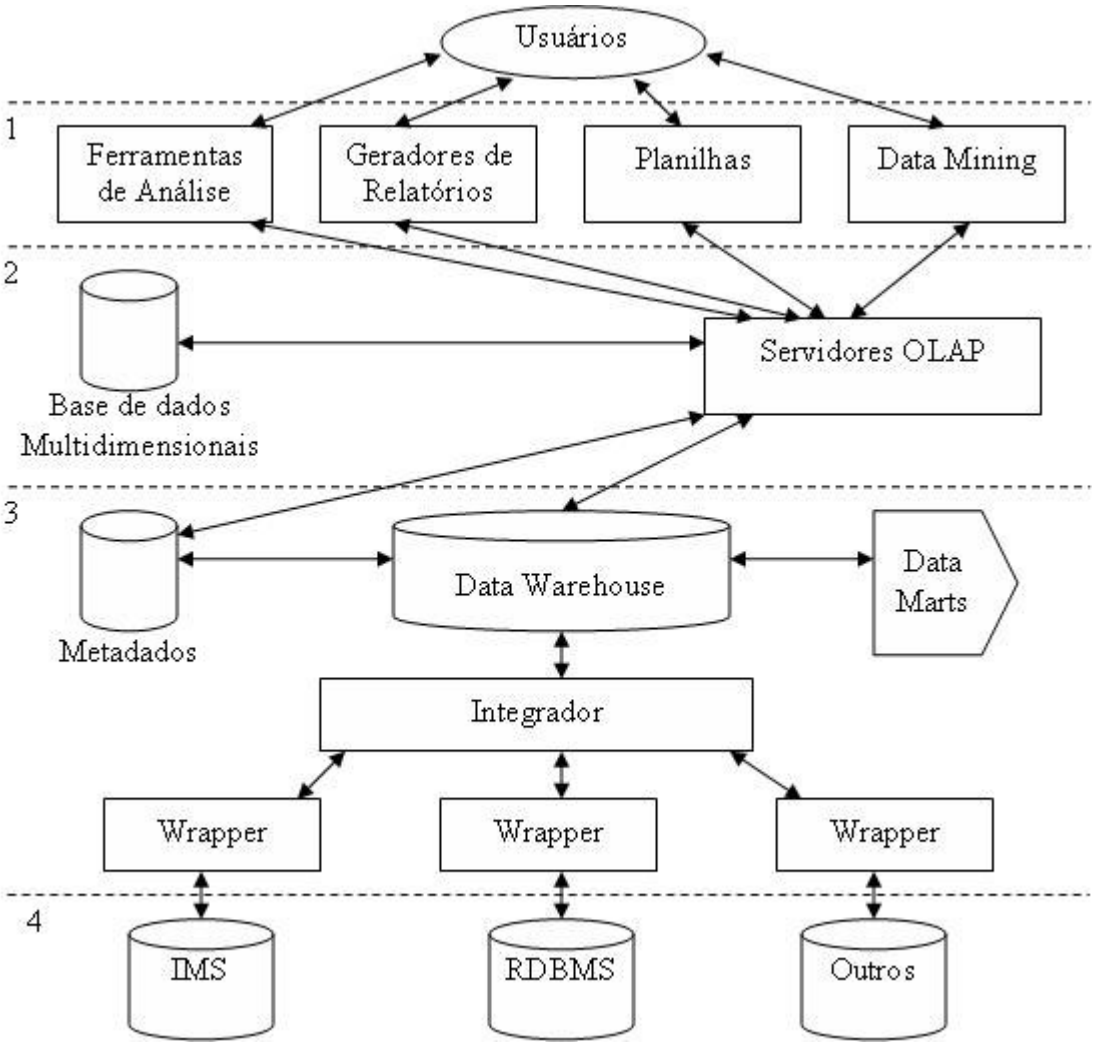


Figura 7 - Arquitetura de um data warehouse proposta por [SMKK98]

Segundo [Sei01], os servidores OLAP podem ser organizados em três diferentes estruturas: MOLAP (Multidimensional On Line Analytical processing), que armazena a informação relacional em cubos pré-construídos, figura 8; ROLAP (Relational On Line Analytical Processing), que se utiliza de estruturas de agregação e consultas feitas diretamente ao banco de dados, figura 9; HOLAP (Hybrid On Line Analytical Processing), que tenta aperfeiçoar o uso das duas estruturas anteriores, figura 10.

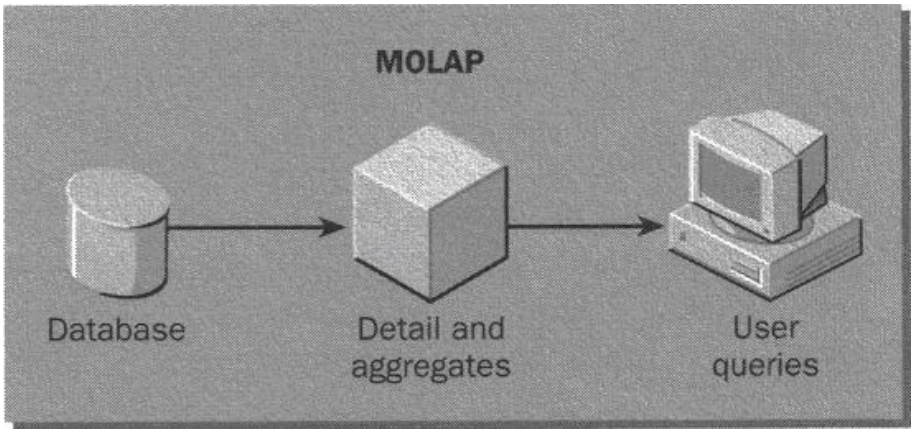


Figura 8 - Estrutura MOLAP retirada de [Sei01]

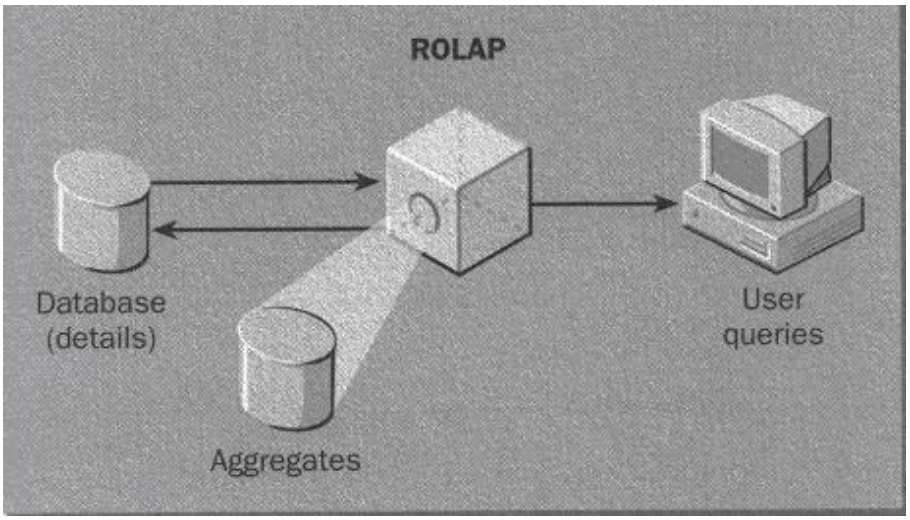


Figura 9 - Estrutura ROLAP retirada de [Sei01]

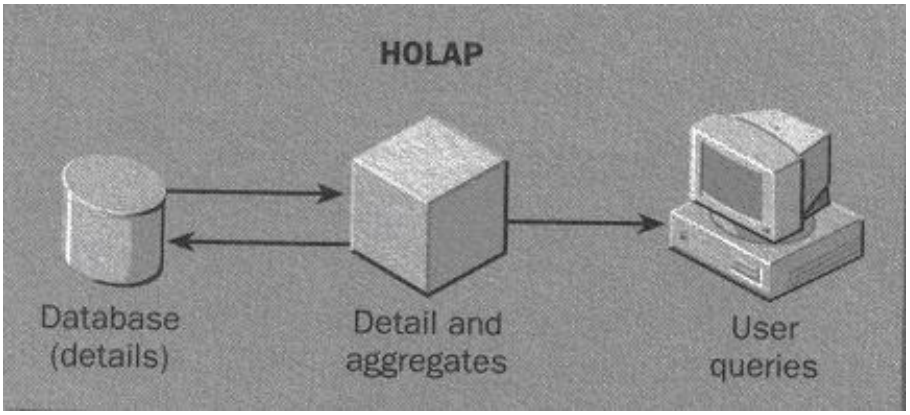


Figura 10 - Estrutura HOLAP retirada de [Sei01]

2.2) O modelo multidimensional

Diferente da modelagem entidade-relacionamento, o modelo multidimensional (ou simplesmente dimensional), se baseia em métricas e parâmetros para ser construído. As métricas constituem tudo aquilo que se deseja medir, contar, somar. Elas também são conhecidas como fato, por estarem presentes na tabela fato, a principal tabela em um modelo dimensional. Os parâmetros, aqui denominados dimensões, são as maneiras, ou pontos de vista, em que as métricas podem ser observadas. Pode-se fazer um paralelo com um gráfico, veja figura 11, de três dimensões *time, geography e products*[*]. Cada eixo representa uma dimensão, os pontos representados no gráfico são nossas métricas, cada valor possui sua própria projeção em cada um dos eixos.

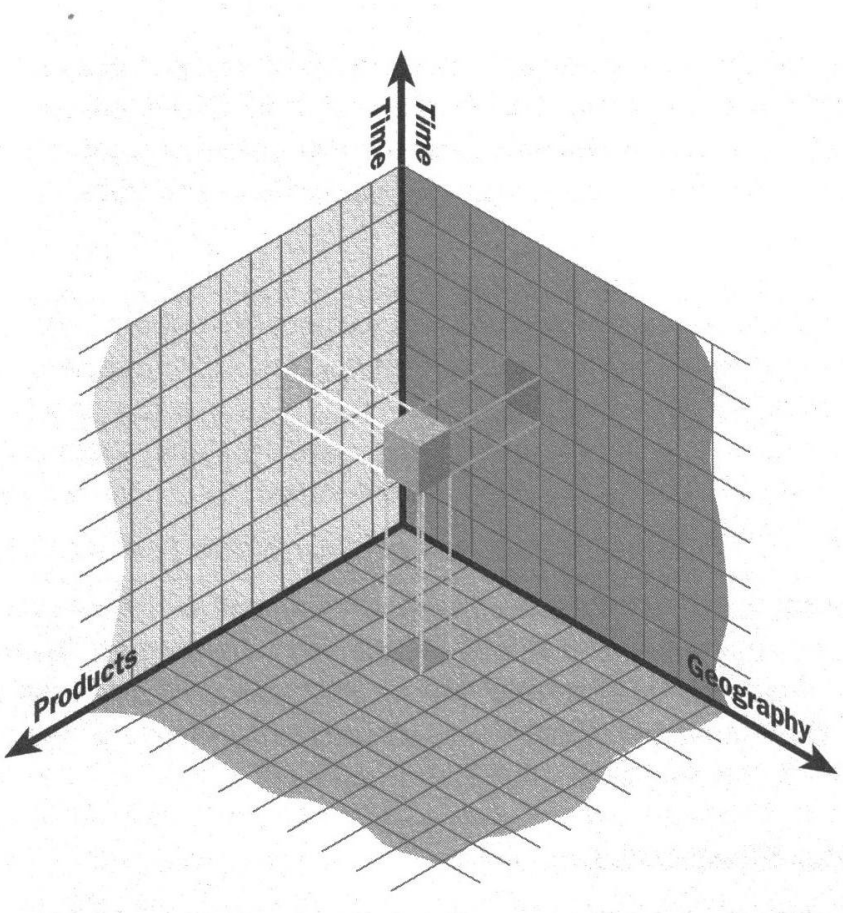


Figura 11 - Medida representada nas dimensões time, geography e products [Sei01]

[*] *Os nomes das dimensões:time, geography e products foram mantidos como no original, em inglês. Sua tradução representa respectivamente: tempo, geografia e produtos.*

Observando a figura 12, vê-se um exemplo de uma estrutura multidimensional para vendas. Este modelo é de fácil compreensão, pois segundo [IFT01] “as coisas que eu avalio” estão na parte central, e “as maneiras de olhar para ela” estão nos quadros em volta.

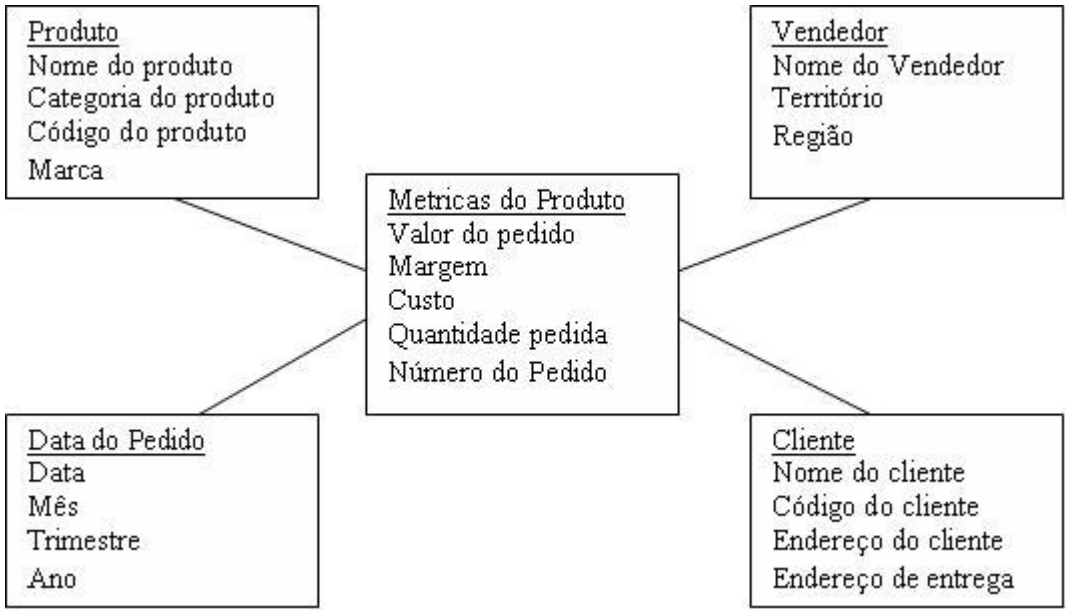


Figura 12 - Modelo multidimensional para um processo de pedidos.

O modelo dimensional foi formalmente definido por [BPT97], sua estrutura pode ser representada por um diagrama entidade-relacionamento, como na figura 13.

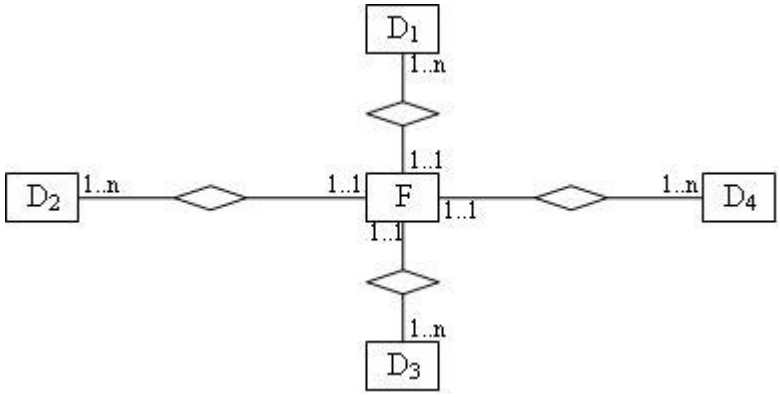


Figura 13 - Representação Entidade-Relacionamento de um banco de dados multidimensional

Definição 1

Um banco de dados multidimensional é uma coleção de relações D_1, D_2, ..., D_n, F, na qual:

- Cada D_i é uma tabela dimensão (ou hierarquia de tabelas de dimensões), e cada uma possui um identificador único por tupla, d_i é chave primária de D_i.
- F é uma tabela fato, e conecta todas as tabelas D_1, D_2, ..., D_n, o identificador de F é composto pelas chaves estrangeiras d_1, d_2, ..., d_n, de todas as tabelas dimensões conectadas. Esta tabela também contém um conjunto de valores V que representam os dados que podem ser sumariados.

Definição 2

Seja D uma tabela dimensão com identificador d. Attr(D), uma hierarquia de atributos em D, é um conjunto de dependências funcionais $FD_D = \{fd_0, fd_1, ..., fd_n\}$, na qual cada fd_i é caracterizado por dois conjuntos de atributos $A^L_i \subset Attr(D)$ e $A^R_i \subset Attr(D)$ (denominados respectivamente de lado esquerdo e direito da dependência); a dependência é representada por fd_i: $A^L_i \rightarrow A^R_i$.

Cada dependência funcional fd_i é uma restrição ao conteúdo da tabela dimensão D, sendo que: para cada par de tuplas t_1 e $t_2 \in D$, $t_1[A^L_i] = t_2[A^L_i] \rightarrow t_1[A^L_i] = t_2[A^R_i]$. Uma dependência fd_0 com $A^L_0 = \{d\}$ e $A^R_0 = \{Attr(D) - d\}$ estará sempre presente em FD_D. As dependências funcionais devem ser acíclicas, isto é, o grafo obtido pelo desenho de um arco partindo de a_x e chegando a a_y, deve ser acíclico, se $\exists fd_i \in fd_D \mid a_x \in A^L_i \wedge a_y \in A^R_i$.

Definição 3

Uma hierarquia de atributos do banco de dados multidimensional FD_{DB} é a união das hierarquias de atributos DF_{DJ} de todas as dimensões D_j existentes no banco de dados multidimensionais.

2.3) Modelagem de um data warehouse

Segundo [Tan97] os elementos de uma estrutura dimensional são: a tabela fato, que contém dados históricos temporais organizados por chaves indexadas, chaves estas que estão relacionadas cada uma a uma tabela dimensão.

Uma tabela dimensão deve conter toda informação relacionada a um domínio da aplicação. Por exemplo, uma dimensão de produto deve conter toda a listagem de produtos possíveis em um sistema de vendas; a dimensão de clientes deve conter todos os clientes, e assim por diante.

A tabela fato contém as métricas ou fatos a serem analisadas dentro de um escopo. Cada linha desta tabela está diretamente relacionada a todas as demais dimensões do sistema. Por exemplo, se a tabela fato representar vendas, uma única venda então, quando vista separadamente, deverá conter uma chave de relacionamento com a tabela produto, nos indicando qual produto foi vendido, assim como chaves para as demais dimensões.

Vejam na figura 14 a modelagem de um esquema de pedidos, que utilizou o modelo denominado *Star Schema*, proposto por [Kim96].

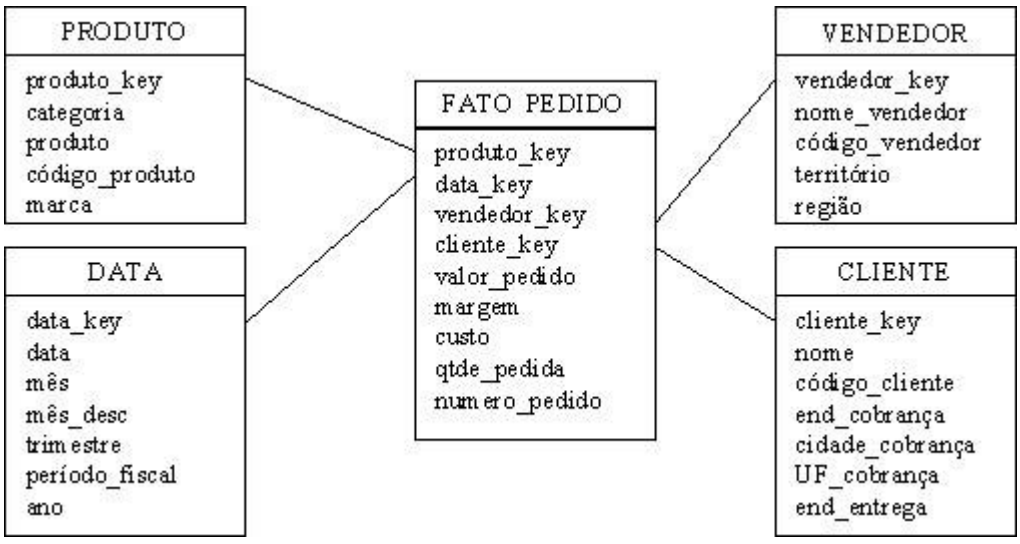


Figura 14 - Modelagem para um esquema de pedidos

Podemos classificar a modelagem de um data warehouse em quatro passos:

1. Definição do escopo
2. Definição da granularidade das informações da tabela fato
3. Escolha das dimensões
4. Enumeração das métricas ou fatos

2.3.1) Definição do escopo

Como já dito, um data warehouse, quando bem modelado, deve representar a forma de pensar dos especialistas no negócio. A definição do escopo tem como objetivo limitar a área de conhecimento de um sistema, ou seja, estabelecer um foco para o sistema.

É muito comum em sistemas grandes e complexos que vários formas de organização dos dados sejam criadas, cada uma com seu respectivo data warehouse e consequentemente, tabelas fato. Por exemplo, um sistema de faturamento pode escolher as vendas realizadas, da mesma forma que um sistema financeiro pode escolher os líquidos recebidos pelas vendas. Embora muito parecidos, o escopo de cada sistema pode variar significativamente.

2.3.2) Definição da granularidade

Segundo [AV98], a granularidade (ou simplesmente grão) da tabela fato representa o nível de detalhamento desejado, e todas as linhas armazenadas devem obrigatoriamente conter o mesmo nível de detalhamento.

Como exemplo imagine uma tabela fato para vendas, se o grão for a venda, cada item registrado na tabela fato conterá um cliente, uma data de venda, porém não conterá apenas um produto, pois o nível de detalhamento de vendas é menos detalhado que o de item da venda. Nesta situação uma venda teria no máximo uma medida sumariada denominada ‘número de produtos’.

Tomando o item da venda como grão, então uma venda com sete produtos registraria na tabela fato sete entradas, cada uma contendo um cliente, uma data de venda, e

um produto. Vale observar que o espaço utilizado por este segundo exemplo é bem maior que o primeiro, porém espaço de armazenamento não é a preocupação principal na construção de um data warehouse.

2.3.3) Escolha das dimensões

As tabelas dimensão contêm informações sobre as dimensões dos dados, devendo ser desenhadas a partir da perspectiva do usuário, pois os valores ali contidos serão posteriormente utilizados para a demonstração dos resultados. As principais funções da tabela dimensão é reunir os atributos que serão utilizados para qualificar as consultas e cujos valores serão utilizados para agrupar e sumariar as métricas [AV98].

O principal atributo na criação das tabelas dimensão é a escolha de sua chave, cada tabela deve conter um valor único. Preferencialmente opta-se por valores genéricos ao invés de valores representativos ou provenientes dos sistemas operacionais, também não é aconselhável a utilização de chaves compostas.

Para cada chave definida, serão associados múltiplos atributos que contenham valores numéricos ou textuais que possam ajudar a descrever a chave, ou melhor, representá-la segundo a perspectiva do usuário. Estas colunas serão também utilizadas para compor os filtros de utilização das medidas da tabela fato.

Os atributos de uma dimensão também podem compor hierarquias, criando níveis mais generalizados. No exemplo da dimensão data, existiria um nível para anos, outro para trimestre, outro para meses e outro para dias.

Porém na modelagem de estruturas dimensionais deve-se resistir ao impulso da normalização [AV98], este processo de separação torna as consultas mais complexas, mais lentas, gerando pouco ganho comparativo em espaço de armazenamento. Como foi visto no exemplo da figura 14, a dimensão data foi apresentada com os atributos data, mês, trimestre e ano, sem normalização.

Como por definição as informações em um data warehouse são um conjunto de dados históricos temporais, vale ressaltar que, segundo [RadD96] é muito raro um modelo dimensional que não inclua a dimensão tempo.

2.3.4) Enumeração das métricas ou fatos

A seleção dos fatos que ajudarão a compor o modelo dimensional, segundo [RadD96] é obtida pela resposta da questão: "o que estamos avaliando?".

A enumeração das métricas está diretamente relacionada às escolhas feitas nos itens anteriores. A métrica deve avaliar o escopo definido, deve respeitar a granularidade, e estar relacionada às dimensões adicionadas ao modelo.

Existem três tipos de métricas que podem ser escolhidas:

- **Métricas completamente aditivas**

São medidas que podem ser sumariadas facilmente, ou seja, seus valores podem ser somados ao longo de qualquer dimensão. Segundo o exemplo de vendas, valores como valor do pedido, quantidade, margem, são medidas completamente aditivas.

Este tipo de métrica permite todos os tipos de operações, como: soma, média, maior, menor e contagem.

- **Métricas semi-aditivas**

São medidas que podem ser sumariadas dependendo da dimensão em que observa. Por exemplo, imagine a medida de saldo de uma conta bancária, este valor é obtido diariamente ao longo do tempo. Caso o usuário possua mais de uma conta, faz sentido a soma dos saldos, porém, não faz sentido somar o saldo da mesma conta ao longo do tempo.

Este tipo de métrica permite todos os tipos de operações, respeitando-se as restrições relativas à dimensão analisada.

- **Métricas não aditivas**

São medidas que não podem ser sumariadas, apresentando, na sua maioria, valores discretos e não contínuos. Como exemplo pode-se citar nomes e descrições.

Este tipo de métrica restringe as operações possíveis, segundo [GMR98], a média, mínimo, máximo e contagem.

Outro componente importante da tabela fato é sua chave primária, que pode ser definida como a união das chaves estrangeiras de todas as tabelas dimensão envolvidas. Cada elemento deve estar representado na tabela fato e em sua respectiva tabela dimensão. A figura 15 demonstra algumas tuplas de uma tabela fato, em destaque encontra-se a chave primária composta.

Tabela Fato

produto_key	data_key	vendedor_key	cliente_key	valor_pedido	margem	custo	qtde_pedido	numero_pedido
501	101	1010	55436	130,00	60,00	70,00	1,00	731
620	113	1020	87654	90,00	50,00	40,00	1,00	732
399	114	2010	20933	150,00	10,00	140,00	3,00	733
502	118	4010	87762	30,00	15,00	15,00	1,00	734

Figura 15 - Chave primária composta de uma tabela fato.

2.4) Modelagem conceitual de um data warehouse

O modelo conceitual denominado *dimensional fact schema* (DF ou esquema de fatos) foi proposto por [GMR98], ele utiliza uma notação gráfica e uma metodologia que resulta em um modelo DF partindo-se de um modelo ER, que serve apenas como uma fonte de entrada dos dados e não uma fonte formal.

Uma argumentação sobre a não utilização do ER foi obtida de [Kim96]: “Os modelos de dados Entidade Relacionamento... não podem ser entendidos pelos usuários e não podem ser navegados de forma proveitosa pelo software de banco de dados. Os

modelos Entidade Relacionamento não podem ser utilizados como base para os data warehouses das empresas".

Os componentes básicos de um DF são: as tabelas fato, que contém o objeto de análise; as dimensões, que determina a granularidade apresentada para representar os fatos; e as hierarquias, que determinam como as instâncias dos fatos podem ser agregadas ou selecionadas.

Uma estrutura em árvore é utilizada para representar o modelo DF, na raiz encontra-se a tabela fato, representada por uma caixa que conterá o nome da tabela fato e as métricas escolhidas. A figura 16 apresenta um fato SALE, que representa as vendas de uma cadeia de lojas, junto com as métricas "qty sold" e "returns", que representam respectivamente a quantidade vendida e o total de lucro.

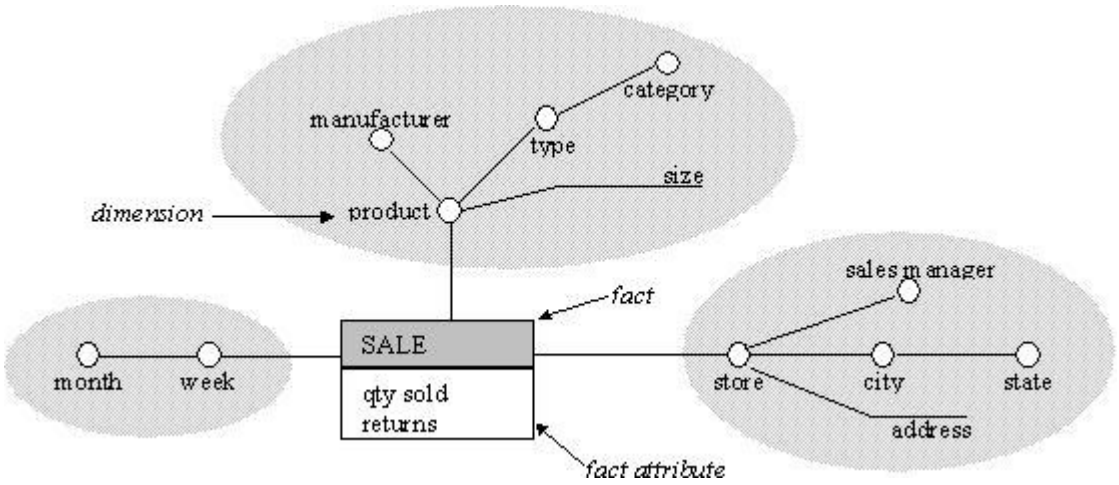


Figura 16 - Modelo dimensional representando as vendas para uma cadeia de lojas [IFT01]

As dimensões e suas hierarquias são representadas em sub-árvores, cada vértice deve estar diretamente ligado à tabela fato, se for uma dimensão, ou à outra dimensão, se for uma hierarquia.

Cada linha representa um relacionamento para-um, desta forma o fato sempre representará o nível mais detalhado possível, cada dimensão ou nível da hierarquia que se afasta da tabela fato contém níveis cada vez menos detalhados.

Os vértices terminais que não contiverem um círculo representam os atributos não dimensionais, servindo apenas para acrescentar alguma informação sobre outro elemento da hierarquia.

Por definição, todas as métricas utilizadas na tabela fato são aditivas, proporcionando assim todo tipo de sumariação. Existem porém as medidas semi-aditivas e não aditivas, que devem ser representadas no modelo por uma linha tracejada partindo da dimensão específica até a medida, deixando registrado no modelo qual o tipo de operação permitida (ou não permitida). A figura 17 mostra a utilização de atributos semi-aditivos.

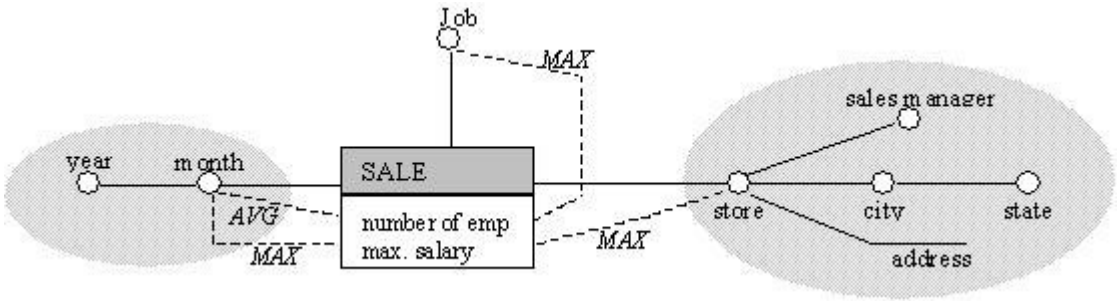


Figura 17 - Representação de uma medida semi-adidiva no esquema de fatos [ITF01].

2.4.1) Modelos de pesquisa para um esquema de fatos

Em um esquema de fatos, uma pesquisa é representada por consultas padrões. No exemplo da figura 18 está representado "a quantidade total vendida e a média de lucro por unidade vendida para cada semana e para cada tipo de produto".

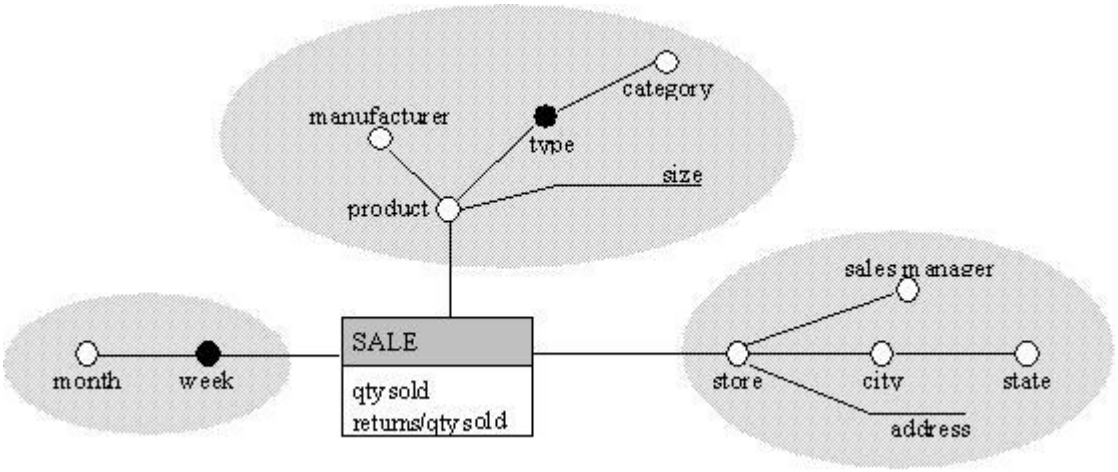


Figura 18 - Representação de um modelo de consulta (query pattern) [ITF01]

As quatro consultas padrões de um ambiente OLAP foram exemplificadas por [MF01] como:

- Pivoteamento: orienta a visão multidimensional, ou seja, escolhem-se as dimensões, hierarquias e medidas para serem analisadas.

Exemplo de pivoteamento: visualização da quantidade vendida por produto por mês

```
SELECT SUM(F.quantidade) as quantidade_vendida, P.nome, D.mês
FROM Fatos as F, Produto as P, Data as D
WHERE F.id_produto =  P.id_produto
        AND F.id_data = D.id_data
GROUP BY P.nome, D.mês;
```

- Rollup: aumenta o nível de agregação, diminui o nível de detalhe sobre uma hierarquia.

Exemplo de rollup, o nível de detalhe aumenta de mês para ano.

```
SELECT SUM(F.quantidade) as quantidade_vendida, P.nome, D.ano
FROM Fatos as F, Produto as P, Data as D
WHERE F.id_produto =  P.id_produto
        AND F.id_data = D.id_data
GROUP BY P.nome, D.ano;
```

- Drill-down: diminui o nível de agregação, aumenta o nível de detalhe sobre uma hierarquia.

Exemplo de drill-down, o nível de detalhe diminui de mês para dia.

```
SELECT SUM(F.quantidade) as quantidade_vendida, P.nome, D.dia
FROM Fatos as F, Produto as P, Data as D
WHERE F.id_produto =  P.id_produto
        AND F.id_data = D.id_data
GROUP BY P.nome, D.dia;
```

- Slice-and-Dice: seleção e projeção, um determinado valor de uma dimensão é selecionada e projetada como filtro de seleção de outras consultas.

Exemplo de slice-and-dice, um produto específico “Garrafa Skol 600ml” é selecionado e utilizado como filtro de uma consulta de quantidades vendidas por cidade.

```
SELECT SUM(F.quantidade) as quantidade_vendida, C.nome
FROM Fatos as F, Cidade as C, Produto as P
WHERE F.id_produto =  P.id_produto
        AND F.id_cidade = C.id_cidade
        AND P.nome = “Garrafa Skol 600ml”
GROUP BY C.nome;
```

2.5) Modelos lógicos e físicos

A concepção dos modelos lógicos, e sua conseqüente implementação física podem ser realizadas utilizando-se os chamados esquemas de modelagem. Os mais comuns são o *star schema* e *snowflake schema*, porém existem variações, que não serão estudadas neste trabalho. Detalhes podem ser encontrados em [ITF01].

2.5.1) O esquema Star

Proposto por [Kim96] este é provavelmente o primeiro esquema utilizado para implementar um data warehouse em um banco de dados relacional.

Neste esquema, cada tabela fato deverá conter:

- Uma chave primária composta pela chave estrangeira de cada uma das dimensões do modelo;
- Métricas associadas ao nível de detalhe definido pela granularidade escolhida que podem ser sumariadas em diferentes níveis de agregação.

Cada dimensão do negócio a ser avaliado deverá conter:

- Uma chave genérica, preferencialmente gerada pelo sistema;
- Uma ou mais colunas de descrição para a dimensão ou para cada elemento da hierarquia, com a possibilidade de se efetuar filtros;
- Um atributo para indicar o nível da hierarquia a que se refere aquela linha.

Conforme apresentado no exemplo da figura 19, que exibe detalhes sobre uma dimensão geografia, não existe uma preocupação com a normalização das tabelas, este é um dos aspectos que diferenciam muito a modelagem de um data warehouse. Outro destaque é o indicador de nível, que pode tanto conter números, quanto valores literais como: “loja”, “cidade”, “estado”, “região”, etc.

geografia_key	região	região_desc	estado	estado_desc	cidade	cidade_desc	loja	nível
1010	1	Sul						
1020	2	Sudeste						
2010	2	Sudeste	10	São Paulo				
2020	2	Sudeste	20	Rio de Janeiro				
3010	2	Sudeste	10	São Paulo	101	São Paulo		
3020	2	Sudeste	10	São Paulo	102	Campinas		
4010	2	Sudeste	10	São Paulo	101	São Paulo	1001	
4020	2	Sudeste	10	São Paulo	102	Campinas	1002	
4030	2	Sudeste	10	São Paulo	102	Campinas	1003	

Figura 19 - Tabela dimensão de geografia, com uma descrição para cada nível da hierarquia.

Uma das dimensões que exige um tratamento especial é a de Tempo, esta utiliza três novos atributos que representam:

- Resolução: tem o mesmo papel que o atributo nível na hierarquia. Contém números ou valores textuais para indicar períodos como “Ano”, “Mês”, etc.
- Seqüência na resolução: indica a ordem temporal dos acontecimentos, não havendo repetições dentro de cada período ou nível na hierarquia.
- Indicador Corrente: Indica que um dado é atual para cada resolução ou nível, deve ser utilizada em conjunto com a coluna seqüência na resolução. Ela é normalmente preenchida com valores simples como “S” ou “N”, e dependendo das condições do negócio, podem indicar o momento em que os dados estarão disponíveis para o usuário.

tempo_key	descrição	resolução	sequencia	corrente	ano	semestre	mês	dia
110	Outubro	Mês	501	S	2004	022004	102004	...
111	Novembro	Mês	502	S	2004	022004	112004	
112	Dezembro	Mês	503	S	2004	022004	122004	
201	Janeiro	Mês	504	S	2005	012005	012005	
202	Fevereiro	Mês	505	N	2005	012005	022005	
14	2 semestre	Semestre	101	S	2004	022004		
21	1 semestre	Semestre	102	S	2005	012005		
1	2004	Ano	1	S	2004			
2	2005	Ano	2	S	2005			

Figura 20 - Tabela dimensão de tempo, com as colunas específicas para este tipo de dimensão.

Na figura 21 pode-se ver um exemplo de um modelo Star com uma tabela fato denominada vendas, e três dimensões, Geografia, Tempo e Produto.

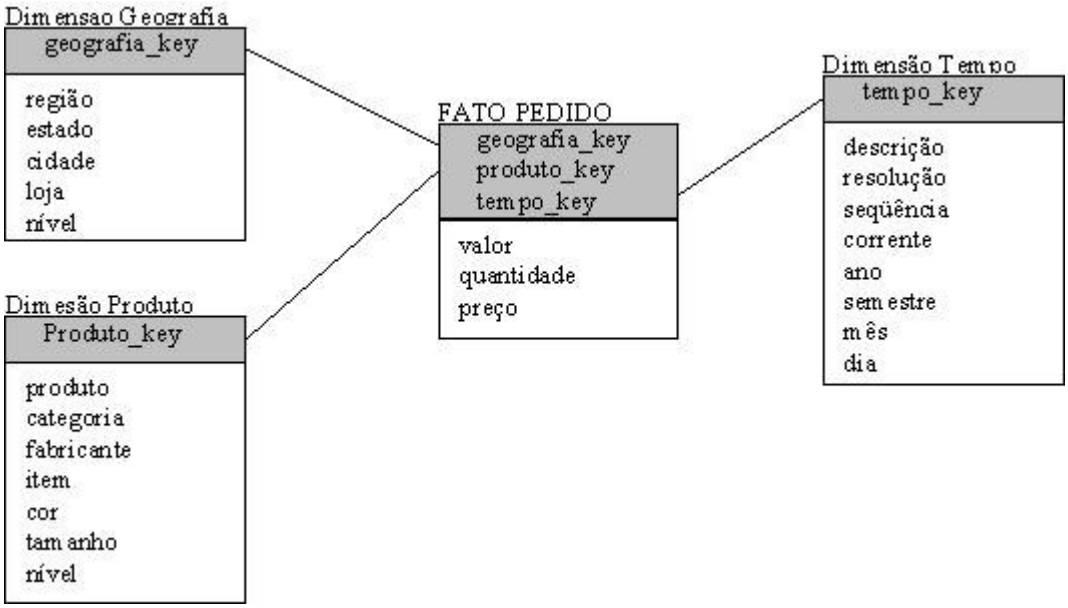


Figura 21 - Exemplo de um data warehouse utilizando o modelo Star clássico

Dentre as vantagens de utilização do esquema Star, pode-se citar: facilidade de entendimento do modelo; ótimo desempenho das consultas, principalmente devido à utilização de chaves genéricas, que geralmente são números inteiros; número reduzido de operações de junção (*join*), já que todos os níveis de uma hierarquia são definidos em uma única tabela por dimensão.

Ele apresenta duas desvantagens, a primeira é com relação ao atributo de nível, é necessário se ter algum conhecimento sobre a estrutura da tabela, antes de se implementá-lo. Ele também apresenta problemas quando novos níveis são incluídos, principalmente quando inseridos entre outros níveis.

O segundo problema é referente ao espaço de armazenamento, pois devido à não normalização, dimensões muito grandes podem gerar uma grande quantidade de informações duplicadas.

2.5.2) O esquema Snowflake: Lookup e Chain

Este esquema utiliza um pouco de normalização nas tabelas dimensão, diminuindo assim o espaço de armazenamento dos dados em troca de um pouco de velocidade e desempenho. Em uma dimensão como a de geografia, uma tabela é utilizada para cada nível hierárquico, loja, cidade, estado e região, eliminando a necessidade do atributo nível.

Existem duas formas de representar este esquema: o *snowflake lookup* contém uma tabela para as chaves da hierarquia e uma tabela adicional para cada descrição necessária, veja um exemplo na figura 22.

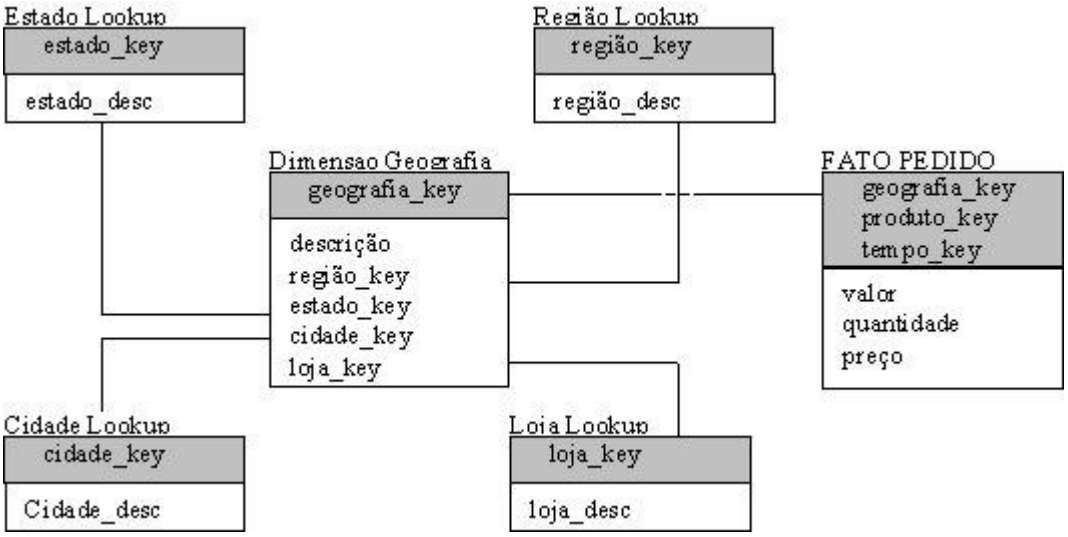


Figura 22 - Representação da dimensão geografia segundo o modelo snowflake lookup

A outra forma denominada *snowflake chain* utiliza uma tabela para cada nível da hierarquia, cada uma contendo sua própria chave primária e a chave de relacionamento com os níveis associados, a figura 23 apresenta um exemplo deste modelo.

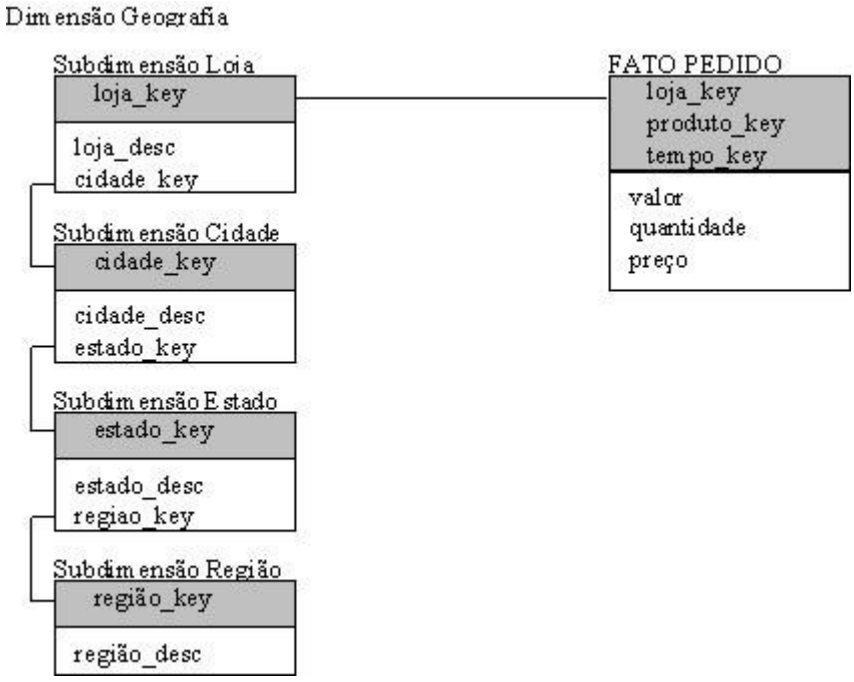


Figura 23 - Representação da dimensão geografia segundo o modelo snowflake chain

Apesar do esquema Star e do Snowflake serem os mais conhecidos, eles não são os únicos, existem muitas variantes, para o star existe o Partial Star [Tan97], o *Fact Partition* (ou *Fact Constelation*) ou o *Dimension Partition*. Para o Snowflake existe o *Snowflake Attribute*.

Existem também outras abordagens como a relatada por [IFT01], que descreve a utilização de visões materializadas, que integram os dados a partir de múltiplas fontes heterogêneas, e eventualmente distribuídas, para a construção de um data warehouse.

3) Séries Temporais

Existem muitas definições sobre séries temporais, muitas delas restritas apenas a um domínio específico.

Definição 1

A definição mais genérica encontrada para séries temporais caracteriza-a como um conjunto de valores de uma variável anotados sequencialmente com o passar do tempo.

Definição 2

De acordo com [Wik06], uma série temporal é uma seqüência de pontos, medidos tipicamente em instantes sucessivos, espaçados em intervalos uniformes.

A análise de séries temporais compreende métodos que tentam identificar tais séries, tanto para entender sua composição e estrutura, bem como para fazer predições.

Toda informação cadastrada em um data warehouse é uma variante no tempo. Sejam essas informações de vendas de clientes, flutuações na bolsa de valores ou batimentos cardíacos de um paciente. O que diferencia este tipo de informação no tempo das demais é que estes dados são correlacionados, ou seja, existe uma relação entre passado, presente e futuro que pode ser explorara por aplicações de datamining[*].

Nesta situação em particular, utiliza-se uma área específica de pesquisa denominada datamining de séries temporais, suas principais atribuições são a descoberta de informações valiosas em séries temporais.

Quando encontramos ciclos de repetições, ou seja, informações com comportamento muito semelhante em momentos distintos no tempo, então podemos definir com que periodicidade tal repetição ocorre. A identificação destes períodos de sazonalidade caracteriza uma das áreas de pesquisa em séries temporais.

[*] *O termo datamining, que pode ser traduzido como mineração de dados, foi mantido como no original em inglês.*

3.1) As técnicas de datamining para análise de séries temporais

3.1.1) Busca por similaridade

A busca por similaridade talvez seja o ramo mais pesquisado no estudo de séries temporais ([FRY94], [HY99]), que se resume a encontrar outras séries temporais similares a uma seqüência de consulta.

Antes de se iniciar a busca por similaridade é necessário ter muito bem definido:

1. O que é similar?

Envolve a definição de um grau de similaridade ou uma margem de erro aceitável quando se compara séries temporais. Normalmente métodos ou modelos nos ajudam a definir quando duas seqüências são similares, mesmo quando em diferentes fatores de escala, em bases diferentes, com ou sem ruído.

2. O que se quer buscar?

Deve-se ter em mãos uma seqüência temporal de consulta, que será a base de toda a pesquisa. Esta série deve representar muito bem o que se deseja buscar. Por exemplo, em um sistema hidroelétrico podemos utilizar um simples amostra que representa um pico de alta tensão, para encontrar todos os momentos em que tais picos ocorreram.

3. Qual o algoritmo de processamento?

O algoritmo será responsável por percorrer a base de pesquisa procurando por similaridades ou anomalias conhecidas. Estes utilizam várias técnicas durante o processamento, como: redução de dimensionalidade, limpeza e remoção de ruídos, transformação e análise de *wavelets*, entre outros.

4. Qual a forma de apresentação dos resultados?

A escolha da visualização deve ser adequada ao que se deseja analisar, suas diferentes formas serão apresentadas mais à frente.

A tarefa de busca por similaridade pode ser dividida em duas categorias:

- Busca por seqüência completa (*whole matching*): representa a busca de uma série temporal em um grande de banco de dados de outras séries temporais individuais, para se encontrar as séries similares.
- Busca por seqüência parcial: (*subsequence matching*): representa a busca de uma pequena subseqüência de uma série temporal em grandes seqüências, na procura do melhor local de similaridade, ou de todos os pontos de similaridade [FRY94].

Embora existam muitos trabalhos nesta área [KK02], na prática esta aplicação se limita aos casos onde algo se sabe *a priori*, sobre o banco que se deseja buscar, normalmente representado pela seqüência de busca.

A busca por subseqüências pode ser facilitada pela divisão da seqüência completa em seções não sobrepostas. Um exemplo seria um eletrocardiograma, cada batimento é representado por uma seção. Este procedimento foi definido por [LKL04] como *chunking*: processo de dividir uma série temporal em pequenas séries individuais, caracterizadas pelo seu período ou forma.

3.1.2) Detecção de Anomalias

Ao contrário da busca por similaridade, a detecção de anomalias baseia-se na identificação de padrões desconhecidos. Esta é uma tarefa muito difícil, pois o que caracteriza uma anomalia pode variar muito dependendo da aplicação.

Uma anomalia é tudo aquilo que não se encaixa no padrão de normalidade, ou segundo [KLC02], uma anomalia é tudo aquilo cuja freqüência dos acontecimentos varia substancialmente daquela esperada, baseada em dados pré-analisados.

Este método pode apresentar uma quantidade muito grande de resultados possíveis, tornando a escolha do método de visualização um dos maiores empecilhos no entendimento dos resultados.

3.1.3) Descoberta dos motivos

Muitas pesquisas baseiam-se na descoberta de padrões super-representados (ou sub-representados) [Lin02], que incluem detecção de intrusos, busca por fraudes, analises ou predições das informações.

A principal idéia está na tentativa de justificar o comportamento de uma série temporal, identificando sua freqüência, sua intensidade e os acontecimentos que levaram àquela expressão.

3.2) Visualizando séries temporais

Conforme apresentado em [LKL04], o olho humano é considerado a mais poderosa ferramenta de datamining, porém relativamente pouco trabalho foi desenvolvido no que se refere à visualização de grandes volumes de dados temporais. Abaixo algumas das soluções mais comuns encontradas na literatura.

3.2.1) TimeSearcher

TimeSearcher [HS01] é uma ferramenta exploratória, que permite a busca de subseqüências através da criação de “timeboxes”, que são localizadores que especificam as regiões onde o usuário está interessado. Versões posteriores [HS04] já incluíram a busca em intervalos variáveis, assim como buscas angulares, que encontram diferenças entre intervalos, e não em seqüências absolutas.

Tal solução ganhou notoriedade pela sua flexibilidade, uma vez que se pode especificar quais partes da série temporal devem ser analisadas, ao invés de analisar toda a seqüência.

Porém apesar de sua facilidade de uso, o TimeSearcher pressupõe algum conhecimento sobre os dados a serem analisados, pois é necessário caracterizar bem as regiões a serem procuradas, além do conteúdo original, ou seja, o padrão a ser procurado.

Apesar de atender a um significativo volume de dados, o TimeSearcher não consegue analisar uma quantidade massiva de dados.

No exemplo da figura 24, observamos o comportamento de duas séries temporais.

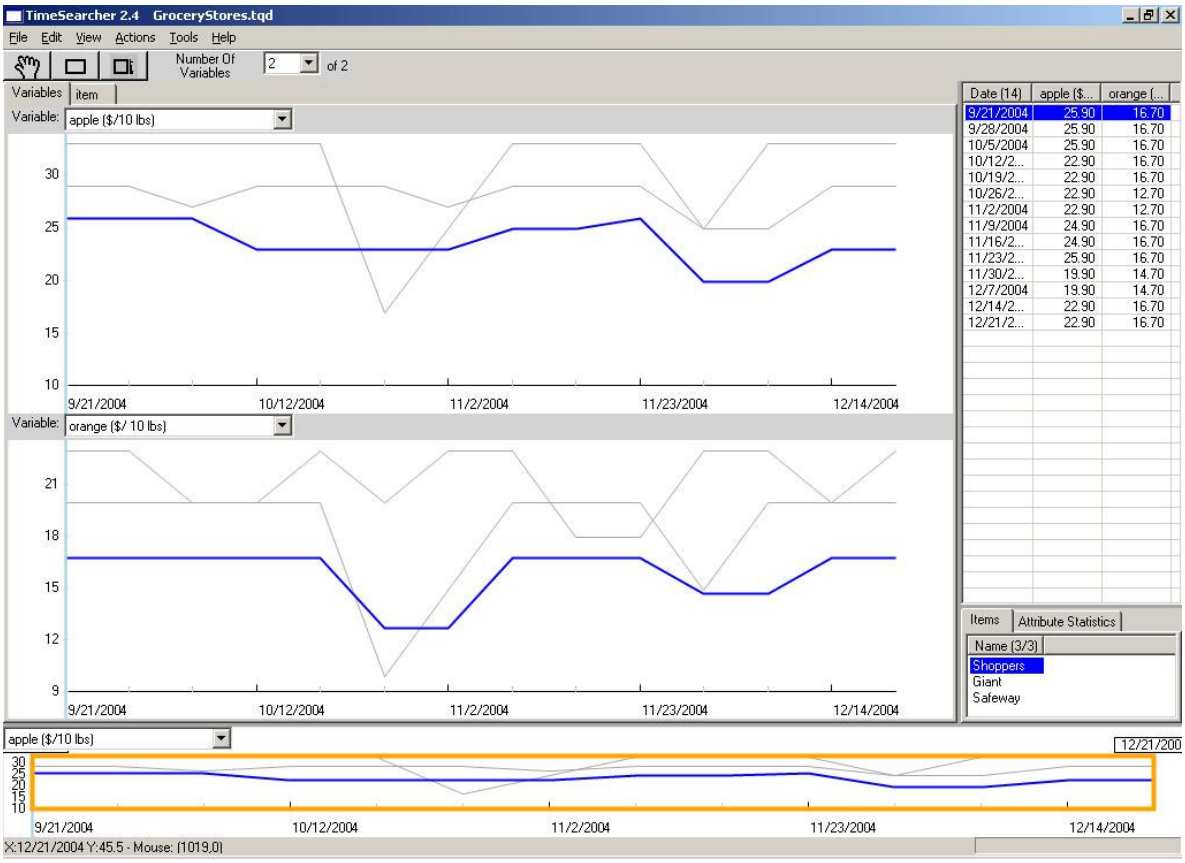


Figura 24 - Interface do TimeSearcher, um usuário pode descartar regiões que não são interessantes para a análise

3.2.2) Clustering e visualização do tipo calendário

A visualização do tipo clustering ou calendário, proposto por [WS99], é obtida pela divisão da série temporal em períodos fixos, como dias, que são agrupados utilizando-se um dos algoritmos de clusterização do tipo “bottom-up”.

Sua visualização apresenta os padrões como dias em um calendário, no qual sua coloração denota o cluster ao qual aquele dia pertence.

Esta abordagem demonstra muito claramente os padrões existentes entre os diversos períodos analisados. Porém, para sua devida utilização, tais sazonalidades precisam estar presentes e serem muito bem definidas, não sendo de grande utilidade em dados que não contenham um padrão de dia/semana/mês/ano bem definido.

O exemplo da figura 25, retirada de [LKL04], demonstra as horas trabalhadas por empregados. Estas horas trabalhadas foram classificadas em seis clusters diferentes, representados cada um por uma cor distinta.

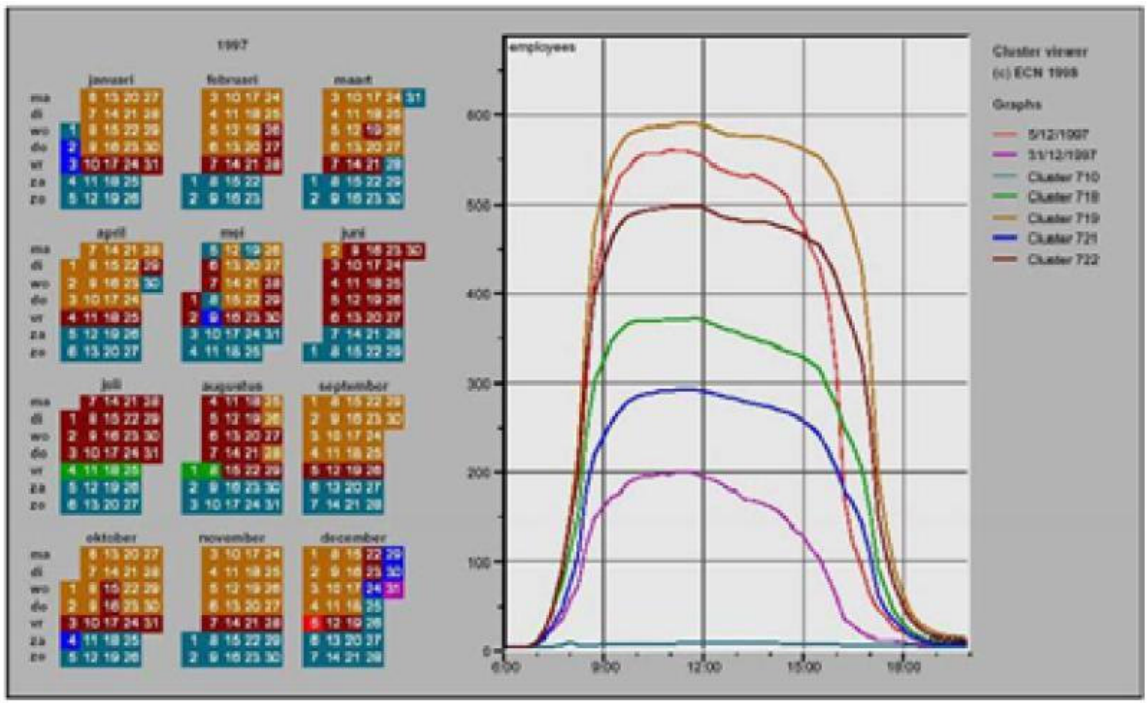


Figura 25 – A interface de visualização do Cluster e baseada em calendário.

3.2.3) Espiral

A visualização em espiral foi proposta por [WAM01]. Nela cada seção periódica da série temporal é mapeada em um anel. Atributos como cor e espessura são utilizadas para caracterizar os dados.

A principal função desta abordagem é a identificação de estruturas periódicas entre os dados. Porém, sua aplicação se limita a séries em que exista um comportamento periódico. O tamanho da série influencia diretamente no número de anéis desta estrutura, estando ela portando limitada à capacidade da memória e resolução do monitor.

Na Figura da figura 26, retirada de [LKL04], pode-se ver o exemplo apresentado por [WAM01], que demonstra a operação de uma usina hidroelétrica. É de fácil visualização que os dias de mais utilização são os mesmos horários dos dias internos da semana, pois não há consumo significativo nos finais de semana.

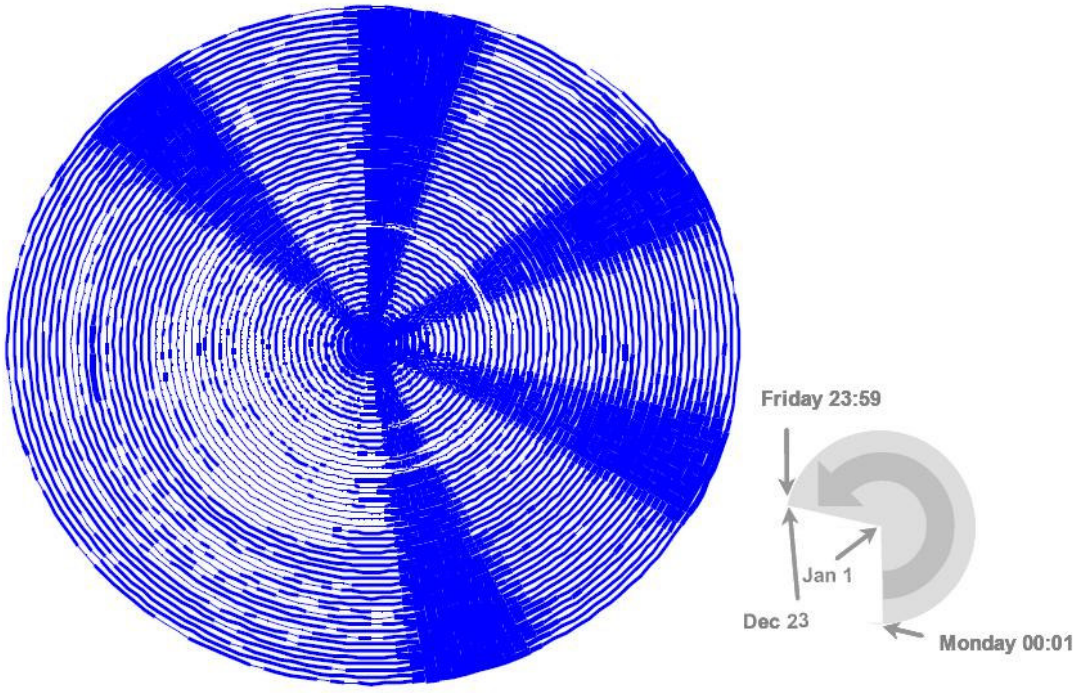


Figura 26 – A interface de modelo em espiral.

3.2.4) VizTree

Esta ferramenta foi apresentada por [LKL04]. Ela baseia-se na transformação da informação armazenada na série temporal em uma árvore de sufixos, tal abordagem oferece facilmente um resumo visual da série, assim como revela padrões inerentes aos dados.

Baseada nas transformações propostas em [Lin03] denominada SAX, e inspirados pelo VisualSys [Kim00], o VizTree se utiliza de quatro índices que precisam ser definidos pelo usuário:

1. A série temporal a ser analisada (C)
2. O tamanho desta série que será transformada em String (n).
3. O número de segmentos de igual tamanho em que a série será dividida (w), cada segmento será depois substituído por um coeficiente.
4. O alfabeto de *strings* {a,b,c,...} que ajudarão na redução da série temporal original.

A figura 27 demonstra como uma dada série pode ser tornar discreta e transformada em uma string.

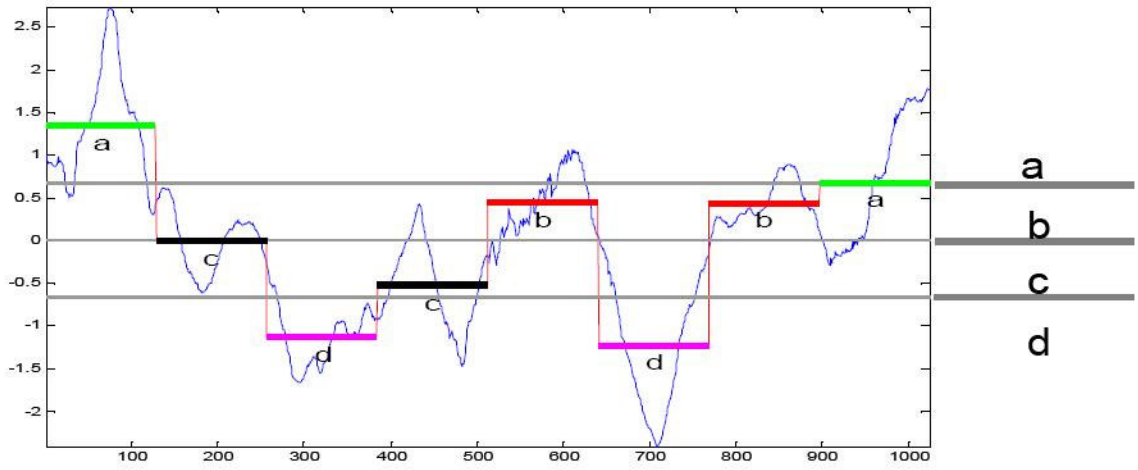


Figura 27 - Série temporal convertida em uma string de oito caracteres "acdcbdba"

Uma vez simplificada e transformada em um alfabeto, a árvore de sufixos pode ser construída. Para tal, cada nó representa uma string do alfabeto, e o caminho da raiz até as folhas representa a subseqüência de strings encontrada. As linhas finas e escuras indicam caminhos ou subseqüências não encontradas, linhas mais grossas vermelhas indicam o padrão seguido pela maioria das informações, linhas finas e vermelhas podem indicar anomalias, ou exceções. Abaixo vemos um exemplo de uma anomalia nos batimentos cardíacos obtidos por um eletrocardiograma sendo detectada.

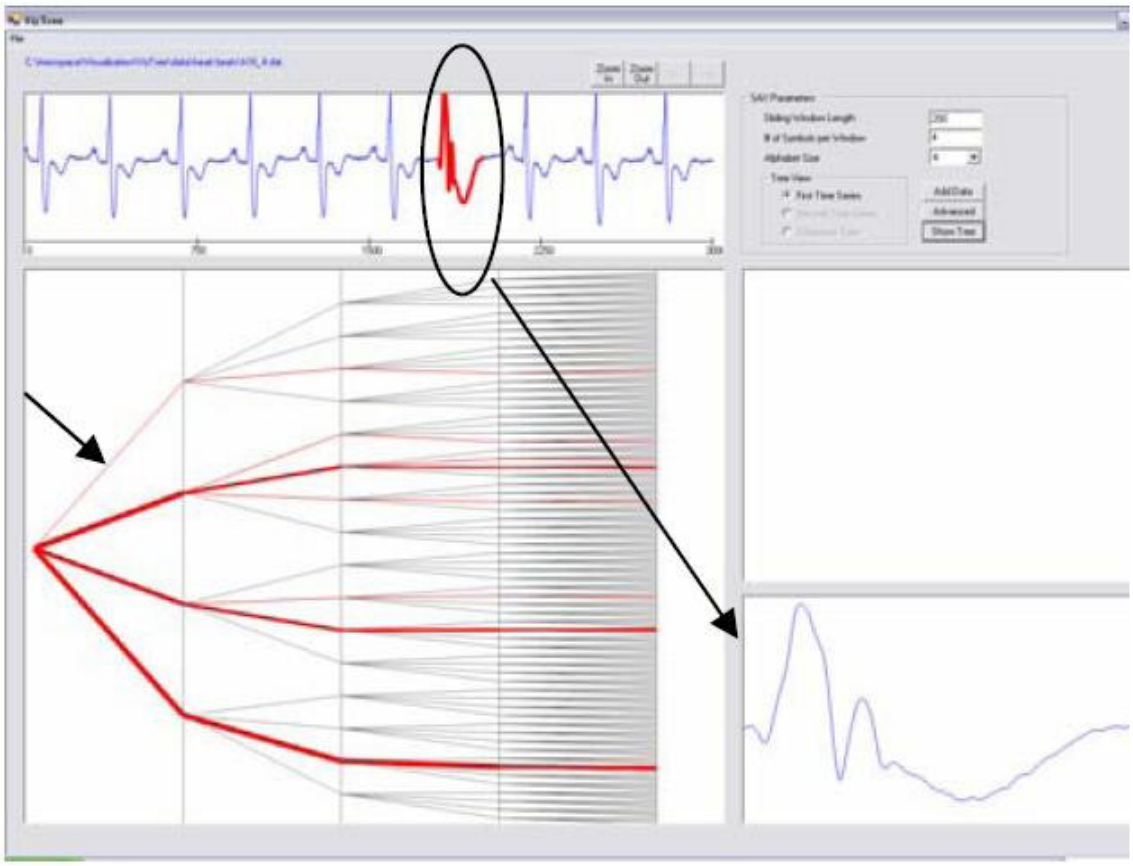


Figura 28 - VizTree, detecção de uma anomalia nos batimentos cardíacos.

Comparação entre árvores (DifTree)

Pela comparação entre duas árvores, o VizTree é capaz de demonstrar as principais diferenças entre elas, apresentando em tons de azul as subseqüências presentes em uma das árvores, e em verde as na outra.

Esta técnica é muito utilizada para detecção de anomalias, quando uma das árvores é representada pelo que se conhece como comportamento “normal” da série. Desta forma, quaisquer variações são facilmente identificadas.

A figura 29 apresenta um exemplo da identificação de um GAP nas informações de duas séries temporais.

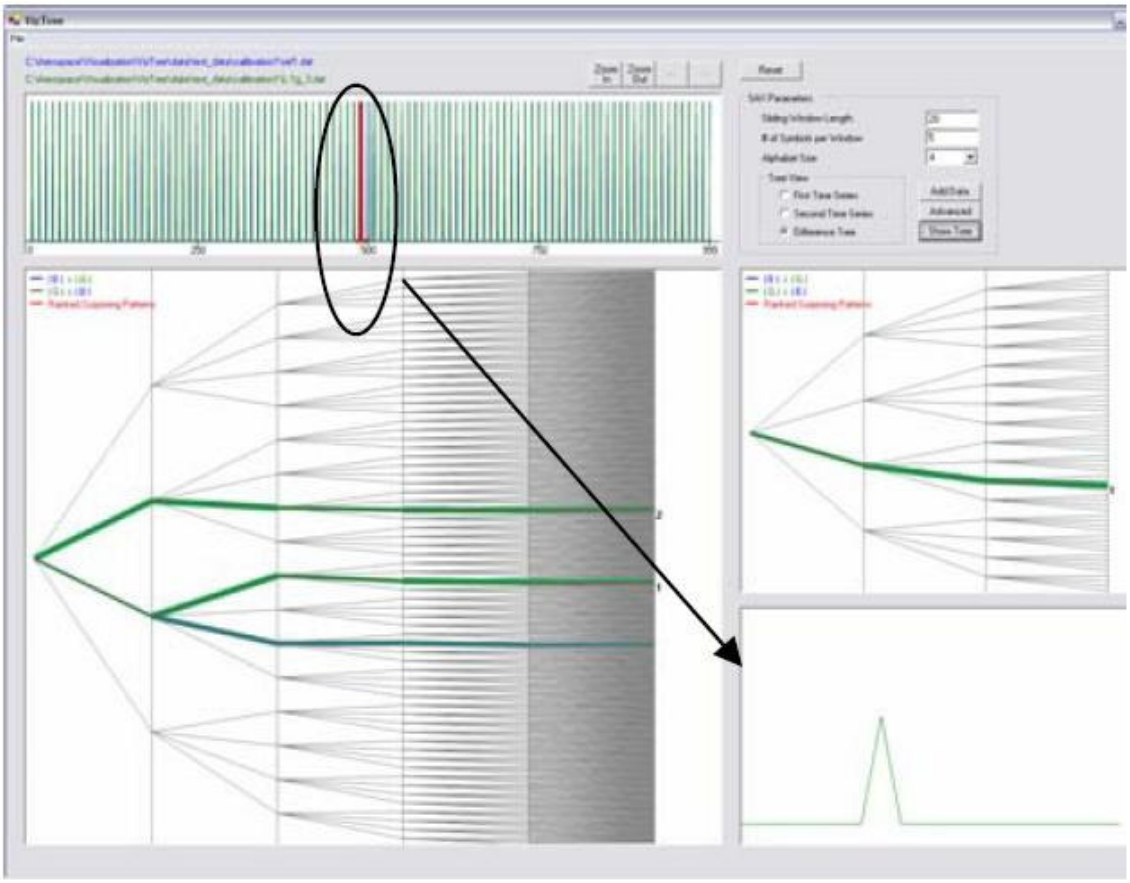


Figura 29 - DifTree, detecção de anomalias obtida pela comparação entre duas séries temporais.

4) Projeto

4.1) O padrão XMLA

O tipo de requisição utilizada no projeto foi o *Extended Makeup Language for Analysis* ou simplesmente XMLA. É um protocolo baseado em SOAP (do inglês *Simple Object Access Protocol)* criado para padronizar o acesso e utilização dos dados entre uma aplicação e um servidor analítico.

A maioria das técnicas de conexão como OLE DB e JDBC, necessita de um cliente de conexão específico para cada servidor instalado na máquina, isto torna os aplicativos desenvolvidos com estas técnicas também específicos para o tipo de plataforma ou servidor suportado pelo cliente.

Desenvolvido em 2002 pela Microsoft Corporation e Hyperion Solutions Corporation, o XMLA se baseou no formato XML (*Extended Makeup Language*) de transferência de dados pela internet, para transmitir também requisições aos servidores que interpretam a solicitação, executam-na e devolvem os dados utilizando o mesmo formato.

As requisições podem ser feitas através de dois métodos: *discover*, utilizado para se questionar o servidor sobre seus meta-dados, e *execute*, onde se passa uma instrução MDX (*Multi Dimensional Expression*) que será executada no servidor analítico.

O diagrama a seguir, figura 30, extraído de [MSC02], ilustra a utilização deste padrão.

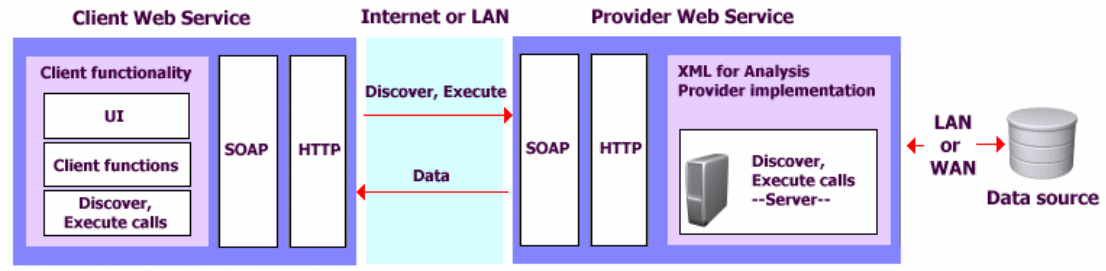


Figura 30 - Estrutura de uma possível implementação de uma aplicação cliente.

4.1.1) O método *discover*

Por definição, deve-se passar três parâmetros ao se utilizar este método: O tipo de requisição (*requestType*), as restrições (*restrictions*), e as propriedades (*properties*).

O tipo de requisição determina o tipo de informação que se deseja buscar, as principais são:

- *DISCOVER_DATASOURCES*: retorna uma lista de fontes de dados disponíveis.
- *DBSCHEMA_CATALOGS*: retorna uma lista de catálogos (bancos de dados analíticos) disponíveis em uma fonte de dados.
- *MDSCHEMA_CUBES*: retorna uma lista de cubos disponíveis em um catálogo.
- *MDSCHEMA_DIMENSIONS*: retorna uma lista de dimensões em um cubo.
- *MDSCHEMA_MEASURES*: retorna uma lista de métricas em um cubo.
- *DISCOVER_SCHEMA_ROWSETS*: retorna uma lista de restrições possíveis.
- *DISCOVER_PROPERTIES*: retorna uma lista de propriedades disponíveis.

As restrições são utilizadas para limitar os dados retornados, como por exemplo, ao listar os cubos existentes deve-se especificar a fonte de dados e o catalogo desejados.

As propriedades servem para regular algum aspecto de sua requisição, como formatação dos dados do resultado, *timeout* (tempo de expiração) da conexão, etc.

A estrutura XMLA de uma mensagem SOAP utilizando o método *discover* pode ser vista na figura 31.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:Discover soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="urn:schemas-microsoft-com:xml-analysis">
   <RequestType>
   </RequestType>
   <Restrictions>
      <RestrictionList>
      </RestrictionList>
   </Restrictions>
   <Properties>
```

```
      <PropertyList>
      </PropertyList>
   </Properties>
  </ns1:Discover>
 </soapenv:Body>
</soapenv:Envelope>
```

Figura 31 - mensagem SOAP utilizando o método discover

A resposta ao método *discover* conterá além dos cabeçalhos SOAP obrigatórios, uma *tag* denominada *row* que contem os dados retornados em formato XML, haverá uma *tag* para cada informação retornada, e um conjunto de *tags* para cada tipo de requisição utilizada. Na figura 32 vê-se a resposta à mensagem do tipo MDSCHEMA_CUBES.

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
<m:return xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:EX="urn:schemas-microsoft-com:xml-analysis:exception">
        <row>
                <CATALOG_NAME> </CATALOG_NAME>
                <CUBE_NAME> </CUBE_NAME>
                <MEASURE_NAME> </MEASURE_NAME>
                <MEASURE_UNIQUE_NAME </MEASURE_UNIQUE_NAME>
                <MEASURE_CAPTION>Tempo  </MEASURE_CAPTION>
                <MEASURE_AGGREGATOR> </MEASURE_AGGREGATOR>
                <DATA_TYPE> </DATA_TYPE>
                <NUMERIC_PRECISION> </NUMERIC_PRECISION>
                <NUMERIC_SCALE> </NUMERIC_SCALE>
                <EXPRESSION </EXPRESSION>
                <MEASURE_IS_VISIBLE> </MEASURE_IS_VISIBLE>
                <MEASURE_NAME_SQL_COLUMN_NAME>
</MEASURE_NAME_SQL_COLUMN_NAME>
        </row>
        ...
</root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Figura 32 - Resposta a uma mensagem SOAP utilizando o método discover, tipo de requisição MDSCHEMA_CUBES

4.1.2) O método execute

Dois parâmetros são necessários para se executar este método: o comando a ser executado no servidor (*command*), e uma lista de propriedades (*properties*).

A estrutura SOAP de um método *execute* pode ser vista na figura 33.

```
<?xml version="1.0" encoding="UTF-8"?>
 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
   <ns1:Execute soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="urn:schemas-microsoft-com:xml-analysis">
    <Command>
          <Statement> </Statement>
    </Command>
    <Properties>
       <PropertyList>
       </PropertyList>
    </Properties>
   </ns1:Execute>
  </soapenv:Body>
 </soapenv:Envelope>
```

Figura 33 - mensagem SOAP utilizando o método execute

A resposta do servidor a uma requisição do tipo *execute* é normalmente um *resultset*, ou seja, um conjunto de dados com os resultados do comando executado. Ele contem três seções: *OLAPInfo* contem informações sobre a estrutura dos eixos (*AxesInfo*) hierarquias de dimensões (*HierarchyInfo*) e algumas informações adicionais para cada dado (*CellInfo*); *Axes* contém os cabeçalhos para cada eixo utilizado; e *CellData* armazena a informação em si. Um exemplo de uma mensagem de retorno a uma instrução *execute* pode ser visto na figura 34.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<m:ExecuteResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
<m:return xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<root xmlns="urn:schemas-microsoft-com:xml-analysis:mddataset"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:EX="urn:schemas-microsoft-com:xml-analysis:exception">
<OlapInfo>
        <AxesInfo>
                <AxisInfo name="">
                        <HierarchyInfo name="">
                                <UName name=""/>
                                <Caption name=""/>
                                <LName name=""/>
                                <LNum name=""/>
                                <DisplayInfo name=""/>
                        </HierarchyInfo>
                </AxisInfo>
        </AxesInfo>
        <CellInfo>
                <Value name=""/>
                <FmtValue name=""/>
                <CellOrdinal name=""/>
        </CellInfo>
</OlapInfo>
<Axes>
        <Axis name="Axis0">
                <Tuples>
                        <Tuple>
                                <Member Hierarchy=" ">
                                        <UName> </UName>
                                        <Caption> </Caption>
                                        <LName> </LName>
                                        <LNum> </LNum>
                                        <DisplayInfo> </DisplayInfo>
                                </Member>
                        </Tuple>
                </Tuples>
        </Axis>
</Axes>
<CellData>
        <Cell CellOrdinal="0">
                <Value xsi:type="xsd:int"> </Value>
                <FmtValue> </FmtValue>
        </Cell>
</CellData>
</root>
</m:return>
</m:ExecuteResponse>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Figura 34 - mensagem SOAP utilizando o método execute

4.2) Implementação

O projeto foi denominado TSA, Time Series Analyser, e foi totalmente desenvolvido em Java, a estratégia adotada foi a introdução de uma camada intermediária entre os sistemas multidimensionais e de visualização de séries temporais. Conforme ilustrado na figura 35 a ferramenta de integração selecionará os dados desejados do servidor OLAP, estes serão convertidos num formato intermediário, que generalizará os padrões de arquivos textos de entrada dos vários sistemas de análise temporal.

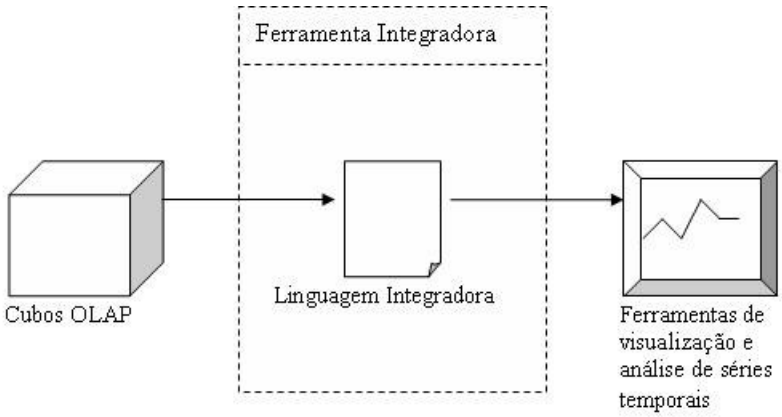


Figura 35 - Arquitetura da ferramenta integradora, uma camada intermediária de conversão entre os ambientes.

O projeto foi desenvolvido em quatro partes: conexão com os bancos de dados multidimensionais, armazenamento e organização dos meta-dados, execução de consultas e a interface de apresentação do projeto.

Toda a parte de conexão com o banco de dados multidimensionais foi desenvolvida utilizando-se a classe *XmlaConnection*. Uma conexão XMLA possui apenas dois atributos obrigatórios: o caminho de um servidor multidimensional e o tipo de requisição (*requestType*). Porém dependendo do tipo de requisição se tornam necessários uma lista de restrições (*restrictionList*), uma lista de propriedades (*propertiesList*) e em alguns casos uma *query* MDX para ser executada no servidor (*statement*).

Os métodos *response* e *responseAsStream* são responsáveis por compor a requisição SOAP, enviá-la ao servidor e coletar sua resposta. O envio e recebimento de mensagens é realizado por dois métodos: *writeToStream*, que envia uma mensagem ao servidor especificado, e *readFromStream* que coleta a resposta do servidor.

O armazenamento e organização dos meta-dados é um conjunto de classes responsáveis por questionar o servidor sobre sua estrutura e desta forma construir uma hierarquia organizada dos dados, por exemplo, um objeto do tipo *Server* pode conter vários objetos *DataSource*, conforme ilustrado na figura 36.

Cada classe individualmente, exceto *Measures* (medidas) e *Dimensions* (dimensões), é responsável por compor a mensagem SOAP do tipo *discover*, enviá-la ao servidor multidimensional, receber a resposta e desta extrair as informações necessárias para se construir os objetos agregados.

A parte de execução de consultas, realizada pela classe *Query* (consulta), é responsável por questionar o servidor analítico, utilizando uma instrução MDX, analisar sua resposta e armazenar seus resultados, denominados *resultset*, em uma estrutura que facilite a conversão para os formatos de saída, no caso, DAT (utilizado pelo *VizTree* [LKL04]), CSV (utilizado pelo *Microsoft Excel*) e TQD (utilizado pelo *TimeSearcher* [HS01]).

A interface é implementada pela classe *MainMenu*, esta apresenta uma interface gráfica, utilizando componentes genéricos das bibliotecas *Awt* e *Swing*, para coletar as opções do usuário sobre os parâmetros da consulta ao servidor.

O diagrama de classes simplificado, apresentado na figura 36, demonstra o relacionamento entre as estruturas. Nota-se que toda classe que precisa se comunicar com o servidor analítico utiliza a classe XmlaConnection, assim como que a classe MainMenu usa todas as classes de meta-dados e de execução de consultas para construir a interface gráfica.

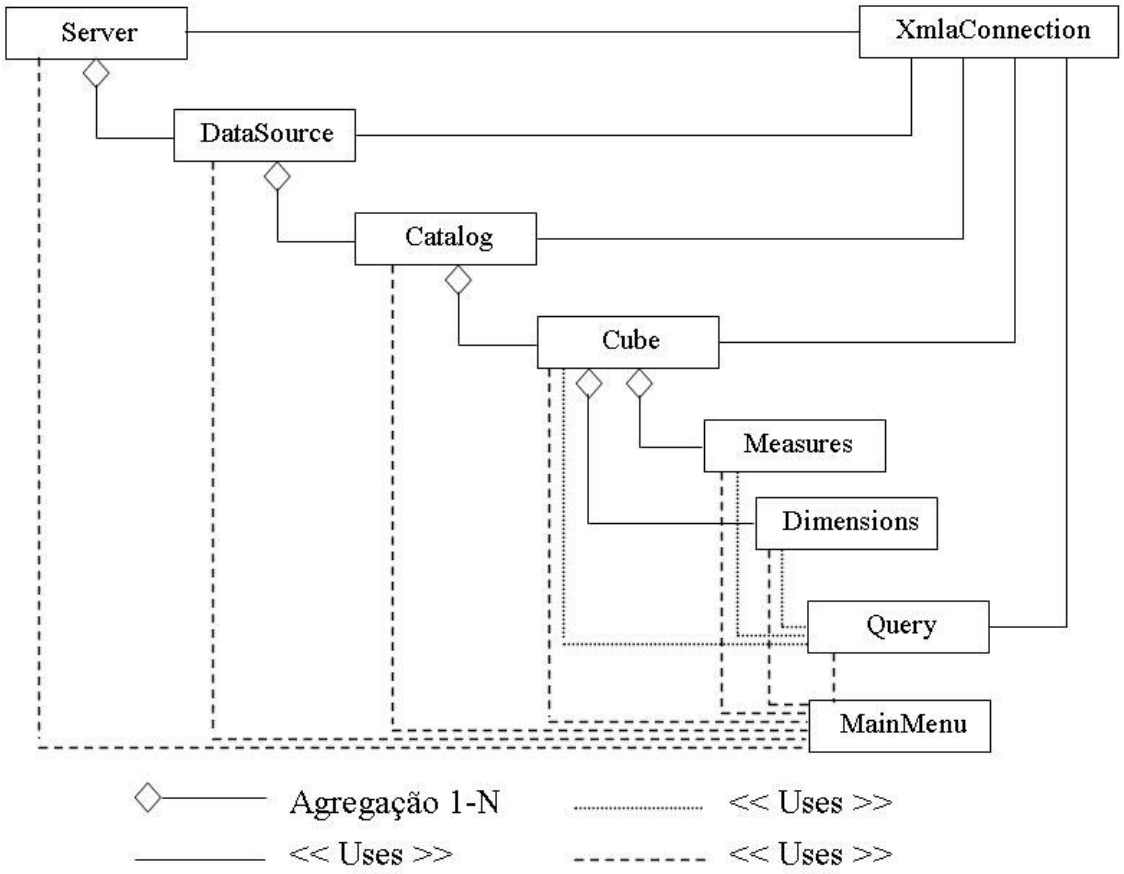


Figura 36 – Diagrama de classes simplificado, com destaque para as relações entre os objetos.

O código fonte desta aplicação pode ser encontrado em http://ime.usp.br/~jef/gbmaia.

O diagrama de seqüência, demonstrando a ordem das mensagens trocadas entre os objetos, pode ser visto na figura 37. Ele representa um cenário sem erros ou tratamento de exceções, partindo da seleção da URL do Servidor, até a visualização de uma série temporal.

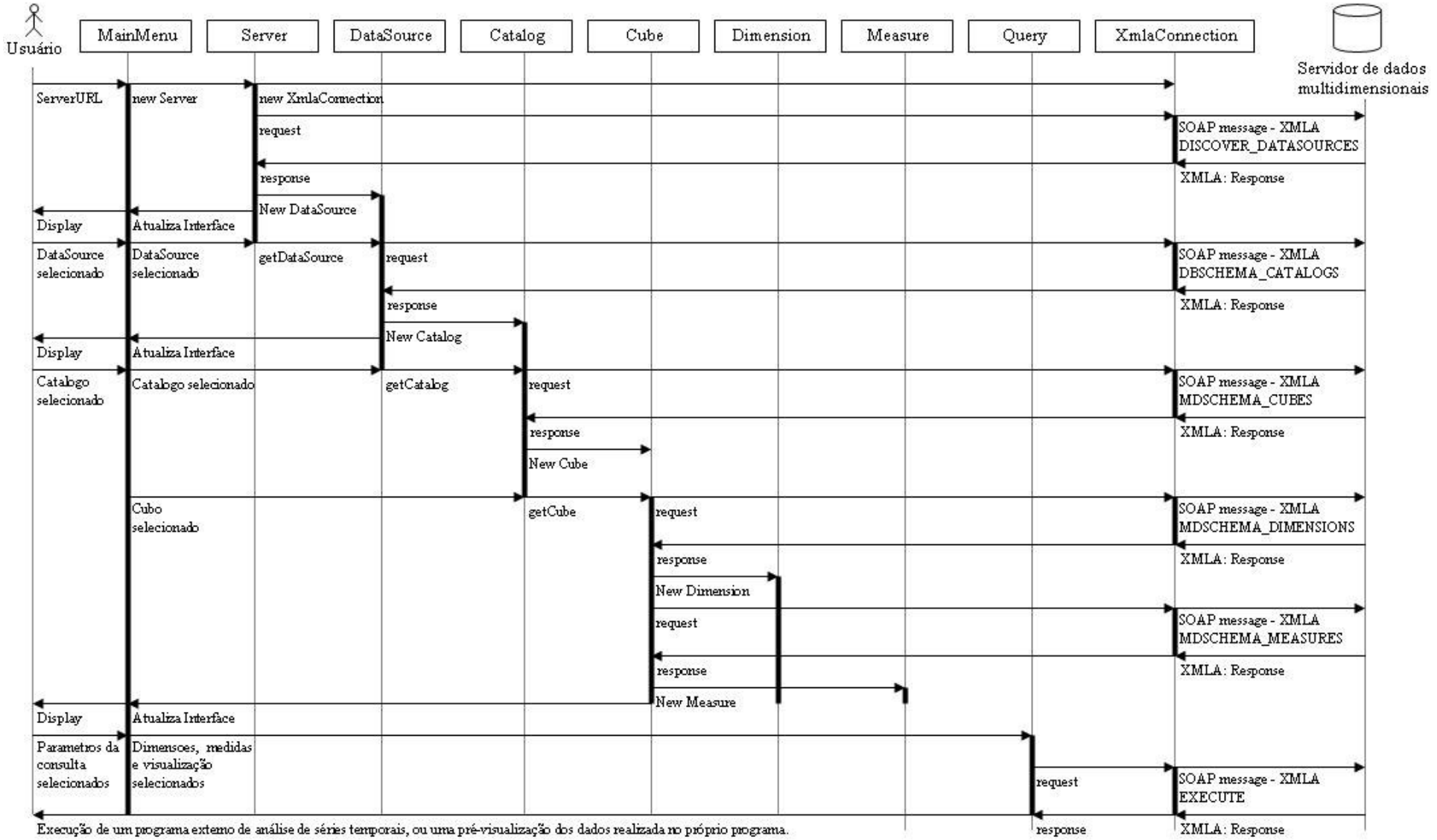


Figura 37 - Diagrama de seqüência

A interface gráfica possui três partes distintas, a primeira, representada na figura 38, apresenta um menu de navegação onde o usuário informa: a URL do Servidor, o datasource, o catálogo, o cubo, uma dimensão de tempo no eixo X, opcionalmente uma dimensão não temporal no eixo Y e uma medida. Uma vez descrito como será a consulta desejada, o usuário terá uma opção de visualização interna utilizando a opção *Preview*, e três externas, sendo estas respectivamente: Microsoft Excel, TimeSearcher (capítulo 3.2.1) e VizTree (capítulo 3.2.4). Estas últimas sendo ferramentas exclusivas de análise de séries temporais.

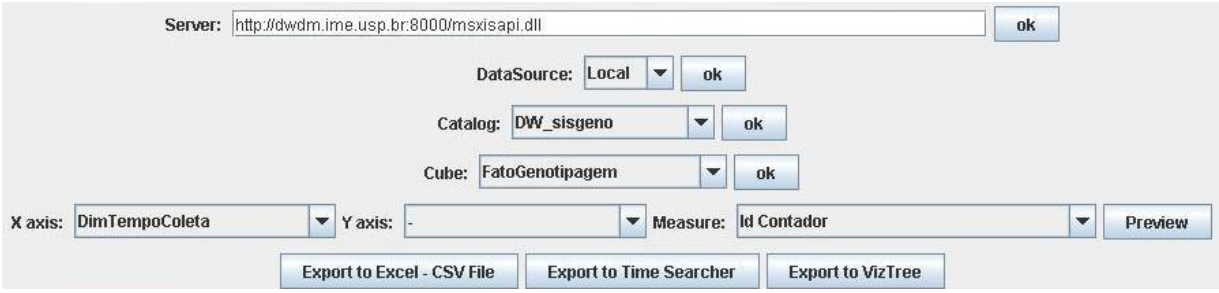


Figura 38 - Interface gráfica, menu de navegação

A navegação deve ser feita seqüencialmente, ou seja, é necessário informar a URL do servidor antes que se possa selecionar um datasource, porém, uma vez informado um valor, pode-se alterar qualquer seleção posterior (diga-se abaixo), sem a necessidade de voltar ao topo para informar os dados novamente, ou seja, uma vez informado o cubo é possível manipular as informações dos eixos e das medidas livremente e quantas vezes se desejar. Caso uma opção superior seja alterada, por exemplo, a URL do servidor, todas as informação dependentes daquele dado são zeradas, ou seja, será preciso informá-las novamente.

Uma nota sobre a exportação para o VizTree [LKL04], devido ao fato deste aplicativo não ter a opção de automaticamente abrir um arquivo passado como parâmetro, a ferramenta se limita a exportar o arquivo .DAT e abrir o programa, cabe ao usuário abrir o arquivo desejado.

A segunda parte da interface, localizada no centro do aplicativo, é uma área reservada para a pré vizualização das séries temporais, conforme exemplo da figura 39. Apesar de não oferecer os recursos avançados de navegação de um ambiente multidimensional como *Rollup*, *Drill-Down* e *Slice and Dice* esta opção permite ao usuário rapidamente validar parte dos dados da série temporal, ou mesmo decidir-se sobre qual medida ou dimensão utilizar em sua consulta, isso sem precisar recorrer às ferramentas externas.

[DimTempoPare...	[DimTempoPare...	[DimTempoPare...	[DimTempoPare...
39378	12	36	96
11739	0	12	36
0	0	0	0
27639	12	24	60
0	0	0	0

Figura 39 - Interface gráfica, área de pré visualização

A terceira parte, encontrada na parte inferior da interface, armazena um log de mensagens SOAP enviadas e recebidas pela aplicação, demonstrando não apenas os casos de sucesso, mas também os erros de comunicação. Ou seja, pode-se recorrer a esta sessão para procurar por erros nas mensagens caso o sistema não consiga realizar a ação solicitada. Na figura 40 está apresentado um exemplo desta área de log.

```
Log Start:
request
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
 <soapenv:Body>
```

Figura 40 - Interface gráfica, área log de mensagens

Uma apresentação desta ferramenta, contendo os três programas, pode ser visualizado na figura 41, neste exemplo o mesmo dado foi exportado para o Microsoft Excel (à esquerda), o TimeSearcher (central) e VizTree (à Direita).

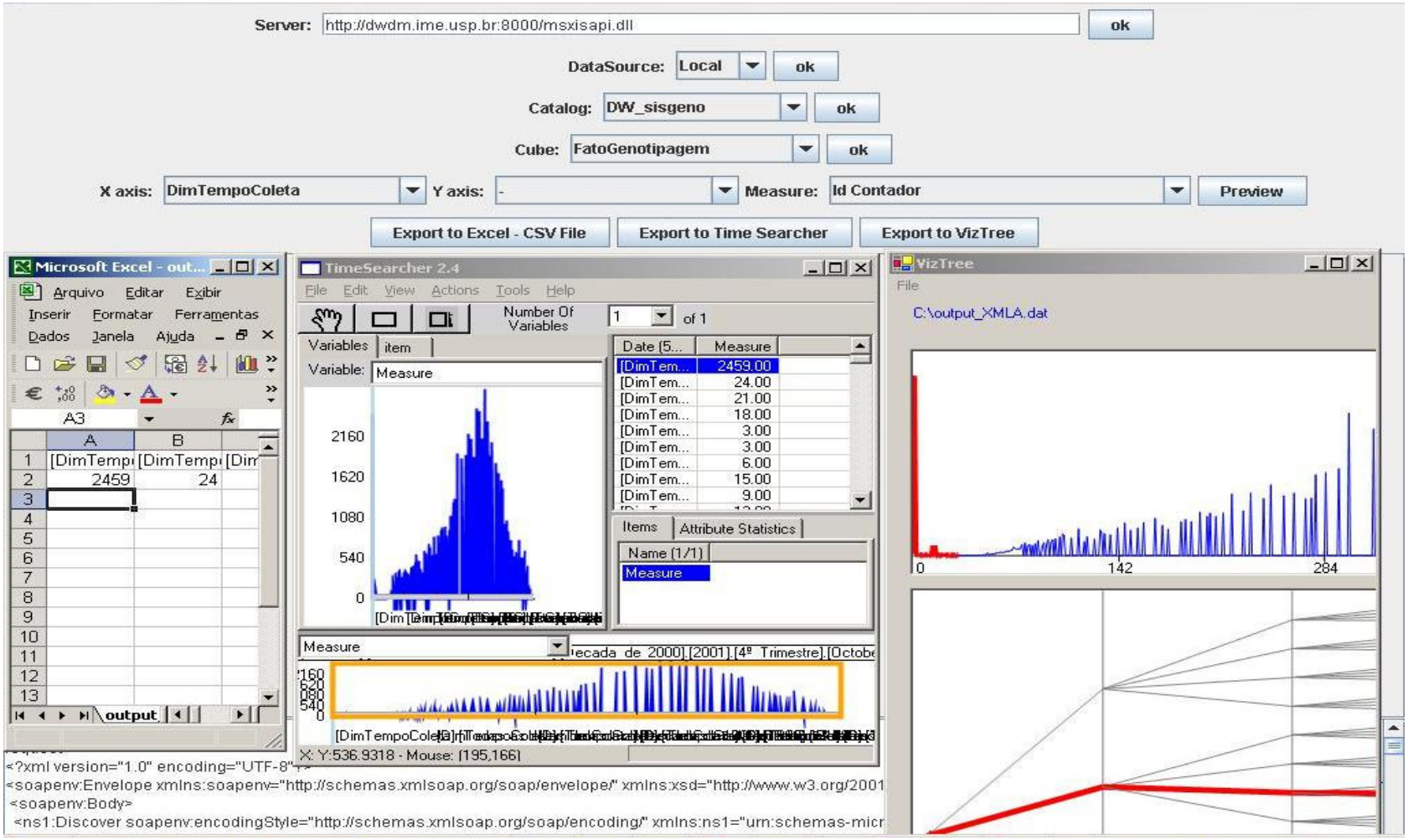


Figura 41 - Interface gráfica com as três opções de exportação abertas

5) Estudo de Caso

Em agosto de 2004 foi elaborado um contrato, Projeto 914/BRA/1101-UNESCO, para a construção de um data warehouse a partir da extração dos dados de três projetos já existentes do ministério da Saúde, sendo eles o Sisgeno, Siclom e Siscel. Tal projeto teve a gerência do PNDST/AIDS (Programa Nacional de Doenças Sexualmente Transmissíveis / AIDS), e foi desenvolvido por alunos de mestrado e doutorado do IME/USP sob a coordenação do Prof. Dr. João Eduardo Ferreira e Dra. Ester Sabino.

A base foi entregue em *Sybase*, posteriormente extraída para o *Microsoft SQL Server*, onde foi feito todo o processo de limpeza e consolidação dos dados, além da construção dos respectivos sistemas analíticos utilizando o *Microsoft Analysis Manager.*

O Sisgeno em específico, foi mapeado em um catálogo denominado DW_Sisgeno. Ele possuía dois cubos: FatoGenotipagem e FatoEsquemaMedicamento, porém apenas o primeiro, contendo dados sobre exames realizados por pacientes do PNDST/AIDS, será citado neste trabalho.

Este cubo possui diversas dimensões e medidas, como visto na figura 42, em imagem extraída da ferramenta analítica utilizada para a construção do cubo.

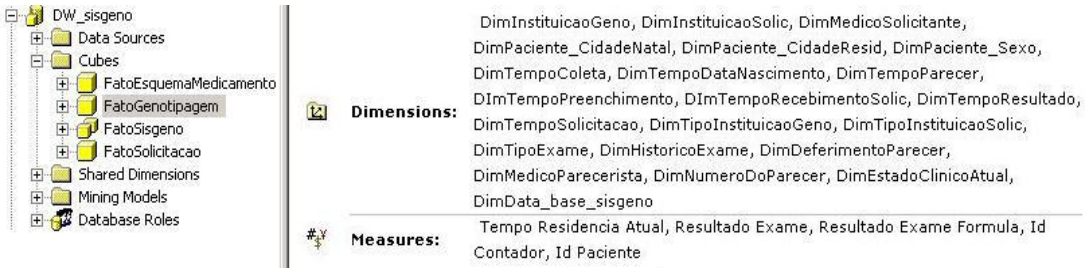


Figura 42 – Dimensões e medidas do cubo FatoGenotipagem, projeto Sisgeno

A proposta é analisar a série temporal formada pelo número de amostras coletadas no tempo. A dimensão de tempo escolhida será DimTempoColeta, que representa as datas em que materiais foram colhidos para os exames laboratoriais do Sisgeno, e a medida será NrAmostras, representando o número de amostras colhidas.

Os parâmetros utilizados para obter esta análise foram:

Server → http://dwdm.ime.usp.br:8000/msxisapi.dll,
DataSource → Local;
Catalog → DW_Sisgeno;
Cube → FatoGenotipagem;
X Axis → DimTempoColeta;
Y Axis → " – " (Vazio);
Measure → NrAmostras

A primeira análise foi a procura por padrões de repetições ou busca por similaridade, para isso a ferramenta utilizada foi o VizTree [LKL04]. Esta série temporal não se provou muito boa para este tipo de análise.

Porém, ajustando-se os parâmetros da ferramenta de visualização temporal, pode-se demonstrar um caso em que duas seqüências se assemelham muito, veja a figura 43, demonstrando um pico menor imediatamente seguido de um pico maior.

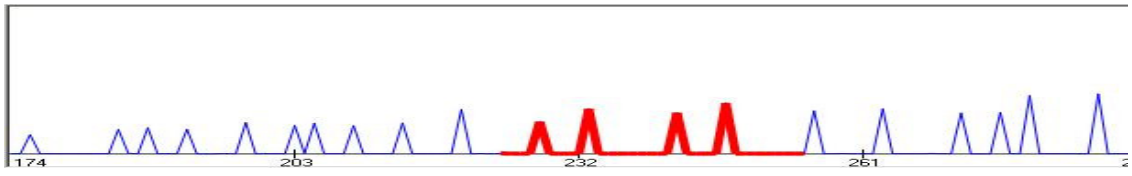


Figura 43 – Identificação de um padrão de repetição no VizTree [LKL04]

O número de ocorrências de cada padrão detectado pode ser observado no mapeamento em forma de árvore montado pela ferramenta, conforme figura 44, o caso acima é uma das linhas escuras de menor intensidade. As linhas de maior intensidade representam a grande maioria da seqüência, onde praticamente não houve padrões de repetição.

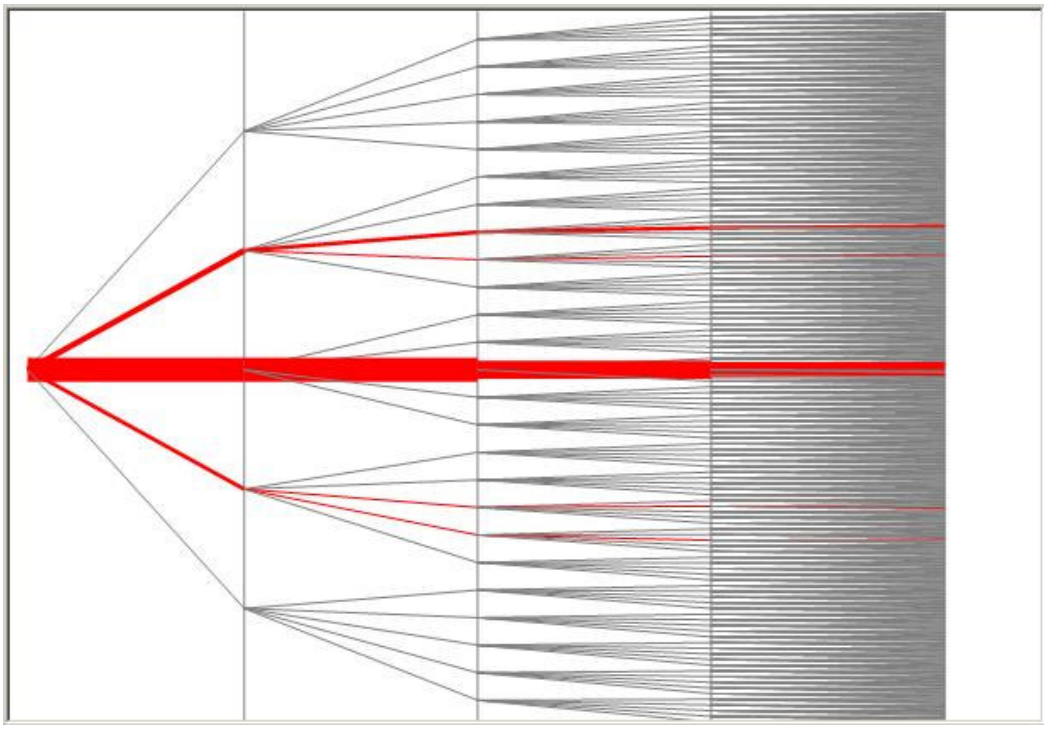


Figura 44 – mapeamento em forma de árvore no VizTree [LKL04]

Em um segundo exemplo de análise, será realizado a busca por padrões visuais globais, na tentativa de se visualizar a seqüência toda de uma só vez, porém, para aumentar a riqueza da análise, uma segunda dimensão foi acrescentada. Os parâmetros ficaram da seguinte forma:

Server → http://dwdm.ime.usp.br:8000/msxisapi.dll,
DataSource → Local;
Catalog → DW_Sisgeno;
Cube → FatoGenotipagem;
X Axis → DimTempoColeta;
Y Axis → DimPaciente_Sexo;
Measure → NrAmostras

A ferramenta de visualização será o TimeSearcher [HS01], e como demonstrado na figura 45, existe a formação de um padrão similar ao de uma função quadrática, demonstrando assim o período de maior coleta do programa Sisgeno.

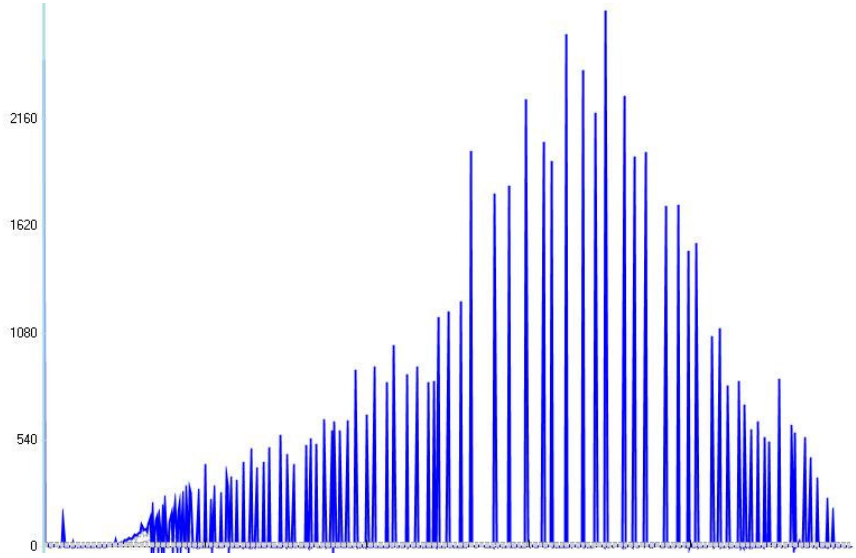


Figura 45 – Análise de uma série temporal com o TimeSearcher [HS01].

Quando analisado o número de amostras por sexo do paciente pelo tempo, sendo este a data de emissão das amostras, observa-se o comportamento similar das três séries temporais, porém é notável que a quantidade de homens é bem superior à das mulheres em todos os períodos da amostragem, conforme visto na figura 45.

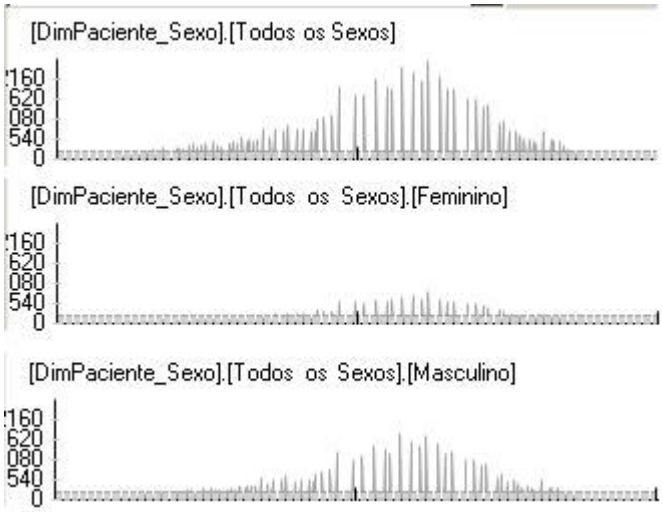


Figura 46 – Visualização de três séries temporais, identificadas pele gênero do paciente.

5.1) Configurações necessárias

Como este projeto se trata de uma ferramenta integradora, ela necessita que os servidores analíticos estejam devidamente configurados, assim como as ferramentas de visualização de séries temporais estejam instaladas na máquina.

5.1.1) Servidores Multidimensionais

Por utilizar-se do padrão XMLA, qualquer servidor que seja capaz de enviar e receber mensagens SOAP, interpretar comandos XMLA e executar consultas em MDX, poderá ser utilizado pela ferramenta proposta. Duas ferramentas foram testadas como provedoras de informação:

- Microsoft SQL Server 2000 Analysis Manager

Esta ferramenta exige a configuração e instalação de alguns aplicativos para que o servidor possa aceitar e executar instruções MDX, estes passos podem ser encontrados em: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/implementxap.asp, e se resumem em: instalar o *XML for Analysis SDK* o pacote de serviços e *parsers* para XML, criar o diretório virtual no IIS (*Internet Information Service*), habilitar as extensões para serviços analíticos (apenas no caso do Windows Server 2003), configurar os provedores de dados (*datasource*) alterando os arquivos xml de configuração do *XML for Analysis SDK*, ajustar os níveis de segurança e finalmente testar a aplicação.

- Mondrian [Sou01]

Esta ferramenta de domínio público é distribuída livremente pela SourceForge. Ela também requer várias configurações para sua instalação encontradas em: http://mondrian.pentaho.org/documentation/installation.php, como: instalação do Java SDK, instalação do Apache/Tomcat, realizar o download e descompactar o arquivo monrian.war no diretório de aplicativos do Tomcat, configurar um banco de dados, configurar o mondrian com servidor XMLA (o que inclui a configuração dos *datasources*).

5.1.2) Ferramentas de análise de séries temporais

Atualmente apenas três formatos de exportação dos dados estão disponíveis, DAT, CSV e TQD, porém, qualquer aplicativo que aceite um destes padrões já pode ser utilizado para análises e devido à forma como foi implementado, este sistema permite a rápida criação de novos formatos de exportação.

Estas ferramentas são:

- Microsoft Excel

Apesar de ser bem limitado quanto a número máximo de colunas (256), linhas (65536), e à quantidade de informação que ele é capaz de disponibilizar na tela, o Excel ainda é muito utilizado devido a sua facilidade de utilização, e se torna uma parte importante na validação da informação exportada para as ferramentas de análise mais especializadas. Ele é parte integrante do Microsoft Office, pacote de produtos da Microsoft.

- TimeSearcher [HS01]

Seu download pode ser feito a partir de: http://www.cs.umd.edu/hcil/timesearcher/, e basta descompactá-lo em um diretório que ele estará pronto para ser utilizado, porém para facilitar a chamada direta pelo integrador, é necessária a criação de um link em C: denominado ts.lnk apontando para o caminho onde o TimeSearcher foi instalado.

- VizTree [LKL04]

Encontrado em http://www.cs.ucr.edu/~jessica/viztree/, o VizTree não requer instalação, basta executá-lo diretamente. Ele também requer a criação de um link denominado viztree.lnk em C: para facilitar sua execução.

6) Conclusões

Este trabalho apresentou uma ferramenta de domínio público para a integração de softwares de visualização de séries temporais e modelos multidimensionais, que facilita não apenas a apresentação dos resultados de maneira mais clara e concisa, mas que também utiliza todo o poder de análise de um ambiente analítico e a riqueza da informação adquirida pela análise temporal.

6.1) Contribuições

Com o auxilio desta ferramenta, usuários de bancos de dados analíticos como o Microsoft SQL Server 2000 Analysis Manager e o Mondrian, assim como quaisquer outras aplicações que sejam capazes de receber requisições XMLA e executar comandos MDX, têm ao seu dispor uma ferramenta de extração de séries temporais integrada a aplicativos de visualização de séries temporais como o TimeSearcher [HS01] e o VizTree [LKL04], ampliando assim o poder de análise de tais bancos.

Da mesma forma, usuários de ferramentas estatísticas ou de análise de séries temporais, em especial os que utilizam arquivos no formato CSV, DAT e TQD, têm à sua disposição uma ferramenta para extração de informações de bancos analíticos, uma fonte de dados já consolidados e sumariados, facilitando assim o processo de aquisição e preparação da informação.

6.2) Limitações do trabalho

Ao longo do desenvolvimento do trabalho, algumas limitações foram encontradas, como:

- Dos quatro modelos de visualização de séries temporais apresentados, apenas dois puderam ser implementados. O modelo clustering e visualização do tipo calendário, apresentado por [WS99] e o modelo em espiral, apresentado por [WAM01], não se tratam de um software, e sim de uma proposta de visualização.
- Embora dois servidores analíticos(Microsoft SQL Server 2000 Analysis Manager e Mondrian) tivessem sido testados para serem utilizados pela ferramenta integradora, outros servidores são conhecidos por utilizar o padrão XMLA como Hyperion, Sas e Microsoft SQL Server 2005, não puderam ser testados pelas dificuldades de aquisição/instalação/configuração dos mesmos.
- O processo de comunicação com o servidor analítico, com a utilização de mensagens SOAP no padrão XMLA, esbarrou em um dos problemas dessa tecnologia, a performance. Quando a quantidade de informação é muito grande, existe a possibilidade de se receber uma mensagem de timeout do servidor, ou que a memória alocada para a aplicação java não suporta a criação da estrutura de objetos de meta-dados para armazenar toda a informação. Estas deficiências podem ser contornadas aumentando o tempo de espera de mensagens, e aumentando a quantidade de memória reservada a aplicação Java para sua execução.

6.3) Futuras pesquisas

Esta ferramenta integradora ainda não está completa, como objetivos futuros incluem-se :

- Registro como uma ferramenta de domínio público.
- Inclusão de novos formatos de exportação, o que engloba novas formas de visualização de séries temporais.
- Um arquivo configurador para facilitar a utilização pelo usuário.
- Implementação das funções de: *Rollup*, *Drill-Down* e *Slice and Dice* quando observadas dentro da aplicação pelo pré visualizador.
- Utilização de transações seguras SSL, para o envio e recebimento das mensagens.
- Adicionar maior compressão de dados, reduzindo o tráfego de dados e possivelmente solucionando os timeouts causados pelo excesso de informação requisitada.

Referências Bibliográficas.

[AV98] C. Adamson, M.Venerable. Data warehouse Design Solutions, John Wiley & Sons, 1998.

[BPT97] E. Baralis, S. Paraboschi, E. Teniente. "Materialized view selection in a multidimensional database". Proceedings of the 23rd VLDB Conference, Atenas, Grécia, 1997.

[FRY94] C. Faloutsos, M. Ranganathan, and Y. Manolopulos. Fast Subsequence Matching in Time-Series Databases. SIGMOD Record 1994; 23(2): 419-429.

[GM96] A. Gupta e I. S. Munick. "What is the data warehousing problem ? (Are materialized views the answer ?)". Proceedings of the 22nd VLDB Conference, Mumbai (Bombay), India, 1996.

[GMR98] M. Golfareli, D. Maio, S. Rizzi. "Conceptual Design of Data warehouses from E/R Schemes". Proceedings of the Hawaii International Conference on System Sciences, Kona, Hawaii, USA, 1998.

[HS01] H. Hochheiser and B. Shneiderman. Interactive Exploration of Time-Series Data. the 4th Int'l Conference on Discovery Science 2001 (Washington D.C.), Springer-Verlag; 441-446

[HS04] H. Hochheiser and B. Shneiderman. Dynamic Query Tools for Time Series Data Sets: Timebox widgets for interactive exploration. Information Visualization 2004; 3: 1-18.

[HY99] Y. W. Huang and P. S. Yu. Adaptive Query Processing for Time-Series Data. the 5th ACM SIGKDD Int'l Conference on Knowledge Discovery and Datamining 1999 (San Diego, CA); 282-286.

[IFT01] I. C. Italiano, J. E. Ferreira, O. K. Takai. "Aspectos Conceituais em Data warehouse", Relatório técnico, Instituto de Matemática e Estatística da Universidade de São Paulo - IME/USP. São Paulo, 2001.

[Kei02] D. A. Keim. Information Visualization and Visual Datamining. IEEE Transactions on Visualization and Computer Graphics 2002; 8(1): 1-8.

[Kel94] S. Kelly. Data Warehousing The Route to Mass Customization, John Wiley & Sons, 1994

[Kim96] R. Kimball. *The data warehouse toolkit*. John Wiley & Sons, 1996.

[Kim00] S. Kim, et al. Visualysis: A Tool for Biological Sequence Analysis. The 4th Int'l Conference on Computational Molecular Biology 2000 (Tokyo, Japan).

[KK02] E. Keogh and S. Kasetty. On the Need for Time Series Datamining Benchmarks: A Survey and Empirical Demonstration. the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Datamining 2002 (Edmonton, Alberta, Canada); 102-111.

[KLC02] E. Keogh, S. Lonardi, and B. Chiu. Finding Surprising Patterns in a Time Series Database in Linear Time and Space. the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Datamining 2002 Edmonton, Alberta, Canada); 550-556.

[Lin02] J. Lin, et al., Finding Motifs in Time Series, in the 2nd Workshop on Temporal Datamining. 2002: Edmonton, Alberta, Canada.

[Lin03] J. Lin, et al., A Symbolic Representation of Time Series, with Implications for Streaming Algorithms, in Workshop on Research Issues in Datamining and Knowledge Discovery. 2003: San Diego, CA.

[LKL04] J. Lin, E. Keogh, S. Lonardi. Visualizing and Discovering Non-Trivial Patterns In Large Time Series Databases. Presented on VLDB 2004.

[MF01] L. A. Mantovani, J. E. Ferreira. *Uma alternativa para simplificação do sincronismo de dados históricos dos Ambientes Operacionais e Analíticos*. Dissertação de mestrado do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, São Carlos, 2001.

[MSC02] Microsoft Corporation, Hyperion Solutions Corporation. “XML for Analysis Specification version 1.1”, disponível em http://www.xmla.org, 2002

[RadD96] N. Raden. "Technology Tutorial: Modeling a data warehouse", disponível em http://techweb.cmp.com/iw/564/64oldat.htm, 1996

[Sei01] C. Seideman. Datamining with SQL Server 2000, Technical Reference. Microsoft Press, 2001.

[Shn96] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. the IEEE Symposium on Visual Languages 1996 (Boulder, CO), IEEE Computer Society Press; 336-343

[SMKK98] S. Mantini, M.Mohania, V. Kumar, Y. Kambayashi, ER Workshops, 1998, pag 81-92

[Sou01] Sourceforge. Mondrian OLAP Server. Common Public License, disponível em http://sourceforge.net/projects/mondrian/

[Tan97] R. Tanler. The intranet data warehouse, John Wiley & Sons, 1997

[Tuf83] E. R. Tufte, The Visual Display of Quantitative Information. Graphics Press: Cheshire, CT, 1983.

[WAM01] M. Weber, M. Alexa and W. Muller. Visualizing Time Series on Spirals. 2001 IEEE Symposium on Information Visualization 2001 (San Diego, CA); 7-14.

[Wik06] Time Series, http://en.wikipedia.org/wiki/Time_series, last modified on 2006 by "C. Matthews". (This article is under the GNU Free Documentation License http://www.gnu.org/copyleft/fdl.html)

[WS99] J.J. van Wijk and E.R. van Selow. Cluster and Calendar Based Visualization of Time Series Data. 1999 IEEE Symposium on Information Visualization 1999 (San Francisco, CA); 4-9.

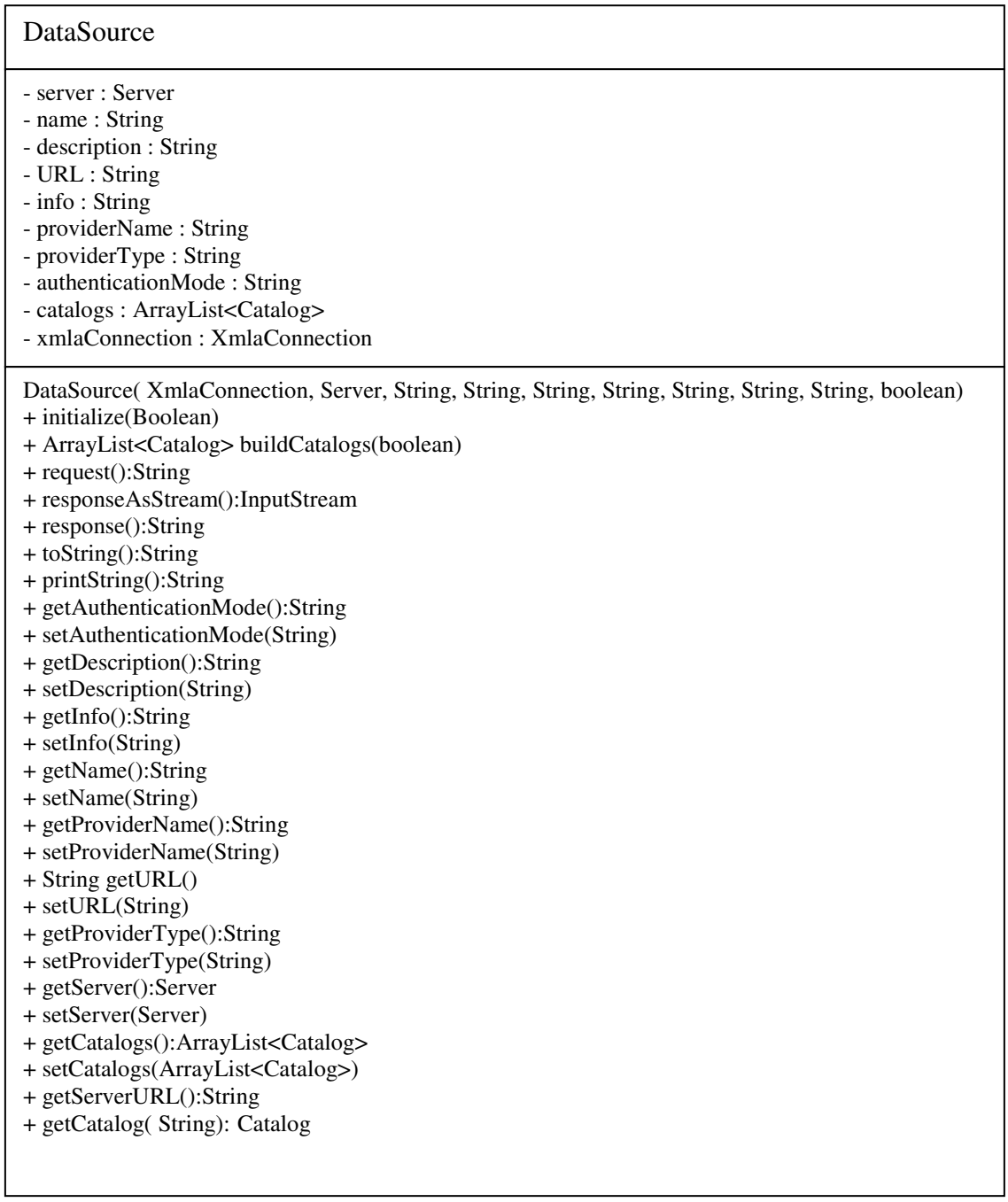
[ZGMHW94] Y. Zhuge, H.Garcia-Molina, J. Hammer e J. Widom. View maintenance in a warehouse environment. Technical report, Stanford University. Disponível: ftp: db.stanford.edu como pub/zhuge/1994/anomaly-full.ps. Outubro de 1994

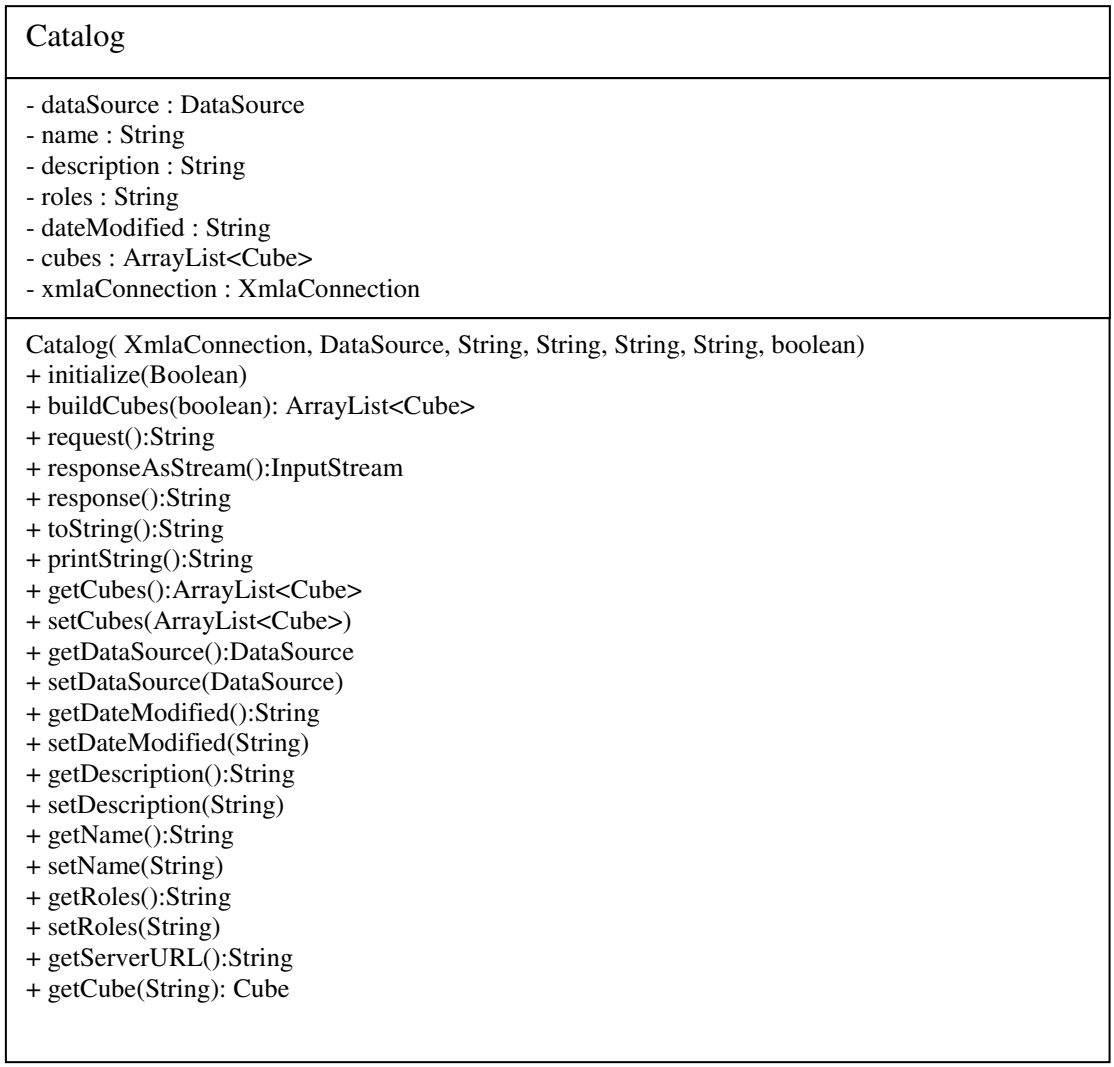
Anexo A – Diagrama de classes

Nesta anexo encontram-se as classes individualmente detalhadas utilizadas no projeto. Para visualizar o relacionamento entre as classes veja a figura 36.

XmlaConnection
- URL : String - restrictionList : HashMap - propertiesList : HashMap - requestType : String - statement : String
XmlaConnection(String) XmlaConnection(String, HashMap, HashMap) XmlaConnection(String, String) XmlaConnection(String, HashMap, HashMap, String) + getPropertiesList(): HashMap + setPropertiesList(HashMap) + getRequestType(): String + setRequestType(String) + getRestrictionList():HashMap + setRestrictionList(HashMap) + getURL():String + setURL(String url) + getStatement():String + setStatement(String) + addProperty(String, String) + addRestriction(String, String) + responseAsStream(String, String): InputStream + responseAsStream():InputStream + response (String, String): String + response():String + readFromStream(InputStream): byte[] + writeToStream(byte[], OutputStream) + request():String + buildXmlFromHashMap(HashMap): String + copy():XmlaConnection

Server
-serverURL : String -datasources : ArrayList<DataSource> -xmlaConnection : XmlaConnection
Server(String, boolean) + initialize(Boolean) + buildDataSources(boolean): ArrayList<DataSource> + request(): String + responseAsStream():InputStream + response():String + getDataSources():ArrayList<DataSource> + setDataSources(ArrayList<DataSource>) + getServerURL():String + setServerURL(String) + getDataSource(String): DataSource

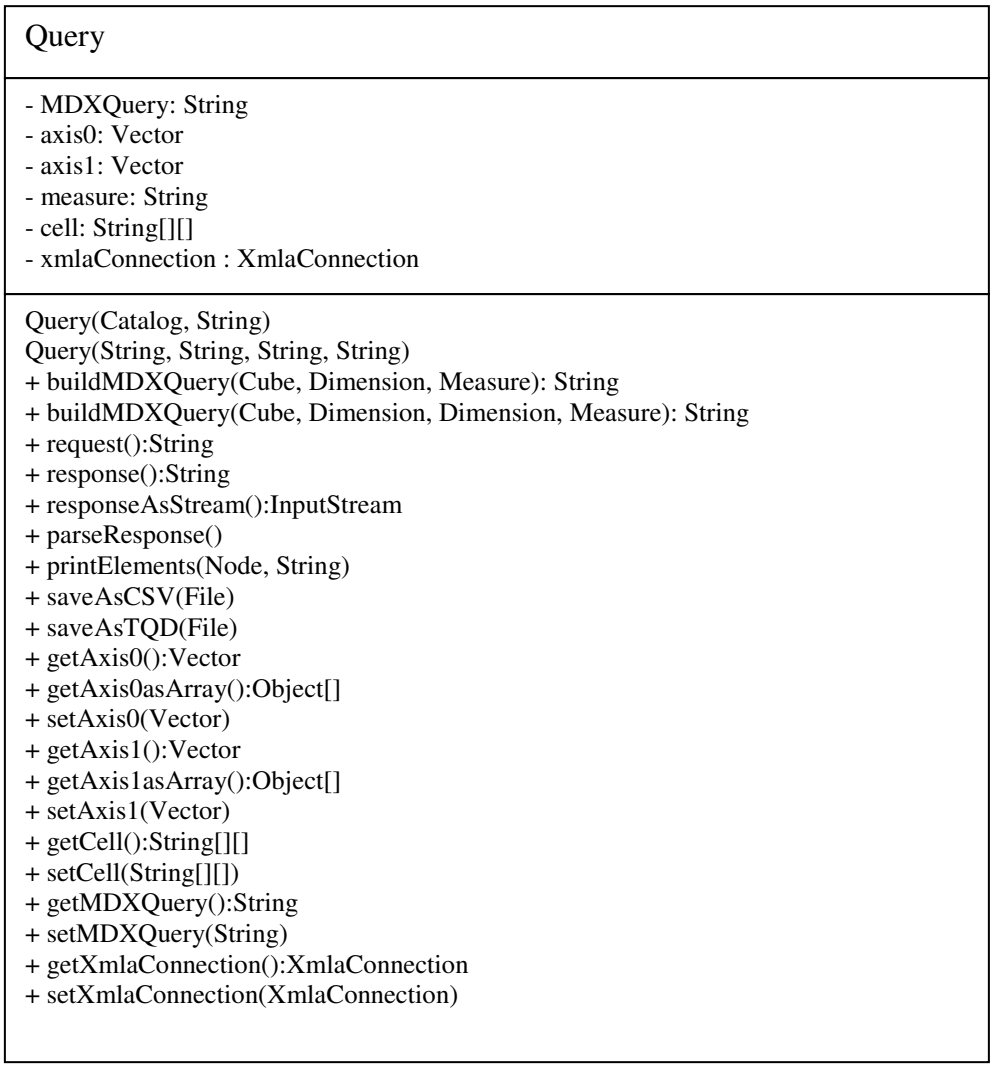

DataSource
- server : Server
- name : String
- description : String
- URL : String
- info : String
- providerName : String
- providerType : String
- authenticationMode : String
- catalogs : ArrayList<Catalog>
- xmlaConnection : XmlaConnection
DataSource(XmlaConnection, Server, String, String, String, String, String, String, String, boolean)
+ initialize(Boolean)
+ ArrayList<Catalog> buildCatalogs(boolean)
+ request():String
+ responseAsStream():InputStream
+ response():String
+ toString():String
+ printString():String
+ getAuthenticationMode():String
+ setAuthenticationMode(String)
+ getDescription():String
+ setDescription(String)
+ getInfo():String
+ setInfo(String)
+ getName():String
+ setName(String)
+ getProviderName():String
+ setProviderName(String)
+ String getURL()
+ setURL(String)
+ getProviderType():String
+ setProviderType(String)
+ getServer():Server
+ setServer(Server)
+ getCatalogs():ArrayList<Catalog>
+ setCatalogs(ArrayList<Catalog>)
+ getServerURL():String
+ getCatalog(String): Catalog


Catalog
- dataSource : DataSource
- name : String
- description : String
- roles : String
- dateModified : String
- cubes : ArrayList<Cube>
- xmlaConnection : XmlaConnection
Catalog(XmlaConnection, DataSource, String, String, String, String, boolean)
+ initialize(Boolean)
+ buildCubes(boolean): ArrayList<Cube>
+ request():String
+ responseAsStream():InputStream
+ response():String
+ toString():String
+ printString():String
+ getCubes():ArrayList<Cube>
+ setCubes(ArrayList<Cube>)
+ getDataSource():DataSource
+ setDataSource(DataSource)
+ getDateModified():String
+ setDateModified(String)
+ getDescription():String
+ setDescription(String)
+ getName():String
+ setName(String)
+ getRoles():String
+ setRoles(String)
+ getServerURL():String
+ getCube(String): Cube

Cube
- catalog : Catalog - name : String - type : String - lastSchemaUpdate : String - lastDataUpdate : String - isDrillThroughEnable : String - islinkable : String - isWriteEnabled : String - isSqlEnabled : String - dimensions : ArrayList<Dimension> - measures : ArrayList<Measure> - xmlaConnection : XmlaConnection
Cube(XmlaConnection, Catalog, String, String, String, String, String, String, String, String, boolean) + initialize(Boolean) + buildDimensions(boolean): ArrayList<Dimension> + buildMeasures(boolean): ArrayList<Measure> + request():String + responseAsStream():InputStream + response():String + toString():String + printString():String + getCatalog():Catalog + setCatalog(Catalog) + getIsDrillThroughEnable():String + setIsDrillThroughEnable(String) + getIslinkable():String + setIslinkable(String) + getIsSqlEnabled():String + setIsSqlEnabled(String) + getIsWriteEnabled():String + setIsWriteEnabled(String) + getLastDataUpdate():String + setLastDataUpdate(String) + getLastSchemaUpdate():String + setLastSchemaUpdate(String) + getName():String + setName(String) + getType():String + setType(String) + getServerURL():String + getDimensions():ArrayList<Dimension> + getTimeDimensions():ArrayList<Dimension> + getNonTimeDimensions():ArrayList<Dimension> + setDimensions(ArrayList<Dimension>) + getMeasures():ArrayList<Measure> + setMeasures(ArrayList<Measure>) + getDimension(String): Dimension + getMeasure(String): Measure


Dimension
- catalog : Catalog
- name : String
- type : String
- lastSchemaUpdate : String
- lastDataUpdate : String
- isDrillThroughEnable : String
- islinkable : String
- isWriteEnabled : String
- isSqlEnabled : String
- dimensions : ArrayList<Dimension>
- measures : ArrayList<Measure>
- xmlaConnection : XmlaConnection
Dimension(XmlaConnection, Cube, String, String, String, String, String, String, String, String, String, String, String, boolean)
+ toString():String
+ printString():String
+ getCaption():String
+ setCaption(String)
+ getCardinality():String
+ setCardinality(String)
+ getCube():Cube
+ setCube(Cube)
+ getDefaultHierarchy():String
+ setDefaultHierarchy(String)
+ getIsReadWrite():String
+ setIsReadWrite(String)
+ getIsVirtual():String
+ setIsVirtual(String)
+ getIsVisible():String
+ setIsVisible(String)
+ getName():String
+ setName(String)
+ getOrdinal():String
+ setOrdinal(String)
+ getType():String
+ setType(String)
+ getUniqueName():String
+ setUniqueName(String)
+ getUniqueSettings():String
+ setUniqueSettings(String)

Measure
- cube : Cube - name : String - uniqueName : String - caption : String - aggregator : String - dataType : String - precision : String - scale : String - isVisible : String - levelsList : String - nameSqlColumn : String - xmlaConnection : XmlaConnection
Measure(XmlaConnection, Cube, String, String, String, String, String, String, String, String, String, String, boolean) + toString():String + printString():String + getAggregator():String + setAggregator(String) + getCaption():String + setCaption(String) + getCube():Cube + setCube(Cube) + getDataType():String + setDataType(String) + getIsVisible():String + setIsVisible(String) + getLevelsList()String + setLevelsList(String) + getName():String + setName(String) + getNameSqlColumn():String + setNameSqlColumn(String) + getPrecision():String + setPrecision(String) + getScale():String + setScale(String scale) + getUniqueName():String + setUniqueName(String)


Query
- MDXQuery: String
- axis0: Vector
- axis1: Vector
- measure: String
- cell: String[][]
- xmlaConnection : XmlaConnection
Query(Catalog, String)
Query(String, String, String, String)
+ buildMDXQuery(Cube, Dimension, Measure): String
+ buildMDXQuery(Cube, Dimension, Dimension, Measure): String
+ request():String
+ response():String
+ responseAsStream():InputStream
+ parseResponse()
+ printElements(Node, String)
+ saveAsCSV(File)
+ saveAsTQD(File)
+ getAxis0():Vector
+ getAxis0asArray():Object[]
+ setAxis0(Vector)
+ getAxis1():Vector
+ getAxis1asArray():Object[]
+ setAxis1(Vector)
+ getCell():String[][]
+ setCell(String[][])
+ getMDXQuery():String
+ setMDXQuery(String)
+ getXmlaConnection():XmlaConnection
+ setXmlaConnection(XmlaConnection)