UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE CIÊNCIAS E TECNOLOGIA

Departamento de Sistemas e Computação

Coordenação de Pós-Graduação em Informática

DISSERTAÇÃO DE MESTRADO

ANÁLISE, PROJETO E IMPLEMENTAÇÃO DE UM ESQUEMA MOLAP DE DATA WAREHOUSE, UTILIZANDO O SGBD-OR ORACLE 8.I

Hilmer Rodrigues Neri

Campina Grande – PB

Fevereiro de 2002

UFPB – UNIVERSIDADE FEDERAL DA PARAÍBA

CCT - CENTRO DE CIÊNCIAS E TECNOLOGIA

DSC – DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO

COPIN - Coordenação de Pós-Graduação em Informática

DISSERTAÇÃO DE MESTRADO

ANÁLISE, PROJETO E IMPLEMENTAÇÃO DE UM ESQUEMA MOLAP DE DATA WAREHOUSE UTILIZANDO O SGBD-OR ORACLE 8.I

Hilmer Rodrigues Neri

Dissertação apresentada à coordenação de Pós-graduação em informática – COPIN – da Universidade Federal da Paraíba – UFPB, como requisito parcial para a obtenção do grau de Mestre em Informática.

Orientador: Marcus Costa Sampaio

Campina Grande – PB

Fevereiro de 2002

Análise, Projeto e Implementação de um Esquema MOLAP de Data Warehouse utilizando o SGBD-OR Oraclce 8i.

Hilmer Rodrigues Neri

Dissertação aprovada em 27/02/2002

Marcus Costa Sampaio, Doutor

Orientador

Marcus Costa Sampaio, Doutor

Componente da Banca

Jacques Philippe Sauvé, PhD

Componente da Banca

Maurício Ayala Rincón, Doutor

Componente da Banca

Campina Grande, 27/02/2002

Ficha catalográfica

NERI, Hilmer Rodrigues
N445A

Análise, Projeto e Implementação de um Esquema MOLAP de Data Warehouse utilizando o SGBD-OR Oracle 8i

Dissertação (Mestrado) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande, Paraíba, Fevereiro de 2002.

113 p. Il.

Orientador: Marcus Costa Sampaio

Palavras Chaves:

1. Banco de Dados
2. Data Warehouse
3. Molap
4. Chunk

CDU - 681.3.07.B

Dedico esta dissertação aos meus pais,
Joarez e Tania, pelo seu amor e dedicação, e
aos meus irmãos , pela nossa união.
O incentivo de vocês foi de fundamental
importância para a realização deste trabalho.

Agradecimentos

Gostaria de agradecer a meus pais (Joarez e Tania) e aos meus irmãos (Haller, Herman e Heiner) por me apoiarem a seguir o caminho que escolhi e por me incentivarem sempre, fazendo-me acreditar que sempre alcançaria meu objetivo.

A Priscila, por ter me mostrado novamente a beleza que existe em viver. Pelo seu despreendimento e amor, sempre mostrados, ao me ajudar me maneira irrestrita.

A toda minha família, mas principalmente a minhas avós, Amanda Nery e Helena Souza, que também sempre me incentivaram, oraram por mim.

Ao professor Doutor Marcus Costa Sampaio pelos ensinamentos, pela paciência, pela seriedade com que conduziu este trabalho e acima de tudo pela confiança em mim depositada.

Ao professor Doutor Jacques Sauvé pelos conhecimentos transmitidos e por praticamente ter-me adotado como um filho.

Ao professor Doutor Jorge Abrantes pela compreensão em meus momentos de dificuldades e por propiciar-me meus momentos de maior descontração, que eram os jogos de futebol.

Ao professor Maurício Ayala pelos esclarecimentos prestados, que tiveram um papel crucial para a realização deste trabalho.

A todos os funcionários da Copin, em especial, a Ana, Vera e Zeneide, pelo total apoio, torcida e prontidão em resolver meus problemas que a elas foram levados.

A todos amigos do mestrado, em especial a André, Alberto, Eduardo e principalmente a Rodrigo pela eterna troca de conhecimentos durante o curso e pelas sugestões sempre pertinentes e apropriadas.

Aos meus amigos de morada: Gustavo, Tarig, Guaraci e suas respectivas famílias, que propiciaram momentos de bastante descontração.

Ao amigo Leandro, que sempre esteve ao meu lado pronto a ajudar.

Ao amigo Atoji, que, mesmo distante, sempre se prontificou a ajudar e me apoiou em momentos bastante delicados.

Aos colegas Alexandre Gomes, Alexandre Magno, Alessandra Requena, David Lestani, Renato Oliveira, Arlon Santuche, Leonardo Antonialli e Tulius Lima que também me ajudaram nesta etapa da minha vida.

Aqueles que, direta ou indiretamente, me ajudaram e incentivaram, aqui não citados, mas que eternamente residem em minha memória e coração.

E por fim, a Deus, que sempre me iluminou e colocou todas estas pessoas no meu caminho, para que eu conseguisse alcançar êxito, nesta, que até o momento, foi a tarefa mais árdua da minha vida.

“Se todos os seus esforços forem
vistos com indiferença não desanime,
porque também o sol, ao nascer,
dá um espetáculo todo especial, e,
no entanto, a maioria da platéia
continua dormindo.”

(autor desconhecido)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ASP	Active Server Page
BD	Banco de Dados
CU	Caso de uso
DTD	Document Type Description
DW	Data Warehouse
DWing	Data Warehousing
ES	Especificação Sumplementar
GUI	Graphical User Interface
HTTP	Hipertext Transfer Protocol
JDBC	Java Database Connectiviy
JSP	Java Server Pages
MOLAP	Multidimensional OLAP
OLAP	Online Analytic Processing
OLTP	Online Transaction Processing
OO	Orientação a Objetos
OR	Objeto-Relacional
ROLAP	Relacional OLAP
RUP	Rational Unified Process
SGBD	Sistema de Gerência de Banco de Dados
SGBDR	Sistemas de Gerência de Banco de Dados Relacionais
SGBDOO	Sistemas de Gerência de Banco de Dados Orientado a Objetos
SGBDOR	Sistemas de Gerência de Banco de Dados Objeto-Relacional
SQL	Structure Query Language
UML	Unified Modeling Language
XML	Extensible Markup Language

Resumo

Com a emergência da tecnologia objeto-relacional em sistemas de gerenciamento de banco de dados (SGBD), várias vertentes de pesquisa têm direcionado seus estudos para avaliar a viabilidade de utilizar estes tipos de SGBD's em consultas OLAP. Este trabalho apresenta a análise, projeto e implementação de um protótipo que utiliza a tecnologia de armazenamento e recuperação de dados por meio de matrizes n-dimensionais (MOLAP) em consultas OLAP, fazendo uso dos recursos providos pelo sistema de gerência de banco de dados objeto-relacional (SGBD-OR) Oracle 8i. Foram implementadas algumas funções que provejam um mecanismo para realização de uma operação de consolidação, bem como um método para dispor os dados de uma matriz *n*-dimensional em tabelas do Oracle 8i. Foi realizada uma análise comparativa entre o desempenho do tempo/resposta de consultas OLAP realizadas pelo protótipo e por um esquema projetado no Oracle, que utilizava a tecnologia de armazenamento e recuperação de dados em tabelas relacionais (ROLAP), e isto nos mostrou vantagens de se utilizar o protótipo.

Abstract

With the emerging relational objects techonologies in relational database management systems (RDBMS), many researchers have directed their research efforts to evaluate the usability of such technology in OLAP queries. This work presents the analysis, design and implementation of a prototype that uses a storage and retrival technology based on n-dimensional matrices (MOLAP) in OLAP queries, making use of Oracle's Oracle 8i resources and functionalities. Some functions were implemented in order to provide a consolidation operation mechanism, as well as a solution for presenting/storing data from a n-dimension matrix into Oracle 8i tables.A comparative analysis between the OLAP response time performance made by the prototype and a Oracle provided ROLAP scheme was performed, showing the advantages of the proposed prototype.

Capítulo 1

Introdução

1.1 Contextualização

Quando a informação se encontra contextualizada como recurso empresarial, devemos considerá-la como bem patrimonial da empresa, assim como mesas, micro-computadores, prédios, máquinas e outros equipamentos e recursos. A informação, pertencente a este contexto, deve ser utilizada de maneira estratégica, para que possa atender muito rapidamente aos objetivos, metas e desafios traçados pela alta gerência da empresa.

No cenário atual de um mundo capitalista e globalizado, roga a máxima da excelência na qualidade total. As empresas encontram-se inseridas nestes ditames, e o que aponta o diferencial competitivo entre elas é a capacidade pela qual suas decisões são tomadas de forma ágil, rápida e eficaz.

Para que as decisões sejam tomadas, estratégias sejam bem planejadas, é necessário que as informações estejam disponíveis, de forma concisa e confiável a quem compete analisá-las. Acontece que muitas vezes estas informações estão dispersas nos mais variados meios de armazenamento, ou até mesmo em ambientes computacionais diferentes. No entanto, os executivos de gerência em nível tático e estratégico precisam fazer uso das informações de forma agregada e integrada. Desenvolver um software que padronize as informações oneraria bastante o custo, além de despender muito tempo. Então, surge a questão: como integrar, de forma consistente, as informações dispersas na empresa, de tal modo que as mais variadas análises corporativas possam ser efetuadas?

Análises corporativas são, em geral, multidimensionais e temporais. Uma dimensão é um critério de agregação de dados numéricos ou fatos, e reflete alguma característica do negócio. Séries temporais de agregados multidimensionais são empregadas para verificar a tendência do negócio.

Para equacionar o problema sobre a questão abordada, surgiu o conceito de Data Warehousing(DWing). De acordo com [1], Data Warehousing "é uma coleção de tecnologias de suporte à decisão disposta para capacitar o gerente a tomar as melhores decisões de forma rápida". DWing é todo um ambiente composto de ferramentas *back-end*, *front-end* e *Data Warehouse (DW)*.

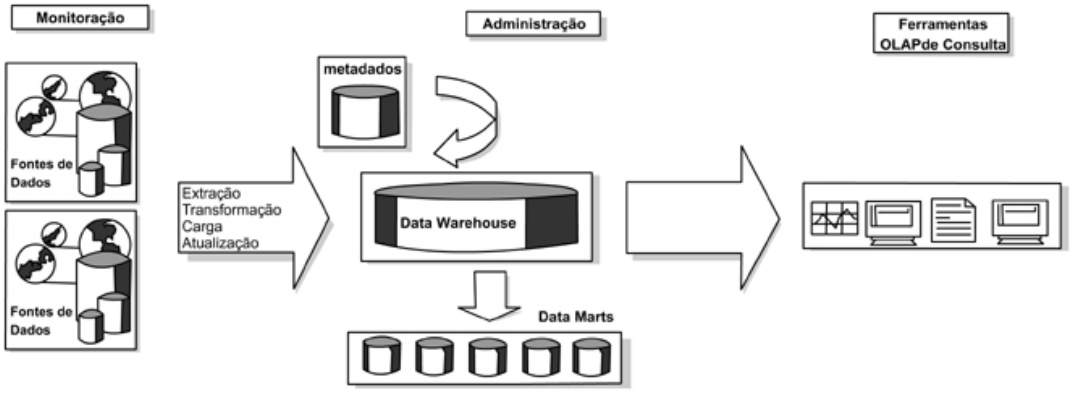

Figura 1: Arquitetura de um ambiente DWing com DW distribuído

Conforme ilustrado na figura 1, as etapas de extração, transformação, carga e atualização do DW, formam o back-end . As ferramentas de apoio às consultas OLAP (On-Line Analytical Processing, descritas no capítulo 2) ao DW representam o front-end. DW é apenas uma das tecnologias que compõem um ambiente maior, que é o DWing. De acordo com [2], DW "é um conjunto de dados baseados em assuntos, integrados, não-volátil, e variável em relação ao tempo, de apoio às decisões gerenciais".

As duas principais tecnologias de suporte a DWs são:

- ROLAP ("Relational OLAP"), que utiliza sistemas de gerência de banco de dados relacionais (SGDB-R);
- MOLAP ("Multidimensional OLAP"), que utiliza estruturas de armazenamento proprietárias de arrays[1] multidimensionais.

Os SGBD-R's apresentam baixo poder de tipagem. As deficiências mais tradicionais do modelo relacional são bem conhecidas [3]. Procurando suprir essas deficiências, os SGBD-R's têm evoluído no sentido de incorporar os conceitos do paradigma da orientação a objetos, resultando nos sistemas de gerência de banco de dados objeto-relacionais (SGBD-OR's). Os SGBD-OR's possuem característica peculiar por fornecer suporte a tipos de dados complexos.

[1] Sempre que for conveniente, este documento tratará array e matriz como sinônimos.

Com a emergência da tecnologia objeto-relacional, uma nova vertente de pesquisas tem se dedicado a estudar as vantagens de se utilizar SGBD-OR's na área de DWing. Mais precisamente, como os SGBD-ORs suportam tipos de dados complexos, por que não analisar a viabilidade de se construir DWs que utilizem a tecnologia MOLAP no topo de um SGBD-OR? Para responder a esta pergunta, construímos um protótipo MOLAP, utilizando o SGBD-OR Oracle 8i e testamos sua eficiência para realização de consultas OLAP.

Muitas companhias de destaque no cenário mundial na construção de ferramentas voltadas para análise multidimensional — incluindo Arbor Software [5] com o Essbase, Pilot [6] com o LightShip e a Oracle com Express from Oracle [7] —, já possuem ferramentas que utilizam a tecnologia MOLAP. No entanto, é do nosso conhecimento que suas implementações de arrays multidimensionais são mantidas em extremo sigilo por questões de estratégia de mercado [4].

Para a concepção do nosso protótipo MOLAP, utilizamos como direcionamento as idéias apresentadas em [4, 8]. Foi implementada uma interface WEB para uma aplicação que permite que os dados sejam "trafegados" da camada do usuário até a camada de persistência, a cargo de um SGBD-OR.

1.3 Objetivos do trabalho

O presente trabalho tem como principal objetivo a especificação e implementação de um ambiente de consultas OLAP, pautado na tecnologia MOLAP, utilizando recursos de um SGBD-OR de ampla aceitação no mercado. No contexto deste trabalho, o SGBD-OR utilizado foi o Oracle 8i.

A solução desenvolvida, chamada de Protótipo MOLAP, provê uma interface de consulta OLAP baseada na WEB, apresenta uma solução para projetar esquemas MOLAP em um SGBD-OR, utiliza a tecnologia MOLAP para a implementação de algumas funções para recuperação e consolidação de dados, e oferece tempos de respostas inferiores das funções implementadas em relação às consultas OLAP baseadas em tecnologia ROLAP, realizadas neste trabalho.

Objetivos Gerais

1) Construção de um protótipo MOLAP no SGBD-OR Oracle 8i.

2) Verificação da viabilidade de se utilizar a tecnologia objeto-relacional para construção de esquemas MOLAP.

3) Utilização do protótipo para a avaliação comparativa do desempenho das tecnologias ROLAP e MOLAP.

Objetivos Específicos

1) Implementar as funcionalidades de:

   - Armazenamento de dados de arrays multidimensionais;
   - Recuperação de células de arrays multidimensionais.
   - Compressão dos dados de arrays multidimensionais.

1.3 Relevância do trabalho

A relevância deste trabalho está pautada nos seguintes aspectos:

- Até o limite do nosso conhecimento, não existe na literatura registro de esforço da utilização direta do SGBD-OR Oracle 8i, ou seja, não há ferramenta de apoio, para concepção de um esquema que possibilite trabalhar com arrays multidimensionais.
- Eliminação da "esparcidade" de arrays por meio da técnica de compressão de dados apresentada em [4].
- Navegação uniforme em um array, pois não há favorecimento de nenhuma dimensão.
- Com relação ao armazenamento de dados dirigido para o negócio é feita uma distribuição dos dados de acordo com a probabilidade de acesso, segundo o perfil das consultas previstas.
- Faz-se aqui a proposta de uma metodologia para o projeto de um esquema MOLAP utilizando o SGGB-OR Oracle 8i.
- Confirma-se a possibilidade de consultas usando tecnologia MOLAP com tempo de resposta melhor que com tecnologia ROLAP.

1.4 Organização do Trabalho

Este trabalho está estruturado em capítulos. O capítulo 1 é a introdução.

No capítulo 2 são descritos, de forma sucinta, alguns conceitos que envolvem um ambiente de DWing, bem como idéias para se construir um esquema MOLAP no topo de SGBD-OR's.

O capítulo 3 detalha o projeto do protótipo: artefatos do "Rational Unified Process" RUP, gerados nas fases de inspeção, elaboração e construção do protótipo, bem como a fase de transição.

O capítulo 4 avalia o nível de satisfação dos requisitos funcionais e não funcionais estabelecidos para o protótipo. Além disso, compara o desempenho do protótipo relativamente a um DW com tecnologia de suporte ROLAP.

A conclusão e as perspectivas do trabalho são o objeto do capítulo 5.

Seguem-se a bibliografia e os seguintes apêndices:

- no apêndice A são apresentadas algumas técnicas da moderna Engenharia de Software, como: - a linguagem para modelagem de sistemas "Unified Modeling Language" (UML), – Design Patterns, que são padrões de projeto para software orientado-objeto, e uma visão geral do RUP;
- no apêndice B é apresentado o arquivo "Extensible Markup Language" (XML), que descreve os metadados para o protótipo;
- os apêndices C e D descrevem, respectivamente, o arquivo utilizado para análise de desempenho da aplicação e um exemplo resumido de como os dados estão organizados nos chunks;
- ao final, no apêndice E, é apresentada a tabela com todos os tempos de execução das consultas realizadas no Protótipo MOLAP e no Esquema em Estrela.

Capítulo 2

Data Warehousing: Uma visão Geral

Este capítulo apresenta algumas definições necessárias à compreensão dos conceitos brevemente abordados na introdução. Inicialmente, são apresentadas as definições do conceito de aplicação OLAP. Em seguida, são analisadas as duas principais tecnologias de suporte a aplicações OLAP (ROLAP e MOLAP). Subseqüentemente, são apresentadas algumas técnicas que nos forneceram suporte para construção do protótipo.

2.1 Interface OLAP de apoio à decisão

Basicamente existem duas grandes visões para classificar aplicações. São elas:

- aplicações do negócio;
- aplicações que tratam do negócio.

Baseando-nos nesta classificação, definimos aplicações OLTP ("On Line Transaction Processing") como aplicações voltadas ao negócio e suportadas por bancos de dados que chamaremos de operacionais. Neste tipo de aplicação, as transações requerem dados atualizados, consistentes e tolerantes a falhas. A maximização de transações concorrentes estabelece melhor desempenho. Já as aplicações OLAP ("On Line Analitical Processing") são aplicações que tratam do negócio. O termo OLAP foi inicialmente proposto por [5], desde então vem sendo utilizado para definir este tipo de aplicação. Nestas aplicações dados sumariados e históricos são mais importantes que dados atômicos. As aplicações OLAP são suportadas por DWs. O objetivo destas aplicações é prover ao usuário a visualização dos dados sob diferentes ângulos gerenciais e ser capaz de suprir toda a demanda da atividade de análise.

A tabela 1 mostra um comparativo entre aplicações OLTP *versus* aplicações OLAP.

OLTP	*OLAP*
Controle operacional	**Tomada de decisão**
Atualização de dados	**Análise de dados**
Pequena complexidade das operações	**Grande complexidade das operações**
O mais baixo nível de agregação dos dados	**Alto nível de agregação dos dados**
Não armazena dados históricos	**Armazena dados históricos**
Voltada ao pessoal operacional	**Voltada aos gestores do negócio**

Tabela 1: Características OLTP versus OLAP

2.2 Análise Multidimensional

Análise Multidimensional é uma atividade de tomada de decisão, por um executivo de uma empresa, que consiste essencialmente em fazer comparações para descobrir *tendências*, e em *fazer perguntas* para saber, com a riqueza de detalhes necessária, porque um determinado evento aconteceu.

O objetivo da análise multidimensional é tornar o conteúdo dos repositórios de dados transparente e intuitivo para análise dos usuários finais. A abordagem multidimensional aproxima o conteúdo dos dados com o modelo mental do analista, de modo a reduzir a complexidade e diminuir a incidência de interpretações errôneas. Devido ao fato de os dados estarem fisicamente armazenados em uma estrutura multidimensional, a velocidade das operações é muito mais rápida e muito mais consistente (os dados já foram tratados) do que se é possível realizar em outras estruturas de banco de dados. Esta combinação de simplicidade e rapidez é um dos grandes benefícios da análise multidimensional.

Os dados multidimensionais estão relacionados com várias dimensões. Uma dimensão é uma entidade independente abstraída do negócio da organização (ex. promoção, produto, loja, região, etc.) e que serve como um critério de sumarização de medidas ou fatos da organização. Uma medida ou um fato está normalmente relacionada com várias dimensões (multidimensional). Assim, por exemplo, podemos visualizar o fato vendas por loja, por produto e por região. Fatos podem ser *aditivos*, *semi-aditivos* ou *não-aditivos*. Fatos aditivos podem ser somados (agregados) através de todas as dimensões. Exemplos: total de um produto por região, total de todos os produtos por região, total da região por produto, total geral, etc. Fatos semi-aditivos não podem ser somados ao longo de uma dimensão, mas ainda podem ser somados em algumas operações. Por exemplo: saldos de uma conta não podem ser somados ao longo do tempo. Entretanto, a média dos saldos num certo período de tempo é

uma informação muito útil. Não faz sentido somar fatos não-aditivos. Em geral, valores percentuais são não-aditivos.

Um aspecto importante da análise multidimensional é a flexibilidade para que se responda a questões em quatro espaços de apoio à decisão [11]. São eles:

- Espaço de Dados (Qual é o preço de um determinado produto?);
- Espaço de Agregação (Quais são as vendas mensais de produtos para uma loja?);
- Espaço de Influência (Que fatores influenciaram as vendas em uma determinada loja?);
- Espaço de Variação (Como têm se comportado as vendas nos três últimos meses?).

Para conseguirmos obter respostas nos espaços de apoio à decisão, destacamos outro importante componente da análise multidimensional: a operação de consolidação dos dados. Para os executivos das empresas são mais úteis e significativas as agregações (ou sumarização) dos valores que descrevem a situação de seus negócios.

A Consolidação é uma das mais importantes operações OLAP. Ela envolve a agregação de dados sobre uma ou mais hierarquias de dimensões. A generalização de um "query"[2] de consolidação pode ser representada formalmente através de:

Select P, $F_1(m_1)$, ... , $F_p(m_p)$
From $C(D_1(A_{11}), \ldots , D_n(A_{n1}))$
Where $\phi(D_1)$ **and** ... **and** $\phi(D_n)$
Group by G

nos quais, P representa os atributos a serem selecionados das dimensões. $F_i(m_i)$ para $(1 \leq i \leq p)$ representa uma função de agregação. Talvez a parte mais estranha deste query seja a cláusula **From** $C(D_1(A_{11}), \ldots , D_n(A_{n1}))$, indicando que a fonte de dados para o query está no cubo C indexado por suas tabelas dimensões. Cada uma das dimensões é referenciada como $D_1, \ldots , D_n$. Cada dimensão D_i contêm K_i atributos $D_i(A_{i1}) , \ldots, D_i(A_{ik_i})$ que descrevem a dimensão. Temos também $\phi(D_i)$, sendo o predicado $D_i(A_{ij}) = v_{ij}$, onde $v_{ij} \in dom(D_i(A_{ij}))$ $(1 \leq i \leq n, 1 \leq j \leq K_i)$. Também temos que $G \subseteq \{D_i(A_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq K_i\}$ e $P \subseteq G$ segundo apresentado em [4]. Neste trabalho implementamos a operação de consolidação descrita através de um algoritmo apresentado na seção 2.5.6.

Pelas diferenças discutidas até aqui, a modelagem para repositórios OLAP distancia-se sobremaneira das técnicas usadas para BD's operacionais. A técnica utilizada para construção de repositórios OLAP é a Modelagem Multidimensional[12].

[2] Sempre que for conveniente, este documento tratará query e consulta como sinônimos.

Existem duas grandes vertentes para Modelagem Multidimensional:

- servidores ROLAP;
- servidores MOLAP, os quais descrevemos nas subseções seguintes.

2.3 Servidores ROLAP(Relational OLAP)

Servidores ROLAP utilizam tabelas relacionais para simular um cubo multidimensional.

Figura 2: Esquema em estrela com 4 dimensões

Na Figura 2 [12] podemos ver um esquema genérico de DW que concentra numa tabela, chamada Tabela de Fatos, as medidas ou fatos do negócio, ou seja, as células de uma matriz conceitual multidimensional. As dimensões da matriz conceitual multidimensional são também representadas por tabelas — Tabelas de Dimensão (Dimensão$_i$ na figura). A chave da tabela de fatos é composta das chaves das tabelas de dimensão: desta forma, os relacionamentos tabela de fatos – tabelas de dimensão ficam implícitos, como é o caso do Modelo Relacional, formando uma estrutura semelhante a uma estrela, originando assim o nome esquema relacional com estrutura em estrela (“Star Squema”) [12].

A tabela de fatos tem geralmente alta cardinalidade, enquanto as tabelas dimensão são normalmente pequenas.

Um exemplo de uma consulta SQL ao esquema apresentado na figura 2 é descrito a seguir:

```
Select Dimensao1.AtributoDescricao1, Dimensao2.AtributoDescricao1,
       Dimensao3.AtributoDescricao1, Dimensao4.AtributoDescricao1
       sum(TabelaDeFato.Atributo1DoFato)
From TabelaDeFato, Dimensao1, Dimensao2,Dimensao3, Dimensao4
Where TabelaDeFato.#Dimensao1=Dimensao1.#Dimensao1 and
      TabelaDeFato.#Dimensao2=Dimensao2.#Dimensao2 and
      TabelaDeFato.#Dimensao3=Dimensao3.#Dimensao3 and
      TabelaDeFato.#Dimensao4=Dimensao4.#Dimensao4 and
      Dimensao1.AtributoDescricao1= D1 and Dimensao2.AtributoDescricao1= D2 and
      Dimensao3.AtributoDescricao1= D3 and Dimensao4.AtributoDescricao1= D4
Group by Dimensao1.AtributoDescricao1,Dimensao2.AtributoDescricao1;
```

A grande vantagem da tecnologia ROLAP é a maturidade da tecnologia relacional subjacente, com seu amplo universo de produtos e soluções. Este sucesso enorme se deve à:

- simplicidade do modelo de dados;
- interface declarativa padrão SQL;
- segurança / integridade;
- processamento otimizado de consultas convencionais (pelo menos a maioria dos tipos de consulta no mundo dos negócios);
- "esparcidade" nula: em um DW ROLAP, em geral a tabela de fatos primária é bastante esparsa, isto é, há muito menos valores da chave composta do que o produto cartesiano dos valores das chaves de cada dimensão. Nesta abordagem isto é bom porque, do contrário, o tamanho da tabela primária tenderia ao "infinito". Por exemplo: Supermercados X - Dos 30.000 produtos em estoque, somente 3.000 aproximadamente (10%) são comercializados diariamente, por loja. O que acontece na prática é que somente os fatos registrados nas tabelas dos sistemas OLTP vão para a tabela de fatos do DW.

Existem algumas situações, entretanto, nas quais se deve ponderar o uso de servidores ROLAP, como por exemplo, quando o esquema possui muitas tabelas dimensões e estas possuem uma quantidade grande de registros e atributos. O grande número de operações "join", necessário para processar as consultas, comprometeria o desempenho do tempo de resposta das mesmas. Alguns outros fatores devem ser considerados quanto à utilização de servidores ROLAP:

- a otimização do esquema é intensiva, já que não existe um mapeamento direto das tabelas relacionais para o modelo multidimensional;
- a administração dos metadados deve ser rigorosamente policiada, face à necessidade de se definir claramente o comportamento multidimensional das tabelas utilizadas no esquema do DW;
- falta de estruturas de dados adequadas para sistemas OLAP.

Apesar de apresentar estas desvantagens, atualmente, com muita criatividade, na maioria das vezes da parte dos projetistas de DW, os DW Relacionais atendem satisfatoriamente às exigências de aplicações OLAP.

2.4 Servidores MOLAP (Multidimensional OLAP)

Nesta tecnologia, um DW é um array multidimensional, ou *cubo de dados*[3], ou simplesmente *cubo*, cujas células contêm valores medidos; os lados do cubo definem as dimensões dos dados. Pode-se ter mais de três dimensões no tecnicamente chamado de hiper-cubo, apesar de normalmente os termos cubo e cubo de dados serem usados como sinônimos de hiper-cubo [12].

Considere-se que um BD contém dados de venda relativos a produtos, datas e fornecedores. O valor vendas é funcionalmente determinado pelos valores dos outros atributos. Podemos ter uma visão "hiper-cúbica" destes fatos como ilustrado na Figura 3.

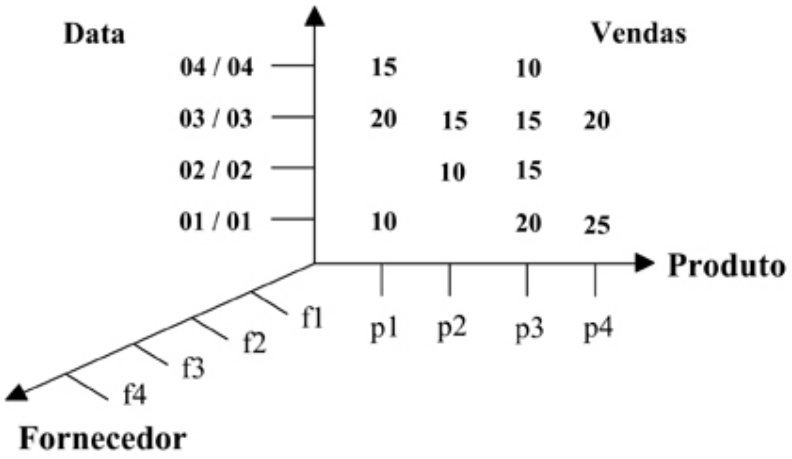

Figura 3: Exemplo de um Cubo de dados (a)

[3] Aliás, esta é mais uma nomenclatura metaforicamente adotada, posto que matematicamente um cubo tem três, e apenas três, dimensões.

Atributos determinantes como Produto, Data, Fornecedor, etc., são chamados de dimensões, enquanto os atributos determinados são chamados de medidas, fatos, agregados ou sumários.

Agora observemos um outro exemplo da representação de um cubo de dados ilustrado na figura 4:

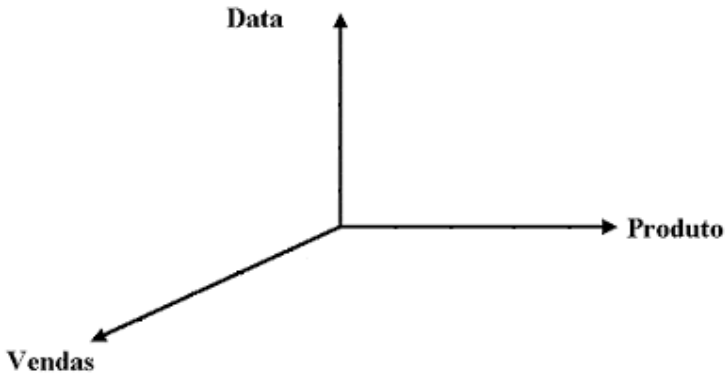

Figura 4: Exemplo de um Cubo de dados (b)

No cubo (ou hiper-cubo, ou cubo de dados) da Figura 3, vendas são medidas, enquanto no outro cubo da figura 4, vendas são valores da dimensão Vendas. Isto significa dizer que dimensões e medidas podem ser tratadas simetricamente.

Tipicamente, dimensões são associadas com *hierarquias,* envolvendo atributos que as descrevem. Estas hierarquias representam os diferentes níveis de agregação dos fatos. Por exemplo, a dimensão *Loja,* descrita pelos atributos nome_loja, cidade, estado e região, entre outros atributos, poderia ter associada a ela a hierarquia, *nome_loja* → *cidade* → *estado* → *região*, e outras hierarquias. Similarmente, a dimensão *Produto* poderia ter *produto* → *categoria* → *família*.

As principais operações relacionadas a cubos de dados (também conhecidas como operações OLAP) são:

- Pivoting - rotação do cubo, para mostrar uma face particular.
- Slicing-dicing - seleção de um subconjunto do cubo.
- Operações em hierarquias: drill-up, drill-down - a primeira, agregação de "baixo" para cima em uma hierarquia; a segunda, agregação de "cima" para "baixo" em uma hierarquia.
- Drill-across - concatenação de vários cubos.
- Séries temporais - comparações de medidas no tempo.

Os operadores multidimensionais básicos, a partir dos quais os operadores OLAP podem ser construídos são: Push, Pull, Restriction, Destroy, Join, Merge, definidos através de um modelo multidimensional em [14]. Os cinco primeiros operadores são primitivas. Este modelo multidimensional captura as funcionalidades providas pelos produtos OLAP no mercado. No entanto, ele vai além, oferecendo ainda:

- tratamento simétrico de dimensões e medidas;
- flexibilidade: múltiplas hierarquias, agregados ad hoc;
- conjunto pequeno de operadores básicos;
- compatibilidade com a álgebra relacional: operadores facilmente traduzíveis para SQL;
- analogia com os operadores relacionais: projeção, união e interseção (a diferença pode ser obtida de interseção e união) podem ser expressos em termos dos operadores propostos;
- poder de expressão: os operadores OLAP podem ser facilmente definidos em termos dos operadores apresentados.

Baseando-nos neste modelo apresentamos o exemplo de uma consulta ao cubo dados com as dimensões: *tempo*, *produto*, *loja*, ilustrado na figura 5. O que queremos saber é: Qual a quantidade de meias vendidas durante o meses de Março e Abril de 2000 na Loja E?

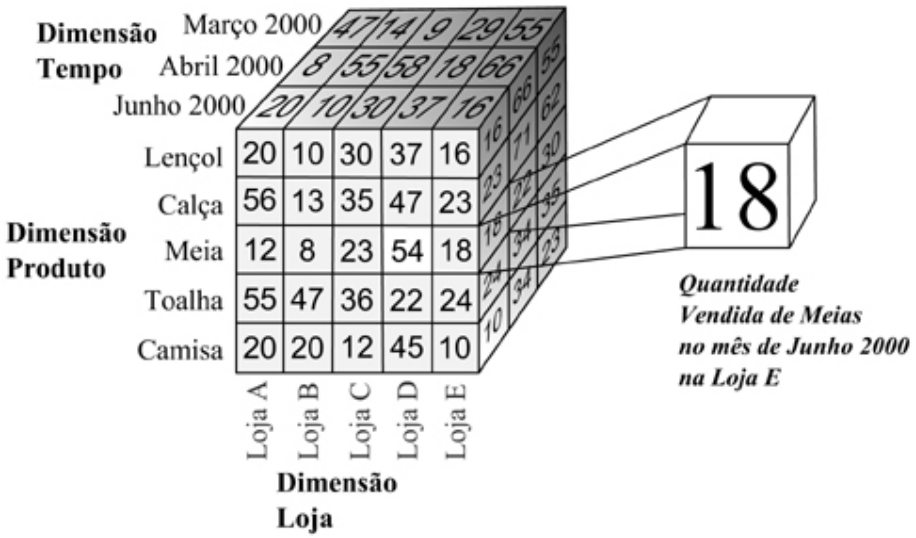

Figura 5: Cubo de dados Vendas

Um exemplo de uma consulta baseada em [14] no esquema apresentado na figura 5 é descrito a seguir:

```
1 restrict(C, Loja, nome_loja = Loja E) = C1.
2 destroy(C1, Loja) = C2.
3 restrict(C2, mês-ano , trimestre between {03/2000, 04/2000}) = C3.
4 merge(C3, {[Quantidade por trimestre]}, SUM) = Cans.
```

O operador restrict é executado sobre uma dimensão de um cubo, removendo os valores da dimensão que não satisfazem a condição estabelecida. Nota-se que este operador realiza a operação slicing/dicing da terminologia OLAP.

O operador destroy remove uma dimensão que tem um único valor no seu domínio.

O operador *merge* é uma operação de agregação. Múltiplos membros de elementos do cubo são agredados para produzir uma dimensão com um domínio possivelmente menor ("drill up"). A função de agregação é especificada de maneira *ad hoc*.

Então, na linha 1, *restrict(C, Loja, nome_loja = Loja E) = C1* recebe como entrada o cubo da dados da figura 5 através do parâmetro C, a dimensão a ser restringida (Loja) e na dimensão Loja, a loja a ser restringida (Loja E). C1 é o cubo-resposta tendo sido removidas a demais lojas (Loja A, Loja B, Loja C e Loja D).

Na linha 2, *destroy(C1, Loja) = C2* recebe como entrada o cubo C1, descrito no parágrafo anterior, e a dimensão com um valor único é removida (Loja). C2 é o cubo resposta com duas dimensões (Produto, Data).

Na linha 3, *restrict(C2, mês-ano , trimestre in {01/2000, 03/2000}) = C3* recebe como entrada o cubo C2, descrito no parágrafo anterior, a dimensão a ser restringida (Data) e na dimensão Data, as datas a serem restringidas (01/2000, 02/2000, 03/2000). C3 é o cubo resposta, tendo sido removido o mês de Junho.

Na linha 4, *merge(C3, {[Quantidade por trimestre]}, SUM) = C_{ans}* recebe como entrada o cubo C3, descrito no parágrafo anterior, o tipo de agregação (Quantidade por trimestre), e a função de agregação (SUM). C_{ans} é o cubo-resposta, contendo os valores das quantidades agregadas (52).

A principal vantagem da utilização desta tecnologia é a melhoria do tempo de resposta a consultas, pois não existe a utilização da operação "join", que onera o processamento de consultas entre tabelas que estão relacionadas através de uma chave estrangeira [4]. Algumas outras vantagens podem ser citadas:

- esquema natural para modelagem dimensional;
- sumarização fácil, visto que uma célula do cubo já é posição natural que descreve uma medida relativa às dimensões componentes dos lados do cubo.
- suporte natural para operações OLAP.

A principal desvantagem de DWs MOLAP em relação a DWs ROLAP diz respeito ao volume do cubo de dados. Cubos são geralmente esparsos. Voltando ao nosso exemplo, nem todo produto pode ser vendido por todos os fornecedores durante todo um período de tempo. Isto pode acarretar um grande problema, pois um cubo de 100MB `líquidos´ poderia facilmente requerer, a título de ilustração, 10GB. Algumas outras desvantagens podem ser citadas:

- tecnologia proprietária: a arquitetura não é aberta, o que equivale a dizer que cada fabricante de DW's MOLAP possui uma implementação proprietária.
- ausência de padrões: não existe um modelo definido no mercado que se caracterize como um padrão, como é o caso do modelo relacional.

Para concluir, MOLAP OU ROLAP? Qual escolher? Para quem vai utilizar a tecnologia o mais importante é entender os negócios da empresa para então decidir pela solução que melhor atende o volume de dados, às necessidades de análise da empresa, o estágio de organização das fontes de informação e a capacitação da equipe.

O objeto de estudo deste trabalho é a construção de uma estrutura MOLAP em um SGBD-OR, o que é discutido nas seções subseqüentes.

2.5. Uma implementação para Cubos de Dados

Uma primeira questão a ser considerada é: Como implementar um cubo de dados em um SGBD-OR que está pautado no modelo Relacional?

Na nossa implementação, um DW é assim definido:

- um array *n*-dimensional mapeado em uma tabela, representando os fatos. Esta implementação é minuciosamente explicada no capítulo 3;
- cada dimensão do array é representada por uma tabela relacional, cujas colunas são a chave da dimensão e os atributos descritivos da mesma;
- as tabelas de dimensão são indexadas — árvores-B — pelas chaves respectivas.

Este trabalho é inspirado pelas idéias apresentadas em [4,8]. Mas ele vai além, oferecendo:

- uma implementação de uma estrutura MOLAP no SGBD-OR Oracle 8.i, sem a necessidade de nenhuma ferramenta de apoio adicional;
- suporte a metadados através da utilização de XML;
- melhor desempenho de consultas OLAP, pautado na tecnologia MOLAP em relação à tecnologia ROLAP, utilizando o Oracle 8.i;
- uma sugestão das etapas para projetar um DW MOLAP, utilizando o Oracle 8.i;
- possibilidade de que consultas OLAP sejam realizadas através de um browser.

Voltemos ao nosso DW-exemplo, com as dimensões *Produto*, *Loja* e *Tempo*, e cujos fatos são as vendas por produto, por loja e por data (p.e., dia).

Sejam pid, lid e tid os atributos-chave das dimensões. Sejam *nome*, *categoria e família* os demais atributos de Produto; *cidade*, *estado* e *região* os demais atributos de Loja; e *dia*, *mês* e *ano* os demais atributos de Tempo. As tabelas são Tempo, Loja e Produto, com os índices I-tid, I-lid e I-pid, respectivamente. O array tridimensional é Vendas (Produto, Loja, Tempo).

Suponhamos que uma linha da tabela Tempo com mês = 03 e ano = 2001 seja identificada por tid = t1003. Para achar a célula do array Vendas para pid = p002, lid = l098 e t1003, devemos mapear p002, l098 e t1003 para as entradas — números inteiros — nas dimensões correspondentes de Vendas. Como o mapeamento é feito está ilustrado na figura 4. Por dificuldade para a representação de arrays tridimensionais, a figura 6 exibe somente a projeção de Vendas por (Produto, Loja). As entradas em Vendas são

encontradas nos índices I-tid, I-lid e I-pid. Pode-se ver, na figura 6, que o valor pid = p002 é mapeado para o inteiro 5, que é o valor da dimensão Produto em Vendas. Supondo que l098 é mapeado para 108, e que t1003 o é para 700, a célula correspondente é Vendas (5, 108, 700).

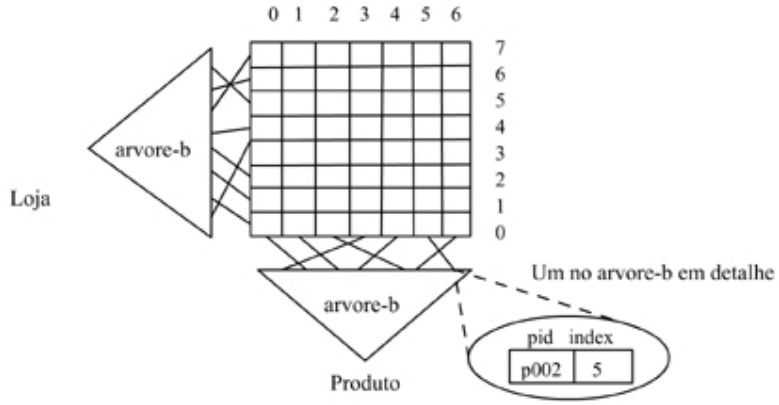

Figura 6: Estrutura de um array bidimensional

Como o SGBD-OR Oracle 8.i não provê suporte para implementação de um array *n*-dimensional, só suporta a utilização de arrays unidimensionais (ao menos diretamente); apresentamos nas seções seguintes algumas técnicas que foram utilizadas nesse trabalho para permitir a implementação do nosso array *n*-dimensional.

2.5.1 Padrão de acesso

Utilizamos esta técnica para otimizar o layout do nosso array n-dimensional, em face da necessidade de fazermos sua implementação no topo da tecnologia relacional.

Considere um array *n*-dimensional *A*. Cada consulta do usuário ao array pode ser modelada como um array *n*-dimensional $\subseteq$ A. Além disso, as consultas podem ser agrupadas em *classes* $C_1,...,C_K$, de tal forma que cada C_i define um array *n*-dimensional, que é o conjunto-resposta de uma consulta, definido pelos intervalos I_{i1}, $I_{i2},..., I_{in}$ das dimensões *D1, ..., Dn* de *A*. Por último, uma consulta$_i$ ocorre com uma probabilidade P_i. Um padrão de acesso é definido formalmente em [8] como:

$$\{(P_i, I_{i1}, I_{i2},..., I_{in}): 1 \leq i \leq K\}$$

A figura 7 ilustra um exemplo de um array bidimensional, 10 x 10. As três áreas hachuradas representam o acesso em duas classes de ordem — [3X4] e [5X3] — correspondentes às seguintes distribuições: {($^1/_3$,[2..4] , [2..5]), ($^1/_3$, [7..9], [2..5]), ($^1/_3$,[1..5],[8..10])}. Desta forma teríamos os padrões de acesso definidos como:

$$\{(^2/_3,[3X4]), (^1/_3, [5X3])\}$$

Optamos por representar as distribuições através da notação de intervalos, pois percebemos que essa notação elimina a ambigüidade de representar as regiões de um array. Nossa representação é uma extensão da representação apresentada em [8].

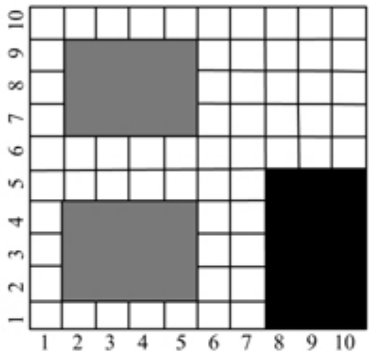

Figura 7: Padrões de acesso a um array bidimensional

O intuito de definir um padrão de acesso é identificar regiões de um array onde se concentram as consultas realizadas pelos gerentes de negócio, incluindo as probabilidades de suas ocorrências.

Um padrão de acesso pode ser definido tanto pelo usuário final, quanto através de estatísticas dos acessos ao banco de dados (array).

2.5.2 "Chunking" arrays

Nesta seção apresentamos a técnica "chunking" como solução a um grande problema a resolver quando se acessam arrays: como fazer com que a navegação no array seja ortogonal à ordem lógica dos índices do array? Também são apresentados alguns dos outros problemas a serem resolvidos quando se utiliza arrays para armazenamento e recuperação de dados em SGBD's:

- esparcidade – este problema foi resolvido com a utilização da técnica de compressão "chunk-offset", tratado na seção 2.5.5;
- como implementar um array multidimensional num SGBD que só suporta (ao menos diretamente) arrays unidimensionais, p.e Oracle? O capítulo 4 apresenta uma solução completa de todas as etapas do desenvolvimento do nosso array, bem como um estudo de caso exemplificando o acesso ao nosso array.

A técnica padrão de armazenamento de arrays utilizada nas linguagens de programação, onde os elementos do array são acessados na ordem de linhas e colunas, não é muito eficiente dentro do contexto de aplicações MOLAP.

Considere-se a representação de um array bidimensional com as dimensões Loja (linha) e Data (coluna). Considere-se também que a ordem de armazenamento das células do array é por linha. Acessar o array por loja é eficiente, já que cada página lida do disco irá conter um `cluster´ de linhas de lojas. Entretanto, acessar o array logicamente por datas, em geral não é eficiente, por não estar de acordo com a ordem física dos dados.

Em síntese, o armazenamento de arrays *n*-dimensionais, privilegiando uma dimensão na ordenação das `linhas´ do array, causa uma assimetria entre as dimensões, favorecendo o acesso à dimensão privilegiada, em detrimento dos acessos pelas outras dimensões.

Para proporcionar um acesso uniforme, em termos de desempenho, para todas as dimensões de um array *n*-dimensional, uma técnica utilizada é a de dividir o array em pedaços (chunks), como sugerido em [13]. Um chunk é uma restrição do arrray *n*-dimensional, de tal maneira que ele caiba numa página ou bloco de disco. Um bloco é a unidade de transferência utilizada pelos sistemas operacionais para transportar dados entre a memória principal e o disco.

Como resultado, chunks minimizam o número médio de blocos recuperados para um determinado padrão de acesso. Na prática, isto equivale a dizer que o desempenho das consultas a um array se torna ortogonal à navegação no array.

Vamos ilustrar esta afirmação com um exemplo. A figura 8(a) mostra um array tridimensional de tamanho $X_1 = 100$, $X_2 = 2000$ e $X_3 = 8000$, armazenado por `linha´. A figura 8(b) mostra o mesmo array armazenado utilizando chunks.

O array está armazenado em um disco e a transferência de dados entre a memória principal e o disco ocorre em páginas de 8KB.

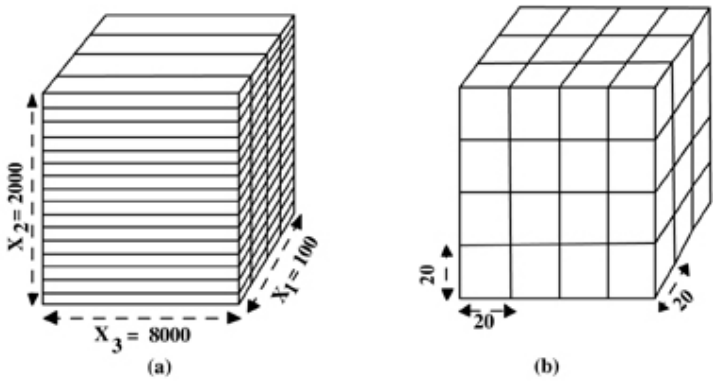

Figura 8: Estrutura de um array tridimensional divido em Chunks [8]

Se o array é armazenado como na figura 8(a), ele fica assim internamente, privilegiando o acesso à terceira dimensão:

{ (1,1,1), (1,1,2), ..., (1,1,8000), (1,2,1), (1,2,2), ..., (1,2,8000), ..., (100,2000,8000) }

Para o array apresentado na figura 8, definimos dois padrões de acesso para as distribuições: {($^1/_2$, [10..20], [200..600], [1500..1510]), ($^1/_2$, [50..70], [900..905], [7000..7400])}. Assim, temos os seguintes padrões:

{ ($^1/_2$, [10 X 400 X 10]), ($^1/_2$, [20 X 5 X 400]) }

Considere-se a recuperação de células do array (a) apresentado na figura 8. Uma célula do array tem o tamanho 1 byte. As células a serem recuperadas estão na região ([10..20], [200..600], [1500..1510]), ou seja, estão contidas em uma região que obedece ao primeiro padrão de acesso. O total de células a serem recuperadas é de 40000, o que na prática implica fazer 5 acessos a disco, já que cada célula possui o tamanho de 1 byte e que a página de disco possui 8KB. Então, supondo que todos elementos de X3 caibam em uma página de disco, para recuperarmos os quatrocentos elementos de X2, teremos que ler 400 vezes os elementos de X3. Analogamente, para recuperarmos os dez elementos de X1, teremos que ler 10 vezes os elementos

de X3, o que nos dá um total de *4000* acessos a disco. Agora considere-se as células a serem recuperadas presentes na região ([50..70], [900..905], [7000..7400]), ou seja, contidas na região que obedece ao segundo padrão de acesso. O total de células a serem recuperadas também é de 40000. Então, para recuperarmos os cinco elementos de X2, teremos que ler 5 vezes os elementos de X3. Analogamente, para recuperarmos os vinte elementos de X1, teremos que ler 20 vezes os elementos de X3, o que nos dá um total de *100* acessos a disco.

Podemos concluir que precisaríamos de 10 acessos a disco para recuperar as células das regiões ([10..20], [200..600], [1500..1510]) e ([50..70], [900..905], [7000..7400]) e na prática estamos fazendo 4100 acessos. Significa dizer que fazemos 4090 acessos a mais para recuperar as células desejadas.

Suponha-se que o array seja dividido em chunks de 8KB. A forma desse chunk é um cubo (20, 20, 20), no qual cada célula do cubo tem o tamanho 1 byte. Podemos conferir o layout desse cubo através da figura 8(b).

Considere-se a recuperação de células do array (b) apresentado na figura 8. As células a serem recuperadas serão as mesmas que recuperamos para o array (a), ou seja, estão na região ([10..20], [200..600], [1500..1510]), que por sua vez, estão contidas em uma região que obedece ao primeiro padrão de acesso. O total de células a serem recuperadas é de 40000, o que na prática implica fazer 5 acessos a disco. Já que cada chunk possui 20 elementos de X1, 20 elementos de X2 e 20 elementos de X3, com um único acesso, recuperaríamos um chunk que conteria os 10 elementos que desejamos recuperar de X1 e os 10 elementos de X3. Resta-nos agora recuperar os 380 elementos restantes de X2. Para isso, precisaríamos de mais 19 acessos, já que a cada acesso recuperaríamos um chunk com 20 elementos de X2. Considere-se novamente as mesmas células recuperadas na segunda consulta para o array (a), ou seja, presentes na região ([50..70], [900..905], [7000..7400]), que por sua vez, estão contidas na região que obedece ao segundo padrão de acesso. O total de células a serem recuperadas também é de 40000. Em um único acesso recuperaríamos um chunk que conteria os 20 elementos que desejamos recuperar de X1 e os 5 elementos de X2. Resta-nos agora recuperar os 380

elementos restantes de X3. Para isso, precisaríamos de mais 19 acessos, já que a cada acesso recuperaríamos um chunk com 20 elementos de X3.

Podemos concluir que para recuperar a mesma quantidade de células, nosso array dividido em chunk, fez 40 acessos a disco contra os 4100 do nosso array sem a divisão de chunk. Isto nos mostra que, ao utilizarmos a técnica de chunk, obtivemos um fator de redução da ordem de 100 vezes em relação ao número de blocos recuperados, comprovando que o desempenho das consultas ao nosso array dividido em chunk, se torna ortogonal à sua navegação.

Na seqüência, apresentamos a definição formal de chunk [8]. Dado um array *n*-dimensional [X_1, X_2, ..., X_n] no qual X_i é o intervalo de uma determinada dimensão do array, o tamanho C da página do disco e os padrões de acesso $\{(P_i, I_{i1}, I_{i2},..., I_{in}): 1 \leq i \leq K\}$, o objetivo é determinar o chunk no qual o array será decomposto, de tal forma que o número de blocos recuperados seja minimizado. A forma do chunk é especificada por uma tupla (c_1, c_2, ..., c_n), onde c_i é o tamanho de uma determinada aresta do chunk multidimensional . O tamanho do chunk deve obedecer as seguintes restrições:

$$\prod_{i=1}^{n} c_i \leq C \qquad \text{(a)}$$

A fórmula (a) impõe a seguinte restrição ao tamanho do chunk: em um chunk (ele mesmo sendo um array *n*-dimensional), cada dimensão tem um intervalo de valores - o produto dos intervalos e do tamanho da célula tem que ser ≤ 8KB, para que o chunk caiba numa página de disco.

$$c_1 \leq X_1, c_2 \leq X_2, ..., c_n \leq X_n \qquad \text{(b)}$$

A restrição (b) impõe que cada aresta do chunk seja menor que a aresta do cubo que ele representa, ou seja, para um cubo cuja aresta x possui 10 elementos, não se pode formar um chunk que representa este array cuja aresta x possua mais de 10 elementos.

Várias técnicas são utilizadas para melhor escolher a forma de um chunk. Em [8] podemos encontrar algumas destas técnicas. Independente da escolha da fórmula, o importante é que a forma gerada para o chunk deve obedecer às restrições das fórmulas (a) e (b).

Neste trabalho, foi utilizada a técnica, denominada *máxima*, de gerar a forma do chunk. Uma forma é *máxima* quando ao incrementar qualquer dos eixos do chunk, continua-se obedecendo às restrições das fórmulas (a) e (b). Por exemplo: se $C = 15$ e $n = 2$, então a forma [5X3] é máxima, enquanto [4X3] e [5X2] não são. Optamos por utilizar esta técnica por entender que ela é mais precisa que as demais sugeridas em [8].

O número médio de blocos recuperados para um específico padrão de acesso e uma forma de chunk é dado por:

$$\sum_{i=1}^{K}\left(\prod_{j=1}^{n}\left\lceil\frac{A_{ij}}{c_j}\right\rceil\right)P_i \qquad (1)$$

onde:

K é o número de classes de probabilidades de acesso ao array definidas pelo padrão de acesso;

n é o número de dimensões;

A é o tamanho do eixo do array;

c é o tamanho do lado do chunk;

e P é a probabilidade de cada padrão de acesso.

O objetivo é escolher a melhor forma para o chunk, que satisfaça as restrições (a) e (b) e minimize (1). Através da fórmula (1) chegaremos a uma forma para o chunk que, na média, atende todos os padrões de acesso, privilegiando os padrões de acesso com maior probabilidade. Isto é bom, pois iremos dispor fisicamente os dados de maneira a melhorar o desempenho do tempo de resposta para os tipos de consultas mais significativos para o negócio. Os resultados apresentados no capítulo 4 comprovam esta afirmação.

No capítulo 3 descrevemos nossa solução de como armazenar chunks no Oracle 8.i.

2.5.3 Reordenamento dos lados do Chunk

A questão que discutimos nessa seção é: qual o melhor layout para um chunk [X_1, X_2, ..., X_n], no qual o objetivo é minimizar o tempo de resposta das consultas realizadas neste chunk? Para responder a esta questão analisamos a técnica de reordenamento. Reordenar os lados de um array significa fazer uma permutação entre suas dimensões. Diante da permutação, é possível identificarmos alguns parâmetros que melhor definem o layout de um chunk

Uma vez que o array foi decomposto em chunks é necessária a utilização de um bom método para dispor as informações do chunk no disco. A maneira natural seria dispor as informações na ordem natural dos eixos do array, ou seja, [X_1, X_2, ..., X_n]. Contudo, diferentes ordens dos eixos resultarão em diferentes maneiras de dispor as informações no chunk. O tempo para recuperar blocos em disco pode ser diminuído, a depender da escolha certa da ordem dos eixos.

Toda a análise feita para aplicar a técnica de reordenamento está intrinsecamente ligada ao padrão de acesso definido. A partir daí, é possível dispor as informações de forma contígua, visando aproximá-las fisicamente, segundo um determinado critério de consulta.

Em [8] são apresentados dois lemas:

Lema 1: O número de trilhas para acesso em disco a partir de uma consulta submetida (y_1, y_2, ..., y_n) é pelo menos,

$$\frac{(z_1 - 1)(d_2 d_3 ... d_n) + ... + (z_{n-1} - 1)d_n + z_n}{B} \qquad (2)$$

onde: $z_i = \left\lceil \frac{y_i}{c_i} \right\rceil$, $d_i = \frac{X_i}{c_i}$ (assumindo-se que c_i divide X_i exatamente) e B é o número de blocos por cilindro em disco.

O objetivo do lema 1 é identificar o número de trilhas recuperadas em uma consulta, para que a partir daí possam ser analisadas diferentes maneiras de dispor os eixos, de forma a minimizar a quantidade de trilhas recuperadas.

Lema 2: Dado um padrão de acesso, o valor da expressão (2), tendo sua média calculada sobre todos os elementos deste padrão, é minimizado para a ordem 1, 2, ..., *n* (sendo *n* o eixo mais interno) se:

$$\frac{a_1 - 1}{d_1 - 1} \leq \frac{a_2 - 2}{d_2 - 2} \leq \ldots \leq \frac{a_n - 1}{d_n - 1}, \qquad d_i \neq 1$$

onde: $a_j = \sum_{i=1}^{K} A'_{ij} \, P_i, \;\; A'_{ij} = \left\lceil \frac{A_{ij}}{c_j} \right\rceil \;\; e \;\; d_i = \frac{X_i}{c_i}.$

O objetivo do lema 2 é calcular a probabilidade das consultas acontecerem em cada lado do array, para assim, podermos identificar qual é a melhor maneira de ordenarmos os lados.

Para ilustrar a vantagem de reordenarmos os eixos do array, considere-se o exemplo apresentado na seção 2.5.2-figura 8. Suponha-se que o número de blocos recuperados por cilindro seja 60 (sessenta) [8]. Então, para o padrão de acesso assumido na seção 2.5.2-figura 8, usando o lema 1, o número de trilhas recuperadas para a ordem array [X_1, X_2, X_3] é 67. Usando-se o lema 2, se reordenarmos a ordem do array para [X_1, X_3, X_2], o número de trilhas recuperadas é 17 [8].

2.5.4 Armazenamento eficiente para o Array

Vimos as vantagens de utilizarmos as técnicas de chunking e reordenamento para recuperação de células de um array. Agora precisamos saber qual é o custo de armazenarmos nossos chunks (que são arrays) em tabelas, já que utilizamos um SGBD-OR no topo da tecnologia relacional. Nesta seção discutimos o custo-benefício de armazenamento entre tabelas *versus* array.

Arrays multidimensionais são claramente eficientes para armazenar dados densos. Por exemplo: considere-se um array n-dimensional com densidade de 100%. Para armazenar os dados descritos anteriormente em uma tabela relacional, cada tupla conteria uma chave estrangeira para cada dimensão D_i, e *p* atributos para um conjunto de medidas M, onde *p* corresponde ao número de atributos que identifica um fato e $M = \{m_1,...,m_p\}$, representando um conjunto de *p* medidas. Por outro lado, em um array, somente as medidas *p*

correspondentes são armazenadas, já que a célula no array é identificada pelos atributos-chave $D_i(A_{i1})$ para $1 \leq i \leq n$. Se T_s e A_s são requisitos para armazenamento em uma tabela e em um array respectivamente, então segundo [4] temos $T_S = \frac{n+p}{p} \times \rho A_S$, (4), onde ρ representa a densidade de dados. Isto significa que o esquema relacional, utilizando estrutura estrela, requer $\frac{n+p}{p}$ vezes mais recursos de armazenamento do que um array multidimensional precisa, quando a densidade dos dados é 100%. Podemos concluir isto diretamente através da interpretação da fórmula (4). Entretanto, esta eficiência de armazenamento acarreta um custo muito alto, que são os dados esparsos (espaços vazios entre as células), quando $\rho < \frac{p}{n+p}$, os requisitos de armazenamento para uma tabela são menores do que um array. Analisando-se a fórmula (4):

- já que a densidade do array é de 100%, ou seja, não possui esparcidade, considere-se um conjunto de medidas M = {faturamento} (p = 1). Considere também que o array possua 3 dimensões (n = 3). Então usando esses valores temos que: $\rho < \frac{1}{3+1}, \Rightarrow \rho < \frac{1}{4} \Rightarrow \rho < 25\%.$

Com isso, podemos concluir que: armazenar um DW com 3 dimensões e uma tabela de fatos com 1 atributo de medida em uma tabela é melhor quando $\rho < 25\%$ [4], pois um array requer potencialmente mais espaço em disco do que uma tabela fato. Entretanto, segundo [4], se tivermos $\rho \geq 25\%$ é melhor armazenarmos em um array.

Felizmente, arrays esparsos são extremamente compressíveis. Para resolver o problema de dados esparsos, existem técnicas de compressão bastante eficientes, como descrevemos na próxima seção.

2.5.5 Compressão Chunk-offset

Uma técnica muito recente usada para compressão de dados em arrays esparsos é a chamada compressão "chunk-offset". Nesta técnica, as células inválidas (células vazias) não são armazenadas. Para cada célula válida (célula com conteúdo que atende a algum critério de interseção entre dimensões), é gravado um par (offsetInChunk, dado). O offsetInChunk é um inteiro que é

computado da seguinte forma: considere-se o chunk como um array normal (não-comprimido). Cada célula *c* deste chunk é definida por um conjunto de índices. Por exemplo, se estivermos trabalhando com um chunk tri-dimensional, de *c* unidades em cada dimensão, um dado endereço é representado pela coordenada (i, j, k) no chunk. Para acessar a célula na memória, convertemos a coordenada (i, j, k) em um deslocamento s = $(i-1)(Y*Z)+(j-1)Z+k$ onde Y e Z representam a ordem das respectivas dimensões como definido em [29]. Este deslocamento é um inteiro "offsetInChunk" que é armazenado.

O array de células válidas é ordenado em forma crescente de acordo com as células do array de chunk offsets. Para um dado conjunto de índices no array (a_1, ... , a_n), nós podemos calcular o número de deslocamento e usar um método de acesso a este array para encontrar uma célula em uma dada posição, se existir conteúdo válido para esta célula na posição(a_1, ... , a_n).

A figura 9 mostra um array bidimensional com esparsidade, representado pelas dimensões *tempo* e *produto*.

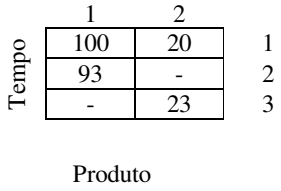

Figura 9: Array bidimensional esparso

A figura 10 mostra o array apresentado na figura 9, linearizado (não comprimido) antes e depois da compressão ChunkOffSet.

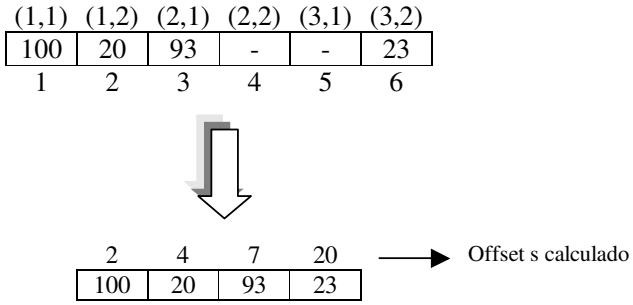

Figura 10: Array antes e depois de ser comprimido com chunkOffSet

2.5.6. Um algoritmo de consolidação para o cubo

Deve-se relembrar que uma consolidação básica em SQL é desta forma:

```
Select cidade, tipo, sum(volume)
From sales, produto, loja
Where sales.pid = produto.pid and sales.sid = loja.sid
Group by loja.cidade, produto.tipo
```

Esta operação de consolidação consiste de um *star join* seguido por um *group by* e uma agregação. No contexto da nossa implementação para o cubo de dados, uma operação de consolidação consistirá de um "*join"* do array dividido em chunks com suas tabelas dimensões associadas, seguidas por um group by e uma agregação.

Segundo [4], apresentamos o algoritmo de consolidação do nosso cubo de dados:

```
Para cada tabela dimensão participante do join faça
{
  use as árvores-b para originar uma lista de índices dos valores a serem selecionados;
  mescle as listas de índices, gerando uma lista final
}
Gere o produto cartesiano de todas listas finais;
Para cada elemento do produto-cartesiano faça
{
        Calcule o número do chunk ;
        Encontre o offset no chunk;
        Se o elemento é valido
            Então some o valor encontrado no chunk à célula do array resultante;
}
```

Capítulo 3

Projeto e Implementação de um Protótipo Molap

Este capítulo descreve o projeto e implementação de um protótipo MOLAP, cujo desenvolvimento considerou as técnicas para implementação de um esquema MOLAP apresentadas no capítulo 2. Como processo de desenvolvimento, foi utilizado o "Rational Unified Process" (RUP), descrito mais detalhadamente no apêndice A. Serão apresentadas as iterações utilizadas durante o desenvolvimento, bem como os artefatos gerados (especificação de requisitos, projeto arquitetural, diagramas de casos de uso, atividades, classes, etc.) durante as fases de concepção, elaboração e construção.

Para concepção do protótipo, foram utilizadas técnicas que possibilitassem a decomposição e persistência de um array *n*-dimensional em chunks, apresentadas no capítulo 2. Para verificar a pertinência da construção deste protótipo, foi realizada uma comparação de desempenho de consultas OLAP, submetidas a um esquema MOLAP projetado sobre o protótipo e a um esquema ROLAP projetado no próprio SGBD, utilizando a técnica de esquema em estrela.

O protótipo foi desenvolvido de forma iterativa e incremental como sugerido em [20], de maneira que, a cada ciclo de desenvolvimento (iteração), um novo conjunto de "problemas" era analisado para que ao final da iteração o protótipo incorporasse novas funcionalidades (incremento).

Outra técnica bastante utilizada durante o desenvolvimento do protótipo foi o refatoramento. Nesta técnica, nenhuma funcionalidade nova é incorporada ao que já foi desenvolvido. O que normalmente se faz é uma modificação da implementação, que não altera seu comportamento funcional, mas que melhora alguma qualidade não-funcional. Por exemplo: mudanças do nome de variáveis, mudança na interface dos objetos, pequenas mudanças arquiteturais, encapsular código repetido em um novo método, generalização de métodos, etc. Para isso, a utilização de padrões de projeto é amplamente recomendada, promovendo-se uma alta coesão e baixo acoplamento do

software. O resultado desta técnica é claramente percebido por facilitar a manutenção e a incorporação de novas funcionalidades em iterações futuras [16].

Para assegurar a qualidade final do protótipo, foi utilizada a técnica de testes de unidade. O que se busca ao utilizar esta técnica é diminuir os defeitos apresentados pelo produto ao longo do desenvolvimento. É importante que se façam testes de unidade quando se utiliza o refatoramento, assim, é possível observar se as modificações promovidas afetaram as funcionalidades já existentes [17]. Testes de unidade normalmente são classes com métodos que exercem funcionalidades que atendem a determinados requisitos, implementados em outras classes. Neste trabalho, foi utilizada a ferramenta de teste Junit, que permite a criação e execução de testes de unidade (neste caso, classes em Java).

A seção 3.1 descreve o planejamento das iterações que foram desenvolvidas no protótipo. Em seguida, a seção 3.2 descreverá o projeto do protótipo no qual são apresentados os artefatos do RUP gerados nas fases de inspeção, elaboração e construção e a apresentação da fase de transição, que juntas propiciarão um completo entendimento de todo o desenvolvimento do protótipo. Por fim, é apresentado um estudo de caso da realização de uma consulta OLAP no Protótipo.

3.1 Planejamento das iterações

Para o desenvolvimento do protótipo, foram planejadas duas iterações. Não foram necessárias mais iterações devido ao escopo do sistema. Todas as funcionalidades esperadas foram contempladas dentro dessas duas iterações.

Na primeira iteração, foram desenvolvidas todas as funcionalidades para se realizar uma consulta no protótipo. Nesta primeira iteração, não foi implementada a funcionalidade de utilização dos conceitos de chunk.

Com o desenvolvimento da primeira iteração, foram tecidas várias considerações que tiveram um impacto decisivo no direcionamento do trabalho.

- O protótipo está apto a realizar uma operação de consolidação? Sim. Foi implementado o algoritmo apresentado na seção 2.5.6. A principal consideração observada nesta iteração foi a disparidade apresentada na comparação do tempo de resposta das consultas submetidas ao esquema MOLAP, projetado no protótipo, e ao

esquema ROLAP, projetado no SGBD. Nesta iteração o SGBD possuía apenas a função de um repositório de dados para o protótipo, sendo que toda a lógica se encontrava na aplicação. Isto nos redirecionou na segunda iteração, na qual toda a lógica de recuperação dos fatos ficou implementada no SGBD.

- A arquitetura do protótipo foi validada? Sim. Além de nos redirecionar quanto às decisões de implementação, esta iteração serviu para validarmos a arquitetura do protótipo (detalhada a seguir na seção 3.2).

Na segunda e última iteração, foi adicionado o incremento de utilização dos conceitos de chunk. Com a adição deste novo incremento, foram observadas melhorias em relação à comparação do tempo de resposta das consultas. O capítulo 5 detalha como foram feitas essas comparações. Também foi adicionado nesta iteração o incremento de carga dinâmica de metadados a partir de arquivos XML.

3.2 Projeto do Protótipo MOLAP

Esta seção examina os artefatos gerados pelo protótipo MOLAP durante as fases de inspeção, elaboração e construção. Será detalhada a funcionalidade de cada uma das interfaces e classes inclusas no protótipo. A notação usada nos diagramas é a UML[18, 19] e os artefatos apresentados são extraídos do RUP[20] .

3.2.1 Artefatos da fase de inspeção

Os seguintes artefatos foram gerados durante a fase de inspeção:

a) Visão do sistema: Este artefato tem o propósito de capturar em um alto nível, os principais requisitos do sistema e identificar as principais restrições do mesmo. Ele serve como um contrato base e inicial entre a equipe técnica e o cliente, sendo capaz de prover um maior detalhamento técnico dos requisitos. Tipicamente, para pequenos projetos, é normal que o documento de visão do sistema possua poucos parágrafos[21].

Visão do Sistema

O objetivo do protótipo MOLAP é prover um mecanismo para que consultas OLAP possam ser realizadas utilizando a abordagem MOLAP em Data Warehouse, por meio de um SGBD-OR. Para isso, são utilizadas algumas técnicas apresentadas no capítulo 2, que são implementadas parte no SGBD e parte na aplicação que utiliza JAVA como linguagem de programação.

A interação com o protótipo deve ser realizada através de uma interface baseada em WEB, na qual o usuário informa a consulta OLAP que ele deseja realizar e o protótipo recupera as informações, atendendo ao critério da consulta informada.

b) Lista dos Principais Riscos: Este artefato tem o propósito de identificar os riscos percebidos que possam vir influenciar no sucesso do projeto. Ele serve para manter-se o foco nos principais riscos, para que possam ser "atacados" o mais cedo possível, com o objetivo de minimizar as chances de o projeto fracassar.

Lista dos Principais Riscos

1) Pouco conhecimento das técnicas de decomposição de um array n-dimensional em chunks;
2) Pouco conhecimento do SGBD utilizado;
3) Pouco conhecimento na integração das tecnologias JAVA (utilizada como linguagem de programação) e XML (utilizada para gerar o arquivo de configuração do protótipo)

c) Especificação dos Requisitos: Este artefato tem o propósito de capturar e organizar os principais Requisitos Funcionais do sistema. A partir dele é gerado o modelo de casos de uso. Ele serve também para determinar a correta orientação e organização de informações, fundamentais para o planejamento da gerência de requisitos. Estes requisitos foram obtidos através de reuniões junto ao professor Dr. Marcus Sampaio, orientador desta dissertação, e também da análise das técnicas apresentadas no capítulo 2.

Especificação dos Requisitos

RF1 – Deve ser projetado o esquema conceitual do DW.

O protótipo deve possuir um esquema conceitual sobre o qual serão realizadas as consultas gerenciais. Este esquema define as dimensões com seus respectivos atributos, bem como os fatos que estão relacionados às dimensões.

O esquema conceitual deve utilizar a abordagem MOLAP, utilizando a técnica de chunk apresentada no capítulo 2. Pautando-nos nesta abordagem deve ser definido um cubo que possua as dimensões Produto, Loja e Tempo, descrevendo as datas, quantidades e valores de venda dos produtos realizadas nas lojas.

A dimensão Produto deve possuir o nome do produto, sua categoria e sua família. A categoria define um nível hierárquico em relação ao produto, bem como a família define um nível hierárquico em relação à categoria.

A dimensão loja deve possuir o nome da loja e o Estado Federativo no qual está situada. O estado define um nível de hierarquia em relação à loja.

A dimensão tempo deve possuir dia, mês, trimestre, semestre e ano. Cada um dos atributos citados define um nível de hierarquia em relação ao anterior, respectivamente.

O fato representado no cubo deve possuir a quantidade e o valor da venda.

O esquema conceitual projetado será descrito em um arquivo XML que contemplará os metadados do protótipo.

RF2 – Deve permitir que consultas sejam realizadas em granularidades maiores.

É bastante comum que um gerente de negócio, ao analisar as informações gerencias, tome a decisão de consultar estas informações

em níveis de hierarquia superiores para chegar às informações que realmente lhe são pertinentes.

Sob este aspecto, o protótipo deve prover um mecanismo de navegação sobre as informações para que este tipo de análise possa ser realizado.

RF3 – Devem ser definidos os padrões de acesso ao cubo, bem como suas respectivas probabilidades de ocorrerem.

Tomando-se por base as consultas realizadas com maior freqüência, devem ser definidos quais critérios de consultas serão mais eficientes. É importante atribuir pesos aos tipos de consulta de maior demanda. Isso favorecerá o tempo de resposta a consultas que atenderem a estes critérios e também influenciará diretamente a forma pela qual as informações serão dispostas em meio magnético.

RF4 – Devem ser implementados algoritmos de consultas sobre o array dividido em chunks.

Uma vez que as informações estão dispostas em um array dividido em chunks, faz-se necessária a implementação de algoritmos que permitam a realização de consultas OLAP.

As funcionalidades dos algoritmos foram definidas no capítulo 2. Basicamente, os algoritmos realizam a operação de consolidação sobre o array divido em chunks. Esses algoritmos realizam respectivamente uma consolidação sem seleção e outra com seleção.

RF5 – Devem ser criados índices de árvore-b que associem os elementos das tabelas dimensões com suas respectivas abscissas no cubo.

Os índices de árvore-b mapeiam os elementos das tabelas-dimensões com suas respectivas abscissas no cubo. Isto significa dizer que um produto A armazenado em uma tabela produto corresponde à abscissa 1 do eixo de produtos no cubo. Logo, o protótipo deve prover um mecanismo para que este mapeamento seja feito.

d) Especificações Suplementares: Este artefato tem o propósito de capturar os requisitos não-funcionais do sistema.

Especificações Suplementares

ES1 – Deve ser configurado o SGBD a ser utilizado

O protótipo deverá utilizar um SGBD na qual será projetado o esquema físico do Datawarehouse. Este SGBD deve prover uma interface que possibilite a implementação de uma estrutura MOLAP.

Deve ser configurada a forma de conexão ao SGBD.

ES2 – Deve ser definida uma forma para o chunk, bem como sua implementação.

Já que o SGBD utilizado não é propriamente multidimensional, é necessário construir um mecanismo de armazenar as informações para que elas sejam recuperadas e analisadas de forma multidimensional.

Como apresentado no capítulo 2, o protótipo deve contemplar a implementação da técnica de divisão de um array n-dimensional em pequenos arrays n-dimensionais, bem como sua persistência em um SGBD-OR, conforme requisitos levantados em *ES1*.

ES3 – Deve ser estabelecida uma forma de reordenamento das informações nos chunks.

Depois de ser definida a forma do chunk, faz-se necessário analisar as informações segundo a disposição das mesmas em suas respectivas dimensões para que os padrões de acesso levantados em *RF3* definam a melhor maneira de dispor as informações fisicamente objetivando sempre a maximização do tempo de resposta as consultas submetidas.

ES4 – Tipo de interface desejada.

Interface baseada em WEB, utilizando Java Server Pages (JSP), na qual são informados os elementos das dimensões que representam um determinado critério de consulta. Como o objetivo deste trabalho não é construir, nem tampouco rebuscar a GUI, uma interface simples e objetiva atenderá aos nossos propósitos. Esta GUI permitirá, entre outras vantagens, a fácil manutenção do sistema aliada ao seu baixo custo, visto que os usuários utilizarão um browser universal.

ES5 – Documentação disponibilizada.

Como documentação interna, será disponibilizado um arquivo gerado pela ferramenta Rational Rose, que permite que seja projetado um sistema utilizando a notação UML, capaz de contemplar o modelo de casos de uso, os diagramas de atividade e os diagramas de classes, entre outros. Também fará parte desta documentação um arquivo JAVADOC que disponibilizará comentários de todas as classes, métodos e atributos do protótipo.

Como documentação externa, esta dissertação possui uma seção voltada para o usuário final com as orientações de como proceder para realizar consultas utilizando o protótipo.

ES6 – Oferecer uma melhoria de desempenho de consultas OLAP utilizando a abordagem MOLAP com relação à abordagem ROLAP via um SGBD-OR.

O protótipo MOLAP deve oferecer melhoria de desempenho em consultas OLAP, via um SGBD-OR de ampla aceitação no mercado.

Utilizando-se o mesmo ambiente computacional, dois esquemas devem ser projetados para realizar consultas OLAP. Um esquema utilizará a abordagem ROLAP na qual é empregada a técnica de *esquema em estrela* e a outra abordagem utilizará uma estrutura multidimensional (MOLAP). Através do comparativo de desempenho entre esses dois esquemas, espera-se que o esquema MOLAP contemplado pelo protótipo obtenha tempos de resposta inferiores.

e) Atores: Um ator define o conjunto de papéis que os usuários podem assumir quando interagem com o sistema. Um ator pode ser um usuário individual ou um sistema externo. Ao encontrar um ator, significa que você está delimitando as fronteiras do seu sistema, pois isto ajuda no entendimento do propósito e estensão do sistema.

Atores

ProjetistaDW: Papel que representa o projetista do Data Warehouse. Este ator é quem vai projetar os esquemas, conceitual e físico, do Data Warehouse, criar o cubo, definir a forma do chunk, definir os padrões de acesso e criar os índices que associam os elementos das dimensões com suas respectivas abscissas no cubo.

UsuárioDW: Papel que representa o usuário que realiza consultas OLAP no cubo.

f) Modelo de casos de uso: O Modelo de Casos de uso é um modelo das funcionalidades desejadas para o sistema, servindo como um contrato entre o usuário e a equipe de desenvolvimento sobre o que será implementado pelo sistema. Este artefato é extraído da especificação de requisitos e serve como entrada para as disciplinas de análise/projeto e testes.

Existem diferentes maneiras de se modelar as funcionalidades de um sistema. O importante aqui, é que, o modelo de casos de uso deve transmitir o comportamento do sistema para o cliente. Para tanto, o modelo deve ser de fácil entendimento para os mesmos.

O Modelo de Casos de Uso é composto por dois artefatos: o diagrama de casos de uso e o detalhamento de casos de uso. O diagrama de casos de uso consiste em um modelo visual dos requisitos do sistema agrupados por um critério de funcionalidade. O detalhamento de casos de uso é uma descrição textual das funcionalidades do caso de uso. Nele é descrito todo o fluxo de eventos executados pelo caso de uso. Um modelo de casos de uso pode conter vários casos de uso, cada um com o seu respectivo detalhamento.

Modelo de Casos de Uso

As figuras 11 e 12 ilustram os diagramas de casos de uso segundo as interações dos atores com os casos de uso. São as seguintes interações:

CasosUso - ProjetistaDW: O ProjetistaDW interage com os casos de uso que definem toda a concepção do protótipo.

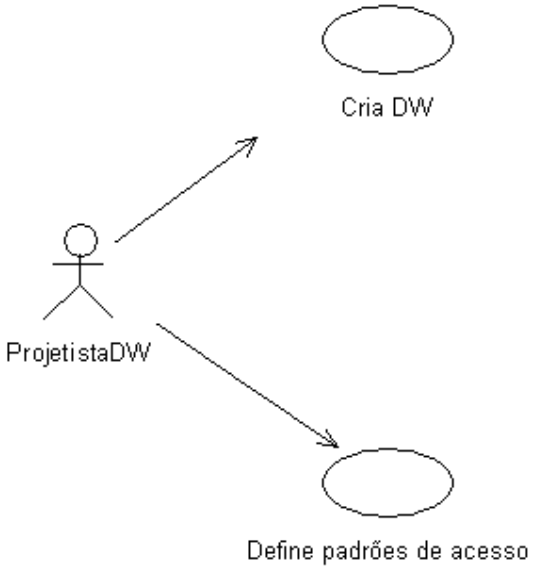

Figura 11: Caso de Uso segundo ator ProjetistaDW

CasosUso - UsuárioDW: O UsuárioDW interage com o caso de uso que define as consultas OLAP, executadas no protótipo e discutidas e analisadas neste trabalho.

Figura 12: Caso de Uso segundo ator UsuarioDW

Além dos diagramas dos casos de uso mostrados acima, foi elaborado o detalhamento para cada caso de uso do protótipo. O fluxo básico descreve os eventos ocorridos em um cenário ideal, ou seja, sem que ocorra qualquer tipo de erro ou exceção na regra de negócio. Os fluxos alternativos definem um comportamento alternativo ao fluxo básico segundo [20].

CU1 – Cria DW

1. Introdução

Este caso de uso permite ao Projetista DW criar um DW com elementos do tipo: DW, Cubo, Dimensão, Atributos-Dimensão. As informações para criação do DW são oriundas do arquivo de metadados. Os metadados em um sistema podem ser entendidos como o conjunto de informações sobre a estrutura e significado dos dados, das aplicações e dos processos que manipulam os dados. O propósito do arquivo de metadados do protótipo é descrever o esquema conceitual do DW projetado.

2. Breve Descrição

Como o SGBD utilizado não possui um tipo primitivo que represente um array n-dimensional, então o DW é representado por uma tabela. A idéia é, através da utilização de uma tabela, simular o comportamento de um array n-dimensional. Isto nada mais é que fazer uma representação linear (registros da tabela) do array n-dimensional. Uma maneira de se fazer isso é utilizando o conceito de chunks. Para isto, deve ser criada nesta tabela uma coluna que identifique o número do chunk. O número do chunk é calculado e pode ser recuperado através de um comando SQL. Depois de recuperado, o chunk torna-se a representação de um pequeno array n-dimensional em memória. Então, um chunk contém uma quantidade de registros recuperados na tabela através da coluna que contém o seu número. É necessária uma maneira de acessar os registros em memória. Para acessar um elemento do chunk, a tabela também deve possuir uma coluna chamada offSetInChunk, gerada a partir do cálculo de uma coordenada (x,y,z), obtida a partir dos elementos das dimensões e das árvores-b associadas a estas dimensões, gerando um valor único que possibilita o acesso a esta coluna como foi descrito na seção 2.5.5.

3. Ator(es)

- ProjetistaDW

4. Pré-Condições

A pré-condição para que este caso de uso seja realizado é que o arquivo com o esquema conceitual do DW tenha sido criado.

5. Pós-Condições

Não se aplica a este caso de uso.

6. Fluxo de Eventos

6.1. Fluxo Básico

Este caso de uso inicia quando o ProjetistaDW escolhe a opção "Criar DW".

6.1.1. O sistema cria o dw, o cubo, as dimensões (produto, loja e tempo), os atributos das dimensões (Produto[nome, categoria, familia], Loja[nome, região], Tempo[dia, mês, trimestre, semestre e ano]) e a conexão com o banco;

6.1.2. O sistema estabelece a conexão com o banco através dos parâmetros (login, senha, ip, porta, sid) segundo informações do metadado;

6.1.3. O sistema carrega o dw, o cubo, as dimensões e os atributos das dimensões segundo informações do metadado;

6.1.4. O sistema associa os elementos das dimensões com suas respectivas abscissas no cubo através das árvores-b;

6.1.5. O caso de uso é finalizado.

6.2. Fluxo Alternativo

6.2.1 Erro Geral

Caso ocorra qualquer erro neste caso de uso, uma exceção é lançada e o caso de uso é finalizado.

6.2.2 Arquivo de metadados inexistente

No passo 6.1.2 do fluxo básico, caso não exista o arquivo de metadados, então o sistema exibirá uma mensagem de erro e o caso de uso será finalizado.

6.2.3 Erro na conexão

No passo 6.1.2 do fluxo básico, caso ocorra erro durante a tentativa de conexão ao banco, então exibirá uma mensagem de erro e o caso de uso será finalizado.

CU2 – Define padrões de acesso

1. Introdução

Este caso de uso permite que se estabeleçam os padrões de acesso para se consultar o cubo. Como descrito no capítulo 2.5.1, um padrão de acesso pode ser definido tanto por estatísticas de acesso às informações no cubo quanto pelo próprio gerente de negócio. No caso do protótipo, os padrões foram definidos segundo uma tendência adotada por empresas que possuem DW com a mesma temática adotada neste trabalho.

2. Breve Descrição

Foram estabelecidos quatro padrões de acesso, segundo os tipos de consulta que serão realizadas no protótipo. Os tipos de consulta são os seguintes:

- faturamento de todas as lojas por mês com probabilidade de 20% de ocorrer;
- faturamento de todas as lojas por trimestre com probabilidade de 10% de ocorrer;
- faturamento de todas as lojas por semestre com probabilidade de 30% de ocorrer;
- faturamento de todas as lojas por ano com probabilidade de 40% de ocorrer;

3. Ator(es)

- ProjetistaDW

4. Pré-Condições

Não se aplica a este caso de uso.

5. Pós-Condições

5.1. Definir a forma do chunk

Esta pós-condição permite que se defina a melhor forma para o chunk. Uma vez definidos os padrões de acesso, o sitema deve estabelecer a melhor forma que o chunk deve possuir. O tamanho do chunk é igual a 8KB, ou seja, do mesmo tamanho de uma página de memória, assim configurada para este trabalho. Isso se deve ao fato de se buscar uma minimização do número de vezes em que é feita a operação de I/O em disco como descrito na seção 2.5.2.

5.2. Reordena dados dos chunks

Esta pós-condição permite que se faça um reordenamento dos dados dos chunks. Como descrito na seção 2.5.3, os dados podem ser ordenados para otimizarem sua recuperação segundo os padrões de acesso definidos. Neste trabalho, os dados dos chunks devem ser reordenados segundo os padrões de acesso definidos no item 2 deste caso de uso.

6. Fluxo de Eventos

6.1. Fluxo Básico

Este caso de uso inicia quando o ProjetistaDW escolhe a opção "Definir padrões de acesso".

6.1.1. O usuário informa as regiões do cubo correspondentes aos tipos de consultas que definen os padrões de acesso apresentados no item 2 deste caso de uso;

6.1.2. O usuário informa a probabilidade de ocorrência de cada padrão de acesso;

6.1.3. O sistema calcula o número médio de blocos recuperados para uma determinada fórmula de chunk segundo a fórmula (1) apresentada na seção 2.5.2.;

6.1.4. O sistema calcula a melhor forma para o chunk definido no passo anterior através da fórmula apresentada no lemma 2 na seção 2.5.3;

6.1.5. O caso de uso é finalizado.

6.2. Fluxo Alternativo

6.2.1 Erro Geral

Caso ocorra qualquer erro neste caso de uso, uma exceção é lançada e o caso de uso é finalizado.

CU3 – Realiza Consultas

1. Introdução

Este caso de uso permite que sejam realizadas consultas gerenciais sobre os esquemas definidos pelo ProjetistaDW, utilizando o protótipo MOLAP.

2. Breve Descrição

Uma consulta é realizada quando o UsuárioDW informa ao protótipo quais são os elementos das dimensões que ele deseja consultar. Estes elementos devem impreterivelmente estar descritos nos metadados do protótipo. A partir daí o protótipo procederá a operação de consolidação dessas informações.

3. Ator(es)

- UsuárioDW

4. Pré-Condições

Não se aplica a este caso de uso.

5. Pós-Condições

Não se aplica a este caso de uso.

6. Fluxo de Eventos

6.1. Fluxo Básico

Este caso de uso inicia quando o UsuárioDW escolhe a opção "Realizar Consulta".

6.1.1. O usuário seleciona o(s) produto(s) a serem pesquisados;

6.1.2. O usuário seleciona o critério de agregação do(s) produto(s) selecionado(s) no passo anterior.

6.1.3. O usuário escolhe a(s) loja(s) a serem pesquisada(s);

6.1.4. O usuário seleciona o critério de agregação da(s) loja(s) selecionada(s) no passo anterior.

6.1.5. O usuário informa o intervalo de datas a serem pesquisadas;

6.1.6. O usuário escolhe a função de agregação;

6.1.7 O usuário submete a consulta ao sistema;

6.1.8. O sistema exibe o resultado da consulta submetida.

6.1.9. O caso de uso é finalizado.

6.2. Fluxo Alternativo

6.2.1 Erro Geral

Caso ocorra qualquer erro neste caso de uso, uma exceção é lançada e o caso de uso é finalizado.

3.2.2 Artefatos da fase de Elaboração

Os seguintes artefatos foram gerados durante a fase de elaboração:

a) Projeto Arquitetural : Este artefato provê uma compreensão sob um ponto de vista de mais alto nível, voltado para a arquitetura do sistema. Ele serve para orientar os desenvolvedores durante a fase de construção sobre quais aspectos relativos à arquitetura devem ser observados.

Documento de Arquitetura do Protótipo

O projeto arquitetural aqui apresentado vem ao encontro dos requisitos levantados anteriormente. O leitor terá à sua disposição diagramas que abordam diferentes perspectivas da arquitetura, a fim de otimizar a compreensão da arquitetura proposta, concentrando sua atenção nos aspectos mais relevantes do processo de decisões. A arquitetura do protótipo está divida em três camadas. A figura 13 ilustra a divisão das camadas do protótipo.

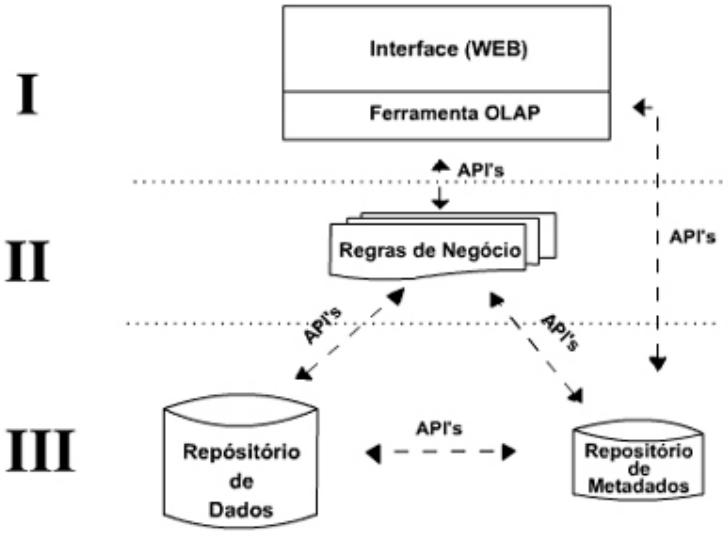

Figura 13: Arquitura em Camadas do Protótipo

A primeira camada - Camada de Apresentação (I) — compreende uma ferramenta OLAP de consulta a DWs apresentada em um browser, cujos esquemas conceituais são projetados no repositório de

metadados. Nela estão implementadas as funcionalidades com as quais o ator UsuárioDW com elas interage.

A segunda camada - Camada de Negócio (II) — trata da implementação de todas as regras de negócio do protótipo. Nela estão implementadas todas funcionalidades apresentadas nos casos de uso cujo ator ProjetistaDW interage.

A terceira camada -Camada de Persistência (III) — compreende as fontes de dados gerenciais e de metadados, que alimentam o DW do protótipo.

Por fim, e permeando as três camadas, temos o Repositório de Metadados. Nele são armazenadas e mantidas informações sobre as estruturas e os conteúdos das três camadas da arquitetura.

Do lado cliente, o UsuárioDW, através de uma interface WEB, comunica-se com o protótipo, que permite o início da execução do sistema. O browser foi a ferramenta escolhida para atender o requisito não-funcional *ES4*. Outra solução factível para a implementação, seria *stand alone*. Entretanto, com a programação WEB estamos não só utilizando o que há de mais novo entre os paradigmas de programação, bem como todas as vantagens que esta abordagem nos propicia. Outro fator relevante é o fato de obter-se um trabalho mais valorizado, visto a utilização de uma tecnologia de ponta.

Do lado servidor, o ProjetistaDW, localmente, através de um editor próprio de XML, especificará todo projeto do esquema conceitual do DW, atendendo assim, ao requisito funcional *RF1*. Realizada esta etapa, o arquivo será disponibilizado para que um *servlet* possa carregá-lo e interagir com o protótipo. Novamente, a possibilidade de se implementar essa solução *stand alone* é plausível, contudo temos um compromisso que está voltado para a qualidade no desenvolvimento de software, e mais uma vez optamos por prezá-la.

A tecnologia JSP foi escolhida com intuito de possibilitar a utilização de um browser. Outra alternativa para implementar uma aplicação com interface web seria a tecnologia *Active Server Page* (ASP), por

exemplo.Como nossa prioridade não é implementar uma GUI com muitos recursos visuais, não havendo assim a necessidade de uma interação com um projetista de interfaces visuais, optamos pela implementação das páginas JSP bastante simples. Através da JSP, o JConfig, biblioteca responsável por traduzir arquivos XML para JavaBeans [22], faz a leitura do arquivo XML no servidor, contendo as informações de conexão ao banco e dos metadados do DW, e retorna uma lista com estes objetos especificados para o sistema. Outra vez buscando a excelência na qualidade do nosso trabalho, estamos utilizando uma tecnologia que padroniza a formatação de documentos, seja ele um arquivo de configuração ou não. Uma vez carregadas as especificações do arquivo XML, o UsuárioDW está apto a submeter consultas ao protótipo. Então, ele informa quais são os elementos das dimensões que ele deseja consolidar e submete a consulta. Os dados da consulta são transportados na rede através do protocolo http. Uma vez recebidos, o protótipo interpreta esses dados e acessa o SGBD. O SGBD, então, processa a consulta e retorna o resultado da consolidação ao protótipo, que por sua vez retransmite através do http para viabilizar a exibição dos dados na página JSP de resposta. As figuras 14 e 15 ilustram a arquitetura proposta.

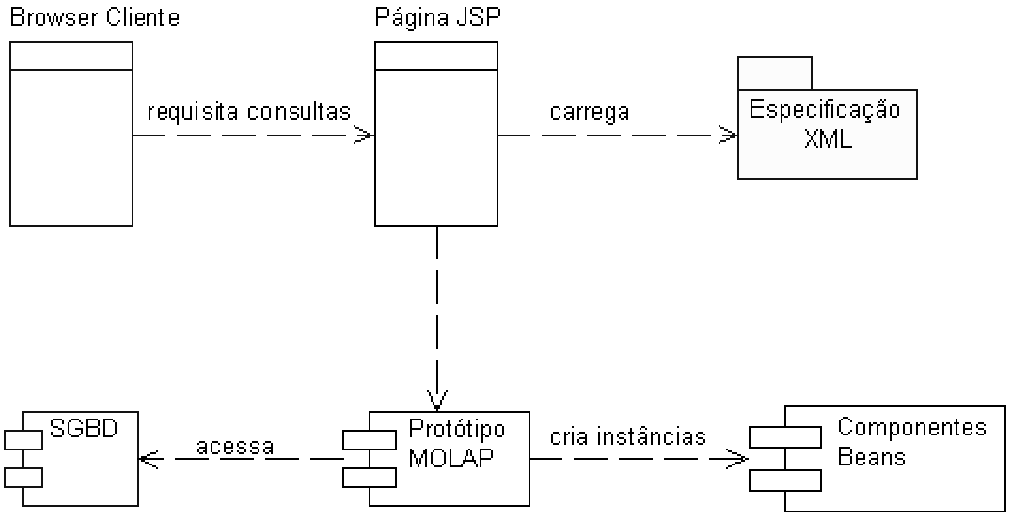

Figura 14: Arquitetura do Protótipo Molap (a)

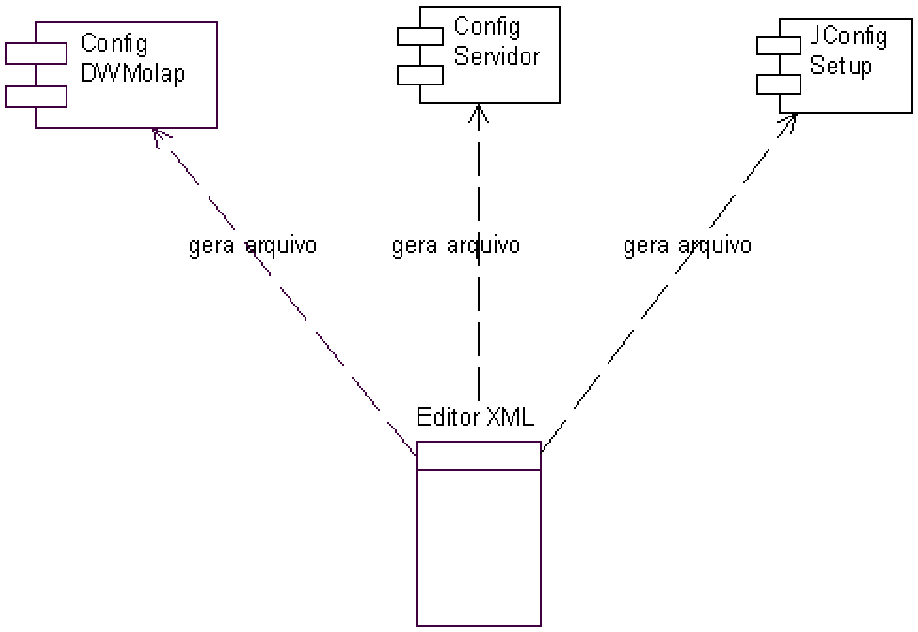

Figura 15: Arquitetura do Protótipo Molap (b)

Ainda do lado servidor, optamos em utilizar o SGBD Oracle 8.i, após fazermos uma análise dos recursos por ele oferecidos. O protótipo possuirá parte da "inteligência" na aplicação e parte no banco, outrora justificado no capítulo 5. Com isso, o protótipo passa a utilizar todos os recursos que o SBGB oferece, como: otimizador de consultas, os planos de "query", etc.

b) Modelo de Análise : O modelo de análise contém as classes levantadas durante a análise do protótipo. É um artefato temporário que auxilia na identificação das principais classes durante o desenvolvimento de um sistema. Freqüentemente, é comum que apareçam as classes persistentes e aquelas que estão diretamente relacionadas com o domínio do problema. O principal produto desse modelo é o diagrama de classes de análise.

Diagrama de Classes de Análise

Neste diagrama, as classes são usadas para representar as principais abstrações do sistema, ainda em um nível bastante conceitual. Ele é a principal referência para elaboração do diagrama de classes na fase de

projeto. A figura 16 ilustra as classes e associações do protótipo identificadas na análise.

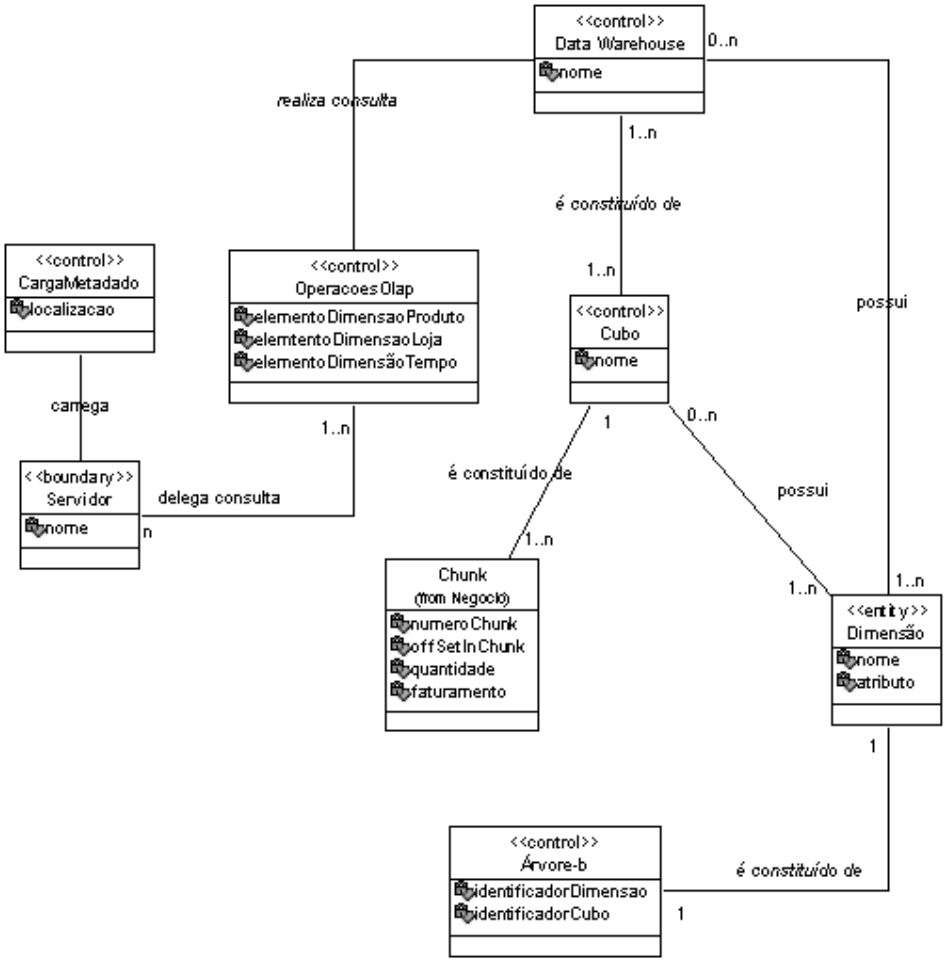

Figura 16: Diagrama de classes de Análise

As classes *DataWarehouse, Cubo e Dimensão* e suas associações representam de forma genérica o esquema conceitual do DW. A classe *Dimensão* está associada com a classe *Árvore-b*, que representa os índices que mapeiam os elementos das dimensões com suas respectivas abscissas no cubo. A classe *Cubo* possui uma associação com a classe *Chunk*, que é responsável por toda a funcionalidade de persistência e tratamento dos fatos, conforme descrito na seção 2.5.2. A classe *Servidor* colabora com a classe *OperacoesOlap*, pois esta atua na

resolução das consultas OLAP. A classe *Servidor* também colabora com a classe *CargaMetadado,* que faz a carga do repositório de metadados com as informações relativas aos esquemas do DW.

c) Modelo de Projeto : O modelo de projeto é um modelo de objetos que descreve a realização dos casos de uso e serve como uma abstração do código-fonte implementado. Este modelo é derivado do modelo de análise. É um artefato que auxilia na identificação das classes que não estão diretamente relacionadas com o domínio do problema. O principal produto desse modelo é o diagrama de classes de projeto.

Pacotes e Subsistemas do Protótipo

As classes e interfaces do protótipo foram agrupadas de forma a facilitar o entendimento e a manutenção do mesmo. Para isso, foram utilizados os conceitos de pacote (*package*) e subsistema (*subsystem*) [23], segundo a notação UML. Estes mecanismos permitem agrupar classes, relacionamentos, diagramas e até outros pacotes.

Pacotes e subsistemas são porções principais resultantes da subdivisão em limites de coesão lógica funcional. Se, grupos de classes relacionam-se entre si através de uma junção forte, ou ainda, se for possível dar um nome a um grupo de classes, isto pode indicar a existência de um bom agrupamento para um pacote ou subsistema [24]. A rigor, pacotes e subsistemas são utilizados para o mesmo propósito. A principal diferença entre eles é que um subsistema necessariamente deve implementar uma ou mais interfaces, portantanto os elementos que o constituem são acessados exclusivamente através de sua(s) interface(s). Já com o pacote, os elementos que o compõe são acessados diretamente.

Uma classe contida em um pacote pode ter sua visibilidade como pública ou privada. Uma classe pública pode estar associada com qualquer outra classe, ao passo que, uma classe privada pode estar associada apenas com classes que compõem o mesmo pacote.

A figura 17 ilustra o relacionamento entre os pacotes e os subsistemas que compõem o protótipo.

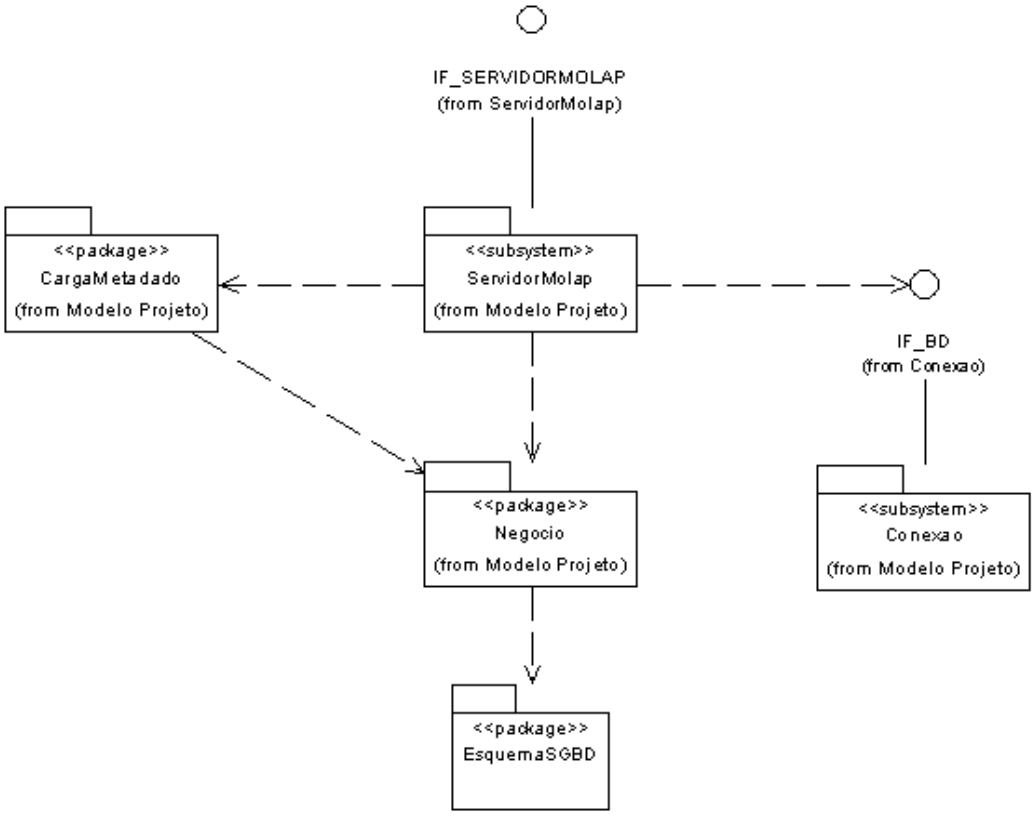

Figura 17: Pacotes e Subsistemas do Protótipo

Diagrama de Classes de Projeto

Neste diagrama, um conjunto de classes compartilham as suas responsabilidades, relacionamentos, operações, atributos e semântica. Freqüentemente, as classes de análise são mantidas neste diagrama. Elas acabam sendo "refinadas", ou seja, são identificados os métodos, atributos com caráter de projeto, relacionamentos mais especializados, etc. São identificadas também as classes com comportamento transiente, aquelas que não possuem comportamento persistente, que colaboram com as classes persistentes, para atender às funcionalidades levantadas.

Como ilustrado na figura 17, o protótipo foi decomposto em 3 (três) pacotes e 2 (dois) subsistemas. As figuras 18, 19, 20 e 21 ilustram as classes e interfaces que formam cada um dos pacotes e subsistemas. Essas classes e interfaces serão explicadas na seção seguinte.

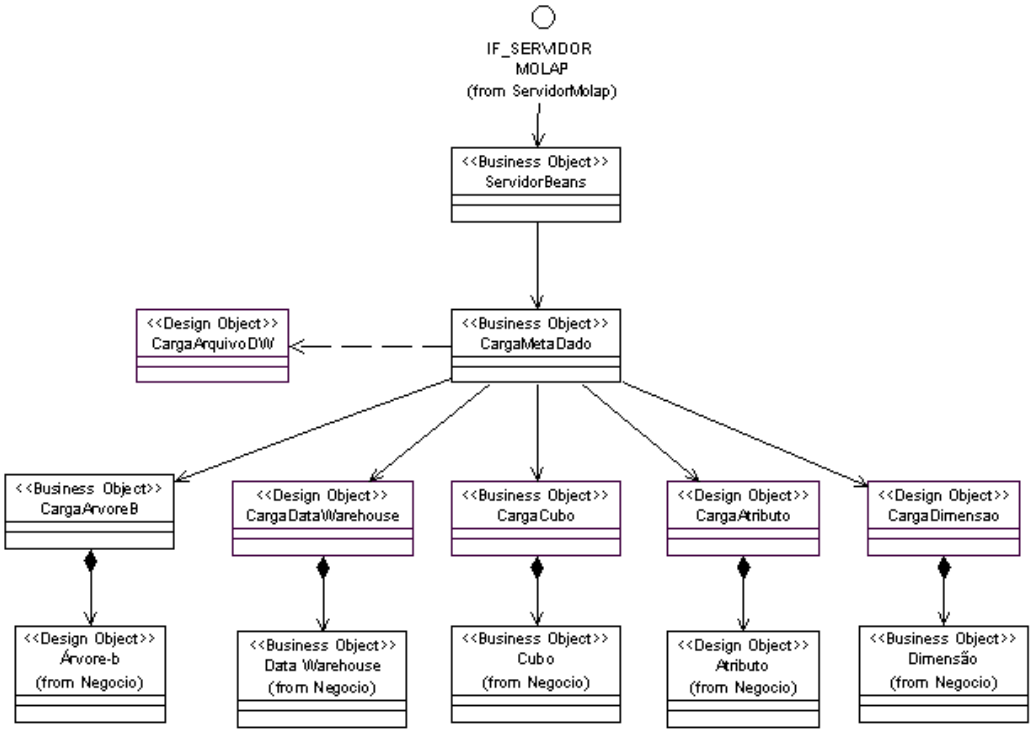

Figura 18: Pacote CargaMetadado

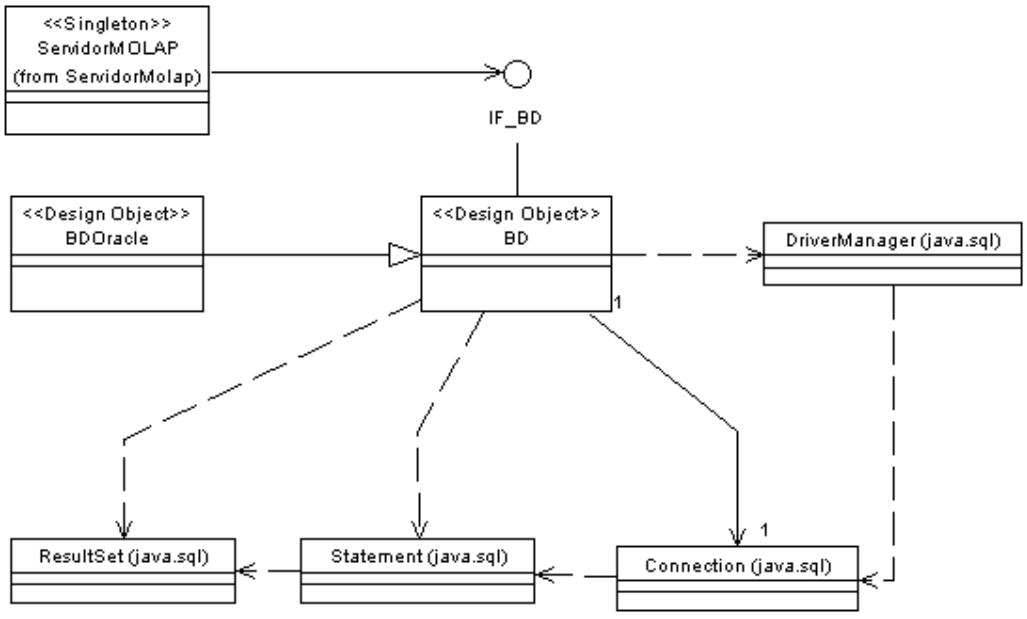

Figura 19: Subsistema Conexao

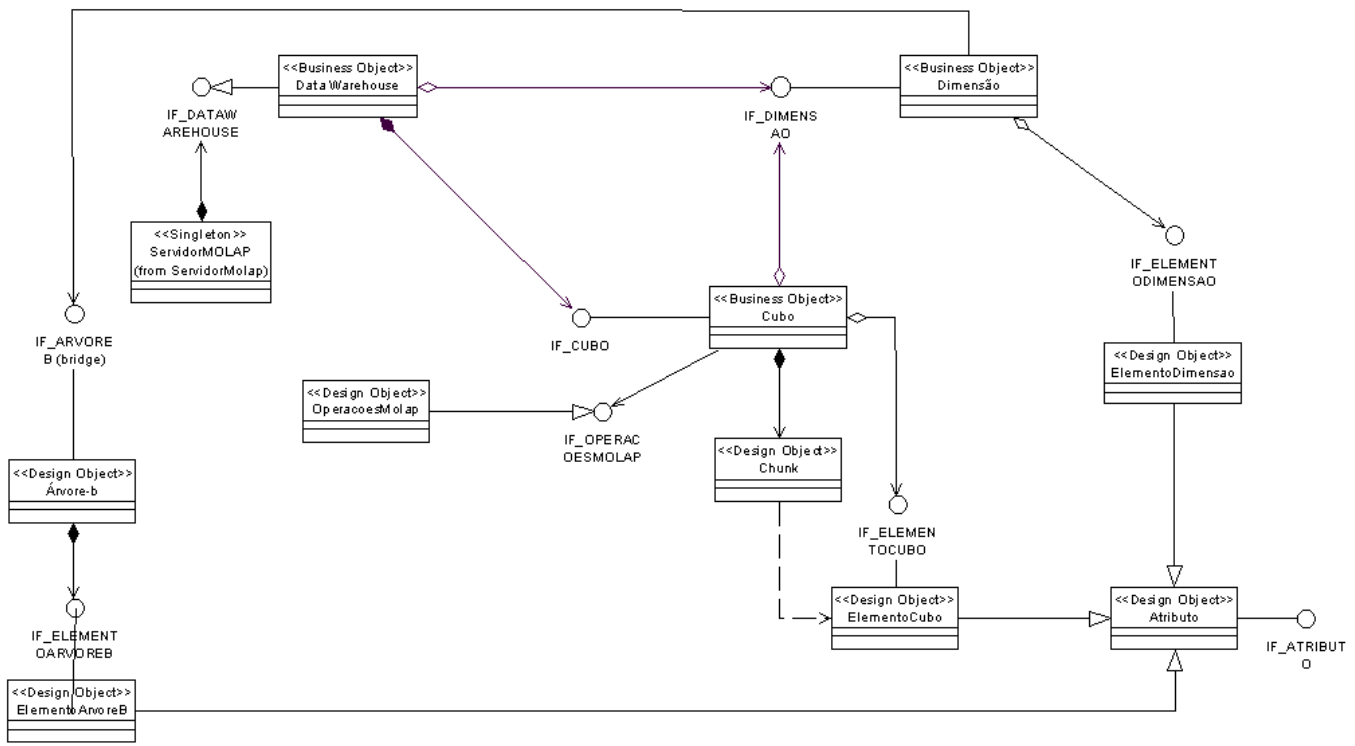

Figura 20: Pacote Negocio

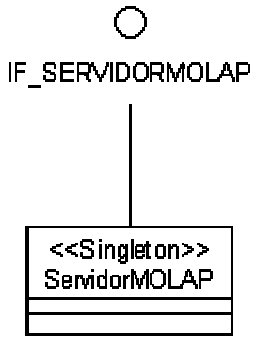

Figura 21: Subsistema ServidorMolap

3.2.3 Artefatos da fase de Construção

O objetivo da fase de construção é completar a implementação do sistema com os requisitos até então não considerados. As duas fases anteriores, inspeção e elaboração, se concentram basicamente em elicitar a maior parte da funcionalidade do sistema, bem como validar a arquitetura, na qual o código gerado está bastante focado nesta última. Percebe-se que, nesta fase, o esforço fica bastante concentrado na codificação. Mas durante a fase de construção também são gerenciados recursos, controle de operações, otimização de custos e qualidade. Os esforços de gerenciamento e de configuração também são presentes nesta fase. Como configuração e gerência não estão no escopo deste trabalho, são apresentadas as descrições das principais interfaces e alguns trechos de código cujos comentários são bastante pertinentes. O artefato desta fase, gerado neste trabalho, foi o modelo de implementação, descrito a seguir.

a) Modelo de implementação: Este artefato provê uma compreensão sob o ponto de vista da implementação. O modelo de implementação é uma coleção de componentes que implementam pacotes e subsistemas. O termo componentes é definido aqui como componentes de entrega, nos quais temos os executáveis, e componentes que geram esses executáveis, tais como classes, interfaces e componentes de software. Serão apresentadas as classes e

interfaces implementadas no protótipo, segundo as divisões nos pacotes e nos subsistemas.

Implementação do Subsistema ServidorMolap

A figura 21, ilustra o subsistema *ServidorMolap.*

A interface **IF_ServidorMolap** provê um comportamento para que a classe **ServidorMolap**, que a implementa, possua uma única instância (padrão de projeto *façade* + *singleton* [25]). Ela tem o objetivo de proporcionar uma interface com os demais pacotes e subsistemas. Serve como ponto único de acesso ao protótipo, fazendo a comunicação entre a camada de apresentação e a camada de negócio. O objeto do tipo *ServidorMolap* possui uma coleção de *DataWarehouse*, que representa o esquema conceitual do DW implementado no protótipo. Ele também possui um conjunto de métodos de criação de outros objetos (padrão de projeto *factory methods* [25]), que através da interface *IF_ServidorMolap*, definem uma forma de deixar subclasses decidirem qual classe instanciar. Percebe-se que desta maneira é eliminada a necessidade de colocar classes específicas da aplicação no código da classe *ServidorMolap*.

Implementação do Subsistema Conexão

Ilustrado na figura 19, o subsistema *Conexão* foi implementado atendendo ao seguinte requisito não-funcional:

- ES1 - Deve ser configurado o SGBD a ser utilizado

A interface **IF_BD** trata de todos aspectos referentes à utilização do SGBD. Sua função é fornecer uma independência do banco de dados, através do desacoplamento entre a camada de negócio e a camada de persistência. Ela utiliza o padrão de projeto *Strategy* [25].

A aplicação faz uso de objetos complexos, que efetivamente são abstrações do domínio do problema, sem se acoplar ao SGBD que está sendo utilizado. Esta abordagem facilita o desenvolvimento na medida que traz maior transparência para o desenvolvimento.

A classe **BD** é uma classe abstrata que implementa a interface IF_BD. Nela são implementados os métodos que são comuns para conseguir efetuar uma conexão com qualquer tipo de SGBD. Os métodos não implementados nesta classe são delegados para as classes que a estendem.

A classe **BDOracle** é uma especialização da classe *BD* e implementa os métodos para se efetuar uma conexão específica para SGBD Oracle.

Implementação do Pacote CargaMetadado

Ilustrado na figura 18, o pacote *CargaMetadado* foi implementado atendendo à seguinte funcionalidade:

- RF1 - Deve ser projetado o esquema conceitual do DW.

A principal função da classe **ServidorBeans** é "ler" o arquivo **ConfigDWMolap.xml**, apresentado no Apêndice B, que contêm o esquema conceitual do DW utilizado no protótipo e instanciar as classes de negócio que estão associadas com suas respectivas classes de carga. A classe do **ServidorBeans** possui uma referência da classe **CargaMetadado**, que representa o esquema conceitual do DW implementado no protótipo. Sua função principal é “encapsular” a navegação do modelo de metadados, isolando a complexidade para os subsistemas que a acessam. A classe **CargaMetadado,** por sua vez, possui uma referência à classe **CargaArquivoDW**, necessária para integração com a biblioteca JConfig [22]. Ela simplesmente informa ao JConfig qual arquivo contém as informações que definem o comportamento dos objetos a serem instanciados. O referido arquivo é o já citado *ConfigDWMolap.xml*.

A classe **CargaDataWarehouse** - é composta por uma coleção de data warehouses da classe *DataWarehouse*.

A classe **CargaArvore-b** - é composta por uma coleção de árvores-b da classe *Arvore-b*.

A classe **CargaCubo** - é composta por uma coleção de cubos da classe *Cubo*.

A classe **CargaAtributo** - é composta por uma coleção de atributos da classe *Atributo*.

A classe **CargaDimensão** - é composta por uma coleção de dimensões da classe *Dimensão*.

As classes *DataWarehouse*, *Arvore-b*, *Cubo*, *Atributo e Dimensao* serão abordadas no pacote *Negócio*.

Implementação do Pacote Negocio

Ilustrado na figura 20, o pacote *Negocio* foi implementado para atender aos seguintes requisitos funcionais e não-funcionais:

- CU3 - Realiza Consultas
- ES4 - Tipo de interface desejada.
- ES3 - Quem utilizará o protótipo?

A classe **DataWarehouse** implementa a interface **IF_DATAWAREHOUSE** e é composta por uma coleção de dimensões do tipo **IF_DIMENSAO** e uma coleção de cubos do tipo **IF_CUBO**. Os objetos da classe **Dimensao**, que implementam a interface *IF_DIMENSAO*, servem para a montagem dos cubos. Já um objeto da classe **Cubo**, que implementa a interface *IF_CUBO*, é o *cubo de dados,* definido na seção 2.5.

A classe *Cubo* agrega uma coleção de elementos da classe **IF_ELEMENTOCUBO**, uma coleção de operações OLAP do tipo **IF_OPERACOESMOLAP,** e agrega também uma coleção de objetos do tipo chunk da classe **Chunk**, semanticamente definido na seção 2.5.2. A interface *IF_OPERACOESMOLAP* provê métodos para realização de uma consolidação. Na verdade, esta classe delega essa responsabilidade para uma *stored procedure,* implementada no SGBD, que de fato realiza a operação. O mecanismo de realização

de uma consulta no protótipo é apresentado mais detalhadamente na implementação do pacote *EsquemaSGBD* e na seção 3.3. Assim, define-se a estrutura de um *cubo de dados.*

Ainda foram implementadas as classes *JSP*, que atendem ao requisito não-funcional *ES4*. Com isso, qualquer usuário conhecedor do modelo OLAP, poderá realizar consultas no protótipo.

Implementação do Pacote EsquemaSGBD

O esquema físico do DW foi implementado no SGBD-OR Oracle8i.

Temos o cubo *Vendas,* que é composto das dimensões *Loja, Produto e Tempo,* e os elementos (Fatos) *Quantidade e Faturamento*. As agregações são executadas conforme as hierarquias definidas no requisito funcional *RF1*. Ou seja, as mudanças de "granularidade" do cubo devem obedecer às ordens das hierarquias. Por exemplo, para o cubo *Vendas* mudar a "granularidade" da dimensão *Tempo* que está em dia, para mês, deve-se utilizar a hierarquia *Mês*; para mudar a "granularidade" da dimensão *Produto* que está em produto, para categoria do produto, deve-se utilizar a hierarquia *Categoria*; para mudar a "granularidade" da dimensão *Loja* que está em loja, para região da loja, deve-se utilizar a hierarquia *Região*. A dimensão *Loja* possui 30 (trinta) elementos. A dimensão *Produto* possui 50 (cinquenta) elementos e a dimensão *Tempo* possui 2000 (dois mil) elementos. O cubo *Vendas* possui 3000000 (três milhões) fatos (o cubo não possui esparsidade devido a utilização da técnica de compressão Chunk-Offset, visto na seção 2.5.5).

A figura 22 ilustra o esquema físico do DW, contemplado pelo pacote *EsquemaSGBD*.

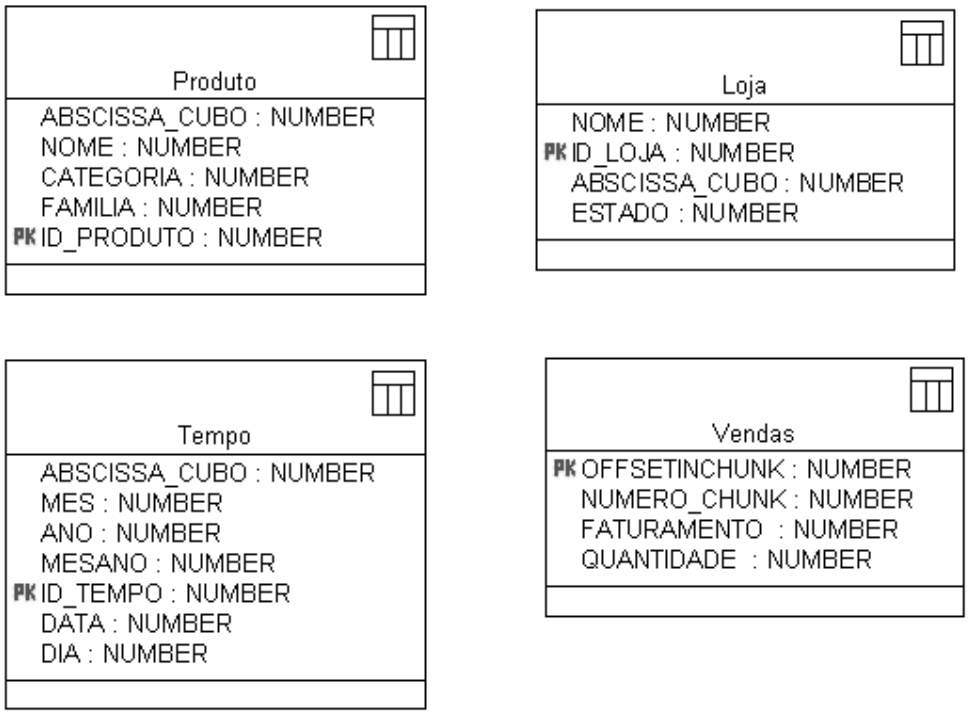

Figura 22: Tabelas do Pacote EsquemaSGBD

Como mencionado na seção 3.1, parte da "inteligência" do protótipo foi implementa no SGBD. No início deste trabalho, primeira iteração, a idéia era manter uma total independência do protótipo no que diz respeito ao SGBD, no qual toda a "inteligência" do protótipo ficava concentrada na aplicação. Entretanto, ao final da primeira iteração, observamos que esta estratégia não proporcionou resultados satisfatórios quando uma consulta era submetida. Depois de analisarmos as informações geradas no arquivo *ProfingMolap*, apresentado no apêndice C, percebemos que o "gargalo" do protótipo se encontrava na quantidade de vezes que a aplicação se comunicava com o SGBD para executar uma leitura ao disco. Aproximadamente, cerca de 74% do tempo da aplicação era dispensado nessa atividade. Para se ter uma noção, uma consulta de consolidação realizada em 5 (cinco) lojas com 15 (quinze) produtos cada, durante 3 (três) meses, o que nos dá um total de 6750 (seis mil e setecentos e cinqüenta)

registros, demorava cerca de 1 (um) minuto e 20 (vinte) segundos para ser realizada. Então chegamos a algumas conclusões:

1) A lógica de recuperação dos fatos não poderia permanecer na aplicação. Logo, optamos em transferir essa "inteligência" para o SGBD.
2) A utilização do conceito de chunk, até então não utilizado, tornou-se imprescindível. Além de transferir a lógica de recuperação para o SGBD, precisávamos otimizar a recuperação dos fatos do nosso cubo.

Pautando-nos nestas conclusões, planejamos nossa segunda iteração, que, fez com que remodelássemos nosso esquema fisico para atender a estas necessidades.

O pacote *EsquemaSGBD* foi implementado atendendo às seguintes funcionalidades:

- CU1 - Cria DW
- CU2 - Define padrões de acesso.
- CU3 - Realiza Consultas
- ES3 - Documentação disponibilizada.

Associando as árvores-b

Como ilustrado na figura 22, cada tabela dimensão possui uma coluna chamada **ABSCISSA_CUBO**. Esta coluna corresponde ao valor mapeado do registro no cubo. Isto significa dizer que um produto da tabela *Produto*, cujo **ID_PRODUTO = 1** e **ABSCISSA_CUBO = 1** está mapeado no eixo dos *Produtos* do cubo cuja abscissa é 1. Isso vale para as demais tabelas de dimensão. Foi criado um índice composto do tipo árvore-b do Oracle destas duas colunas. Desta forma, está associado o índice de cada registro das tabelas dimensão, com seus respectivos valores no cubo, atendendo assim, ao requisito funcional *RF5*.

Definindo padrões de acesso

Como mencionado na seção 2.5.1, padrões de acesso podem ser definidos através de estatísticas de acesso ao banco ou pelo usuário final. Desta maneira, foram estabelecidos 4 (quatro) padrões de acesso ao protótipo:

- **Padrão 1:** (30, 50, 31) com probabilidade de 20% de acontecer. Isto significa dizer que consultas referentes ao faturamento de todas as lojas por mês se enquadram neste padrão.

- **Padrão 2:** (30, 50, 90) com probabilidade de 10% de acontecer. Isto significa dizer que consultas referentes ao faturamento de todas as lojas por trimeste se enquadram neste padrão.

- **Padrão 3:** (30, 50, 180) com probabilidade de 30% de acontecer. Isto significa dizer que consultas referentes ao faturamento de todas as lojas por semestre se enquadram neste padrão.

- **Padrão 4:** (30, 50, 365) com probabilidade de 40% de acontecer. Isto significa dizer que consultas referentes ao faturamento de todas as lojas por ano se enquadram neste padrão.

Com a definição destes padrões, temos o caso de uso *CU2* atendido.

Definindo a forma do chunk

Uma vez estabelecidos os padrões de acesso, o passo seguinte foi definir a forma do chunk. Para se fazer isso, a primeira análise a ser feita era: Quantos registros da tabela **VENDAS** cabem em uma página de disco? (Como mencionado na seção 2.5.2, neste trabalho, a página de disco utilizada possui o de o tamanho de 8KB.)

Em [26], é apresentada a fórmula para se calcular o tamanho em bytes de um registro em uma tabela. Para o tipo NUMERIC do Oracle, a fórmula é a seguinte:

$$((p + s) / 2) + hs$$

onde: *p* é precisão do número, *s* é a escala do número e *hs* é o tamanho do cabeçalho do registro.

Obs.: Para este tipo, o cabeçalho do registro equivale a um byte por coluna. Logo, para um registro de 4 colunas, o cabeçalho possui o tamanho de 4 bytes.

A tabela VENDAS foi definida da seguinte maneira:

```
CREATE OR REPLACE TABLE VENDAS (
OFFSETINCHUNCK NUMERIC(10,0),
NUMERO_CHUNK NUMERIC(4,0) PRIMARY KEY,
QUANTIDADE NUMERIC(6,0)PRIMARY KEY,
FATURAMENTO NUMERIC(10,2));
```

Aplicando-se a fórmula do cálculo do tamanho do registro, encontramos o tamanho de 20 bytes para cada registro.
Já que o tamanho da página de disco é 8KB e cada registro possui o tamanho de 20 bytes, então temos um total de 400 registros por página.

Para encontrar a melhor forma do chunk, primeiro desenvolvemos um algoritmo (A) que calculasse a combinação $C_r(400,3)$, sendo que para cada elemento da combinação eram verificadas as restrições (a) e (b) apresentadas na seção 2.5.2. Estávamos procurando todas as possibilidades de formarmos um elemento com três números que obedecessem às restrições. Os elementos encontrados são apresentados na tabela 2.

[1, 1, 400]	[2, 1, 200]	[5, 1, 80]	[10, 1, 40]
[1, 2, 200]	[2, 2, 100]	[5, 2, 40]	[10, 2, 20]
[1, 5, 80]	[2, 5, 40]	[5, 5, 16]	[10, 5, 8]
[1, 10, 40]	[2, 10, 20]	[5, 10, 8]	[10, 10, 4]

Tabela 2: Elementos gerados pelo algoritmo (A)

O código Java do algoritmo (A) foi:

```
// Vetor que receberá todos os elementos gerados através da combinação C_r(400,3)
   que satisfaçam as restrições (a) e (b) apresentadas na seção 2.5.2 deste
   trabalho.

Vector elementosGeradosCombinacao = new Vector();

                                                                    C_r(400,3)
// Três laços aninhados que percorrem todos os elementos da combinação
for (int i=1 ; i <= 400;i++)
{
  for (int j=1 ; j <= 400;j++)
  {
    for (int k=1 ; k <= 400;k++)
    {
      // Condição que testa se o elemento [i,j,k] da combinação atende as
         restrições (a) e (b) apresentadas na seção 2.5.2 deste trabalho.
      if ( (i*j*k) == 400 & ((30 % i == 0) & (50 % j == 0) & (2000 % k == 0)) )
      {
        int[] elementoCombinacao = new int[3];

        elementoCombinacao[0] = i;
        elementoCombinacao[1] = j;
        elementoCombinacao[2] = k;

        elementosGeradosCombinacao.add(elementoCombinacao);
      }
    }
  }
}
```

Os elementos foram armazenados em um objeto Java do tipo Vector do pacote java.util.

Desenvolvemos um algoritmo (B) que calculasse a fórmula (1) apresentada na seção 2.5.2. A fórmula nos fornece o número médio de blocos recuperados para um específico padrão de acesso e uma forma de "chunk". O objetivo desta fórmula é escolher a melhor forma para o "chunk", que satisfaça as restrições (a) e (b), apresentadas também na seção 2.5.2, e minimize a fórmula (1). Como já identificamos os elementos que satisfizeram as restrições (a) e (b), restava-nos calcular qual deles minimizaria (1), encontrando assim, a melhor forma para nosso chunk.

O código Java do algoritmo (B) foi:

```
.
.
.
Vector colecaoPadroesAcesso = new Vector();
int[] elementoGeradoCombinacao = new int[3];
double result = 100000.0d;
double result2 = 0.0d;

int[] padraAcesso1 = new int[3];
padraAcesso1[0] = 30;
padraAcesso1[1] = 50;
padraAcesso1[2] = 31;
```

```
int[] padraAcesso2 = new int[3];
padraAcesso2[0] = 30;
padraAcesso2[1] = 50;
padraAcesso2[2] = 90;

int[] padraAcesso3 = new int[3];
padraAcesso3[0] = 30;
padraAcesso3[1] = 50;
padraAcesso3[2] = 180;
```

```
int[] padraAcesso4 = new int[3];
padraAcesso4[0] = 30;
padraAcesso4[1] = 50;
padraAcesso4[2] = 365;

int[] ordemCubo = new int[3];
ordemCubo[0] = 30;
ordemCubo[1] = 50;
ordemCubo[2] = 2000;

int[] formaChunk = new int[3];

colecaoPadroesAcesso.add(padraAcesso1);
colecaoPadroesAcesso.add(padraAcesso2);
colecaoPadroesAcesso.add(padraAcesso3);
colecaoPadroesAcesso.add(padraAcesso4);

double[] probabilidadesPadroesAcesso = new double[5];
probabilidadesPadroesAcesso[0] = 0.2d;
probabilidadesPadroesAcesso[1] = 0.1d;
probabilidadesPadroesAcesso[2] = 0.3d;
probabilidadesPadroesAcesso[3] = 0.4d;

// Laço que varre o Vector que contêm os elementos gerados pelo algoritmo (A)
for (int i =0; i < elementosGeradosCombinacao.size(); i++)
{
   elementoGeradoCombinacao = (int[])elementosGeradosCombinacao.get(i);
   // São passados como parâmeto: um elemento gerado pela combinação do algoritmo (A),
      os padrões de acesso, a ordem cubo e as probabilidades de cada padrão de acesso
   result2 = numeroMedioBlocos(elementoGeradoCombinacao, colecaoPadroesAcesso, 3,
                               probabilidadesPadroesAcesso);
   if (result > result2)
   {
    result = result2;
    formaChunk[0] = elementoGeradoCombinacao[0];
    formaChunk[1] = elementoGeradoCombinacao[1];
    formaChunk[2] = elementoGeradoCombinacao[2];
   }
}
.
.
.
```

Método numeroMedioBlocos

```
public static double numeroMedioBlocos(int[] ordenadaChunk, Vector k, int n,
                                       double[] probabilidades ){
   double somatorio, produtorio, elementoProdutorio;
   int[] padraoAcesso = new int[n];
   somatorio = 0;

   // laço que percorre os padrões de acesso e calcula o somatório
   for (int i = 0; i < k.size() ; i++){
     padraoAcesso = (int[])k.get(i);
     produtorio = 1;

     //laço que calcula o produtório
     for (int j = 0; j < n; j++){
        elementoProdutorio = Math.ceil((double)padraoAcesso[j]/(double)ordenadaChunk[j])
      }

      somatorio = somatorio + (produtorio * probabilidades [i]);
    }
    return somatorio;
}
```

Depois de submetermos os elementos gerados pelo algoritmo (A) ao algoritmo (B), encontramos que a melhor forma para o nosso chunk é:

(10, 10, 4)

Com isso, atendemos à funcionalidade do caso de uso *CU1*.

Reordenando os dados dos Chunks

Uma vez estabelecidos os padrões de acesso, e forma do chunck, o passo seguinte foi verificar qual era a melhor disposição dos eixos do chunk para os padrões de acesso estabelecidos. Para se fazer isso, foi utilizado o lemma 1 apresentado na seção 2.5.3 Submetendo os dados até então encontrados, foi observado que se trocássemos a nossa ordem dos eixos inicialmente definidas como Loja→Produto→Tempo para Loja→Tempo→Produto, obteríamos um melhor desempenho das consultas realizadas no protótipo, segundo os padrões de acesso estabelecidos. Com isso, a forma do nosso chunk mudara para:

(10, 4, 10)

Com isso, atendemos à funcionalidade do caso de uso *CU1*.

Realizando uma consulta

Como mencionado na implementação do pacote *Negocio*, a aplicação delega a responsabilidade de realizar a consulta para o SGBD. A aplicação é responsável por receber os elementos das dimensões que o UsuárioDW deseja consolidar. Uma vez recebidas as informações ela delega a responsabilidade para o SGBD.

Foi construída uma *stored procedure* no SGBD, a qual recebe três parâmetros strings com textos de comandos SQL os quais retornam os identificadores e a entradas no cubo dos elementos das dimensões passadas pela aplicação.

Tendo recebido os parâmetros, a *stored procedure* realiza os seguintes passos:

1) executa os três comandos SQL e armazena o resultado da execução de cada um dos comandos em um cursor diferente, utilizando o recurso de SQL dinâmico.

2) constrói três laços de repetição aninhados para gerar o produto cartesiano entre os valores correspondentes às abscissas do cubo, que foram recuperados no passo 1, sendo que cada um dos laços contém as abscissas dos elementos de uma determinada dimensão.

3) para cada elemento do produto cartesiano (abscissa_loja, abscissa_produto, abscissa_tempo) é calculada a posição dentro do chunk na qual a sua ordenada se encontra.

4) são selecionados todos os registros que possuem o número do chunk calculado. Como mencionado na *definição da forma do chunk*, são recuperados 400 registros da tabela VENDAS, ocupando exatamente uma página de disco do tamanho de 8KB.

5) depois de encontrado o número do chunk, é calculada a posição da ordenada que estamos procurando dentro do chunk encontrado, para a partir daí, encontrarmos o offSetInChunk, que retorna o registro que está sendo procurado.

6) verifica-se se o registro recuperado existe. Se existir, o valor é agregado a uma variável que, no fim destes passos, conterá o resultado da consolidação submetida.

Vale frisar que o Oracle, antes de executar uma operação *read* no disco, verifica se o dado pesquisado já está em *cache*. Caso o dado esteja na *cache*, este dado é acessado diretamente em memória. Isto é feito com intuito de diminuir o número de acessos ao disco, o que degrada o desempenho de uma consulta.

Como a tabela *VENDAS* representa a forma linear de um array de ordem [30, 50, 2000] e os dados estão dispostos nesta tabela segundo os conceitos de chunk, estamos tentando aqui maximizar o desempenho das consultas OLAP realizadas no protótipo. A seção 3.3 apresenta um estudo de caso da realização de uma consulta OLAP no protótipo.

O código da *stored procedure* foi:

```
create or replace procedure sys.calculaValorVendido(sum_qt_vendas out number,
sum_vl_vendas, cursor cursorTempo, cursor cursorLoja, cursor cursorProduto out
number) is

 dimTempo number;
 dimLoja number;
 dimProduto number;
 pkHash number;
 qt_vendas number;
 vl_vendas number;


begin
  sum_qt_vendas := 0;
  sum_vl_vendas  := 0;
  dimTempo := 1095;
  dimLoja  := 9;
  dimProduto := 15;
  FOR ct_rec in cursorTempo LOOP
   FOR cl_rec in cursorLoja LOOP
     FOR cp_rec in cursorProduto LOOP
       //Calcula o offSetInChunk
       pkHash := (ct_rec.pk_hash-1)*(dimLoja*dimProduto) +
                 (cl_rec.pk_hash-1)*(dimProduto) + cp_rec.pk_hash;
       select qt_vendida, vl_vendido into qt_vendas, vl_vendas
       from tb_molap_vendas
       where pk_hash = pkHash;
       sum_qt_vendas := sum_qt_vendas + qt_vendas;
       sum_vl_vendas := sum_vl_vendas + vl_vendas;
     END LOOP;
   END LOOP;
 END LOOP;
end calculaValorVendido;
```

O apêndice D apresenta um pequeno exemplo da disposição dos dados (offsetInChunk e numero_chunk) na tabela VENDAS.

Com isso, temos atendido à funcionalidade do caso de uso *CU3*.

Documentação produzida

Ao final da implementação deste pacote, foi gerada toda documentação esperada segundo especificação suplementar *ES5*.

3.2.4 Artefatos da fase de Transição

O objetivo da fase de transição é garantir que o software esteja disponível para o usuário final. Nesta fase, o foco deve ser voltado principalmente para o as revisões e aceites do usuário final, fazendo assim com que o software seja ajustado, configurado, instalado, e que sejam realizados os testes de usabilidade. Freqüentemente, isto é feito quando o software está em produção.

As atividades da fase de transição dependem de um aceite formal do usuário final. Por exemplo, quando são encontrados erros, implementação e testes

normalmente são suficientes. Entretanto, se houver o acréscimo de novas funcionalidades, dever ser realizada uma iteração similar à fase de construção, com análise de requisitos, projeto, etc.

O objetivo deste trabalho não foi construir um software com intuito de entrar em produção; portanto, não houve nenhum artefato gerado nesta fase. Algumas sugestões são colocadas como trabalhos futuros.

3.3 Estudo de caso: Realização de uma consulta OLAP

Antes de entendermos como uma consulta é processada no protótipo, vamos formalizar o cálculo do número do chunk e da posição de uma ordenada (x, y, z) dentro do chunk, uma vez que é sabido como calcular o offSetInChunk (ver seção 2.5.5).

Sejam: os limites de um cubo $C = (l_1, l_2, l_3)$, a forma de um chunk $CK = (f_1, f_2, f_3)$ e capacidade de um chunk $CP = (cp_1, cp_2, cp_3)$ para um cubo C, temos que o número do chunk no qual uma ordenada (x, y, z) se encontra é:

$$(x, y, z) = \left(\left(\left\lceil \frac{x}{f_1} \right\rceil - 1\right) * cp_2 * cp_3 + \left(\left\lceil \frac{y}{f_2} \right\rceil - 1\right) * cp_3 + \left(\left\lceil \frac{z}{f_3} \right\rceil - 1\right)\right) + 1$$

O código Java do algoritmo (C), utilizado para calcular o número do chunk, foi:

```
public double numeroChunk(double x, double y, double z, double f1, double f2, double f3,
                          double cp1, double cp2, double cp3) throws Exception{
   double numeroChunk = 0, ceilXF1, ceilYF2, ceilZF3 = 0;
   ceilXF1 = Math.ceil(x/f1) - 1;
   ceilYF2 = Math.ceil(y/f2) - 1;
   ceilZF3 = Math.ceil(z/f3) - 1;

   numeroChunk = ((ceilXF1 * cp2 * cp3) + (ceilYF2 * cp3) + ceilZF3) + 1;

   return numeroChunk;
}
```

Seja uma ordenada (x, y, z) pertencente a um cubo $C = (l_1, l_2, l_3)$, uma forma de chunk definida como $CK = (f_1, f_2, f_3)$, temos que o deslocamento no chunk cuja uma ordenada (x, y, z) se encontra é:

$$\left(x \bmod f_1, y \bmod f_2, z \bmod f_3\right)$$

O código Java do algoritmo (D) utilizado para calcular a posição no do chunk foi:

```
public int[] calculaPosicaoChunk(int[] ordenada, int[]formaChunk)
{
   int[] ordenadaChunk = new int[ordenada.length];

    for (int i = 0; i < ordenada.length; i++){
      ordenadaChunk [i] = (ordenada [i] % formaChunk [i] == 0 ? formaChunk [i] :
                           ordenada [i] % formaChunk [i]);
    }
   return ordenadaChunk;
}
```

A figura 23 ilustra um array *3*-dimensional de ordem (4, 4, 4) divido em chunks da forma (2, 2, 2).

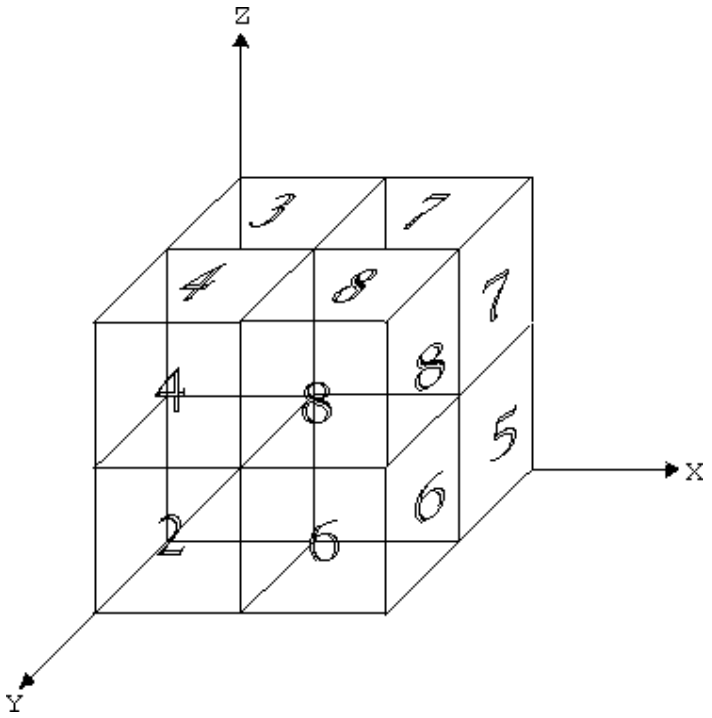

Figura 23: Array (4, 4, 4) divido em Chunks (2, 2, 2)

O que acontece é uma enumeração dos dados do array em estruturas chamadas de chunk. Como é possível observar, primeiramente são fixados os eixos X e Z do cubo, variando o eixo Y até atingir o limite do eixo Y do chunk. Em seguida, o eixo X permanece fixo, ocorrendo as variações dos eixos Z e Y até atingirem respectivamente os limites dos eixos Y e Z do chunk. Por último, são fixados os eixos Y e Z do cubo, variando o eixo X. Isto é feito até que se tenha finalizado o preenchimento do primeiro chunk. Uma vez finalizado o preenchimento do primeiro chunk, segue-se a enumeração através dos demais chunks, até que se tenha um completo preenchimento

de todos. Desta maneira, a navegação no cubo é realizada pelos eixos Y→Z →X. A figura 24 ilustra o preenchimento do array divido em chunks apresentado na figura 23.

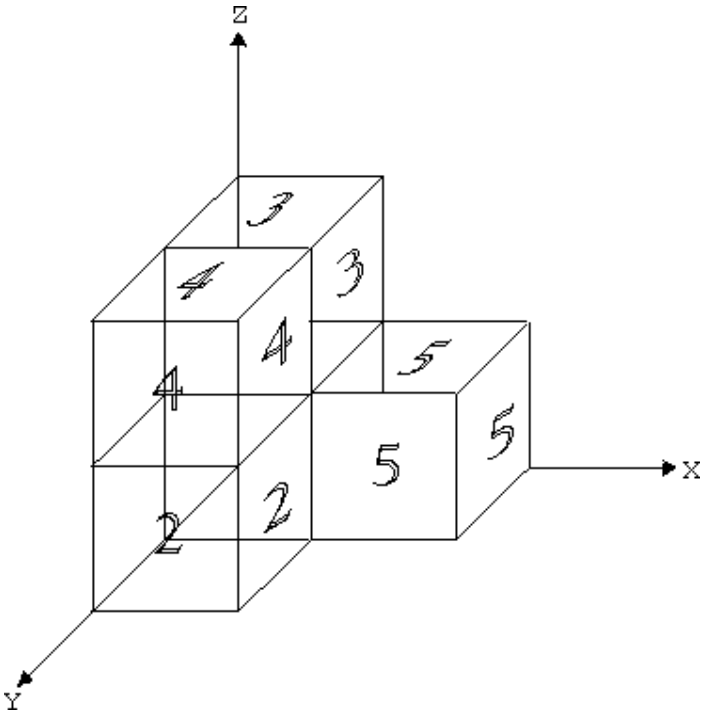

Figura 24: Preenchimento de um array (4, 4, 4) divido em Chunks (2, 2, 2)

A tabela 3 ilustra a disposição física dos dados do array apresentado na figura 23.

OffSetInChunk	Abscissa		OffSetInChunk	Abscissa		OffSetInChunk	Abscissa		OffSetInChunk	Abscissa	
	chunk	cubo		chunk	cubo		chunk	cubo		chunk	cubo
Nº Chunk: 1			**Nº Chunk: 3**			**Nº Chunk: 6**			**Nº Chunk: 7**		
1	1 1 1	1 1 1	1	1 1 1	1 3 1	1	1 1 1	3 1 1	1	1 1 1	3 3 1
2	1 1 2	1 1 2	2	1 1 2	1 3 2	2	1 1 2	3 1 2	2	1 1 2	3 3 2
5	1 2 1	1 2 1	5	1 2 1	1 4 1	5	1 2 1	3 2 1	5	1 2 1	3 4 1
6	1 2 2	1 2 2	6	1 2 2	1 4 2	6	1 2 2	3 2 2	6	1 2 2	3 4 2
17	2 1 1	2 1 1	17	2 1 1	2 3 1	17	2 1 1	4 1 1	17	2 1 1	4 3 1
18	2 1 2	2 1 2	18	2 1 2	2 3 2	18	2 1 2	4 1 2	18	2 1 2	4 3 2
21	2 2 1	2 2 1	21	2 2 1	2 4 1	21	2 2 1	4 2 1	21	2 2 1	4 4 1
22	2 2 2	2 2 2	22	2 2 2	2 4 2	22	2 2 2	4 2 2	22	2 2 2	4 4 2
Nº Chunk: 2			**Nº Chunk: 4**			**Nº Chunk: 5**			**Nº Chunk: 8**		
1	1 1 1	1 1 3	1	1 1 1	1 3 3	1	1 1 1	3 1 3	1	1 1 1	3 3 3
2	1 1 2	1 1 4	2	1 1 2	1 3 4	2	1 1 2	3 1 4	2	1 1 2	3 3 4
5	1 2 1	1 2 3	5	1 2 1	1 4 3	5	1 2 1	3 2 3	5	1 2 1	3 4 3
6	1 2 2	1 2 4	6	1 2 2	1 4 4	6	1 2 2	3 2 4	6	1 2 2	3 4 4
17	2 1 1	2 1 3	17	2 1 1	2 3 3	17	2 1 1	4 1 3	17	2 1 1	4 3 3
18	2 1 2	2 1 4	18	2 1 2	2 3 4	18	2 1 2	4 1 4	18	2 1 2	4 3 4
21	2 2 1	2 2 3	21	2 2 1	2 4 3	21	2 2 1	4 2 3	21	2 2 1	4 4 3
22	2 2 2	2 2 4	22	2 2 2	2 4 4	22	2 2 2	4 2 4	22	2 2 2	4 4 4

Tabela 3: Array de Chunks Linearizado

A seguir, apresentamos o funcionamento interno do protótipo para a resolução da operação de consolidação. A operação será executada no esquema DW Vendas, apresentado na seção 3.2.3 (implementação do pacote *EsquemaSGBD*).

A figura 25 ilustra a interface utilizada pelo usuário para a realização da consulta.

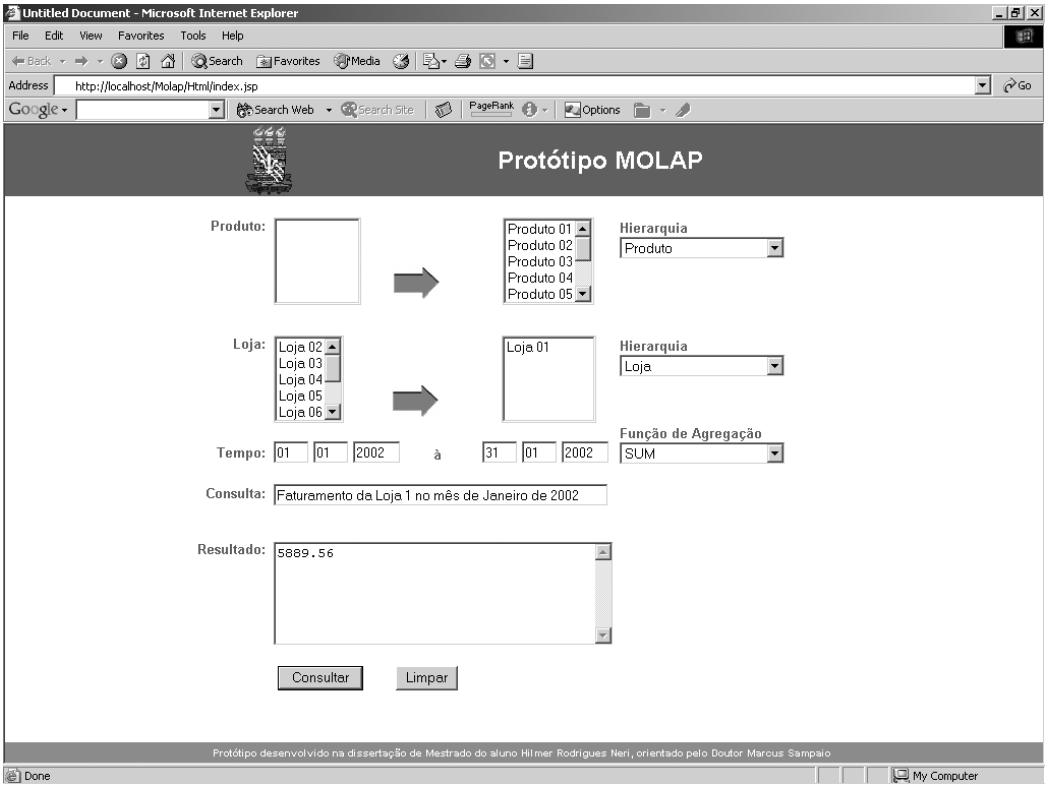

Figura 25: Interface do protótipo para o processamento de consultas OLAP

Os passos para resolução da consulta apresentada na figura 25 são os seguintes:

1) montar três strings de comandos SQL, que retornem os elementos das tabelas dimensões informados pelo usuário, passando-os como parâmetro para a *stored procedure calculaValorVendido*.

2) a *stored procedure* recebe os três strings e executa os seguintes passos:

   2.1) seleciona na tabela dimensão PRODUTO todas as colunas ABSCISSA_CUBO dos produtos cujos nomes sejam iguais aos informados pelo usuário e armazena o resultado no *cursorProduto*.

2.2) seleciona na tabela dimensão LOJA a coluna ABSCISSA_CUBO da loja cujo nome seja igual a *Loja1* e armazena o resultado no *cursorLoja*.

2.3) seleciona na tabela dimensão TEMPO todas as colunas ABSCISSA_CUBO, dos dias compreendidos entre a data 01/01/2002 e 31/01/2002 e armazena o resultado no *cursorTempo*.

2.4) gera um produto cartesiano entre os elementos dos três cursores.

2.5) para cada elemento do produto cartesiano, a *stored procedure* calcula o número do chunk onde aquele elemento se encontra. Por exemplo: Para ordenada (10, 4, 19) o número do chunk calculado pelo algoritmo (C) é 2.

2.6) recupera na tabela VENDAS todos os registros cuja coluna NUMERO_CHUNK seja igual a 2.

2.7) uma vez recuperados os registros do chunk procurado, a *stored procedure* calcula a posição dentro do chunk da ordenada procurada. Retomando a ordenada apresentada no passo 2.5 é retornada a ordenada (10, 4, 9) correspondente ao deslocamento dentro do chunk.

2.8) identificada a posição dentro do chunk, a *stored procedure* calcula o offSetInChunk. Novamente tomando como exemplo a ordenada apresentada no passo 2.5, cujo deslocamento foi calculado no passo 2.7, é calculado o offSetInChunk tendo o número 180039 como retorno, correspondente ao identificador deste deslocamento.

2.9) é recuperado na tabela VENDAS o registro cuja coluna NUMERO_CHUNK seja igual ao número do chunk encontrado no passo 2.5 e a coluna OFFSETINCHUNK que seja igual ao identificador do deslocamento no chunk, encontrado no passo 2.8.

2.10) se o registro foi recuperado, então o valor da coluna QUANTIDADE é agregado na variável *sum_qt_vendas* e o valor da coluna FATURAMENTO é agregado na variável *sum_vl_vendas*.

2.11) os passos 2.8 ao 2.10 são executados até que todos os elementos do produto cartesiano tenham sido processados.

Capítulo 4

Avaliação do Trabalho

Este capítulo analisa se os requisitos funcionais e não funcionais esperados do *protótipo MOLAP* foram atingidos. Além disso, avalia a aplicabilidade do *protótipo* na realização de consultas OLAP.

A seção 4.1 verifica se os requisitos funcionais explicados na seção 3.2.1 foram atingidos. Em seguida, a seção 4.2 analisa até que ponto foram alcançados os requisitos não funcionais (especificações suplementares), também estabelecidos na seção 3.2.1.

4.1 Verificação dos requisitos funcionais

Para analisar até que ponto os requisitos funcionais explicados na seção 3.2.1 foram atendidos, é apresentada uma explicação de cada um deles em separado. Uma boa maneira de se verificar até que ponto os requisitos foram atendidos é explicar como o protótipo atendeu a cada funcionalidade.

RF1 - Deve ser projetado o esquema conceitual do DW.

Como descrito no apêndice B, foi projetado um esquema conceitual que contempla os metadados do protótipo. Este esquema torna possível a navegação nos metadados, desta forma, quando a aplicação está sendo executada, os dados do DW são apresentados segundo a descrição projetada no esquema conceitual.

O esquema conceitual poderia ter sido projetado em um arquivo que é tratado de forma estática, uma maneira muito utilizada para descrever metadados. Isso implica que toda vez que surge uma necessidade de alterar o esquema conceitual, necessariamente tem que se fazer uma alteração na aplicação. Com a utilização do arquivo XML, uma vez realizada a alteração do esquema conceitual no arquivo XML, a aplicação ao carregar este arquivo, passa a trabalhar com as alterações

de forma dinâmica, ou seja, não é preciso realizar mudanças no código para que a aplicação passe a "enxergar" essas alterações. Isto facilita mudanças e justifica a decisão de não utilizar arquivos estáticos.

RF2 - Deve permitir que consultas sejam realizadas em granularidades maiores.

Consultas em granularidades maiores dizem respeito à navegação nos agregados de um DW. Foi implementado o algoritmo apresentado na seção 2.5, que realiza uma consolidação. O protótipo não contemplou a funcionalidade de se realizar consultas a partir dos agregados, portanto, nos testes apresentados na seção 4.2 deste capítulo não é apresentada nenhuma consulta OLAP que esteja relacionada com a navegação a partir dos agregados. Na avaliação comparativa de desempenho entre o protótipo e o esquema projetado com a técnica de *Esquema em Estrela* não foram levadas em consideração consultas que recuperam dados a partir dos agregados. Na seção 5.2 se sugere como trabalho futuro, a implementação desta funcionalidade.

RF3 – Devem ser definidos os padrões de acesso ao cubo, bem como suas respectivas probabilidades de ocorrerem.

No caso de uso *CU2* da seção 3.2.1, foram definidos os padrões de acesso para as consultas realizadas no protótipo. Os padrões de acesso e respectivas probabilidades, foram estabelecidos seguindo uma tendência de tipos de consultas mais comumente realizadas em um ambiente de DWing.

RF4 – Devem ser implementados algoritmos de consultas sobre o array dividido em chunks.

Foi implementada uma *stored procedure* que permite a realização de consultas OLAP pelo protótipo. Esta *stored procedure* recebe três *strings* com os comandos SQL para recuperar os elementos das dimensões que foram informados pelo usuário. Cada comando SQL é executado e o resultado é armazenado em um cursor. É realizado um

produto cartesiano entre os elementos dos três cursores e para cada elemento do produto cartesiano é recuperado um registro na tabela de fatos. Caso seja recuperado o registro, então são agregados os valores correspondentes aos fatos. Ao final do produto cartesiano, tem-se sumariados os fatos referentes aos elementos das dimensões informados pelo usuário. A *stored procedure* foi apresentada na seção 3.2.2.

RF5 – Devem ser criados índices de árvore-b que associem os elementos das tabelas dimensões com suas respectivas abscissas no cubo.

Foram criados três índices do tipo árvore-b, disponível no Oracle, no qual os argumentos desses índices são as colunas que representam o identificador do elemento da dimensão e sua abscissa no cubo, de tal forma que, ao recuperar o elemento da dimensão, recupera-se a coluna que mapea sua entrada no cubo.

4.2 Verificação dos requisitos não funcionais (especificações suplementares)

Esta seção dedica-se a analisar o nível de satisfação dos requisitos não funcionais elicitados nas especificações suplementares, definidos na seção 3.2.1. São apresentados os resultados dos testes comparativos de desempenho entre as consultas OLAP realizadas no protótipo e no esquema em estrela.

ES1 - Deve ser configurado o SGBD a ser utilizado

Como mencionado no capítulo 3, o SGBD escolhido para este trabalho foi o SGBD-OR Oracle 8i. Para se conectar ao banco a aplicação carrega o arquivo XML *ConfigDWMOLAP.xml* (descrito no apêndice B) através da classe *ServidorBeans*. A classe *CargaMetaDado* instancia um objeto do tipo *CargaDataWarehouse,* que por sua vez mantém as informações necessárias para se conectar ao Oracle (login, senha, ip, porta e sid). Uma vez carregadas às

informações, a classe *ServidorMOLAP,* através do método `criaConexao(CargaDataWarehouse cargaDataWarehouse)`, encarrega-se de estabelecer a conexão com o banco. O SGBD-OR Oracle 8i possui uma linguagem procedural, que suporta alguns conceitos de orientação a objetos chamada de *Procedural Language Extensions* to SQL (PL/SQL). Através desta linguagem, foi possível implementar os conceitos de chunk.

Não foi utilizado outro SGBD, pois um dos objetivos do trabalho era estudar a viabilidade da construção de um protótipo MOLAP utilizando um SGBD-OR, no caso o Oracle 8i.

ES2 – Deve ser definida uma forma para o chunk, bem como sua implementação.

Os padrões de acesso definidos foram submetidos ao algoritmo (B) apresentado na seção 3.2.3, que implementa a fórmula 1 apresentada na seção 2.5.2, objetivando identificar a melhor forma para o chunk. Foi escolhida a forma (10, 10, 4) que melhor atendia aos padrões de acesso estabelecidos para este trabalho.

ES3 – Deve ser estabelecida uma forma de reordenamento das informações nos chunks.

Os padrões de acesso, forma do chunk, e dimensões do cubo foram submetidos aos lemas 1 e 2 da seção 2.5.3. O objetivo foi reorganizar a disposição dos dados de forma a diminuir o número de acessos ao meio magnético. Por exemplo, com a reordenação, a forma do chunk foi alterada para [10, 4, 10]. Com os padrões definidos e a forma reordenada do chunk, os dados foram dispostos fisicamente (ver apêndice D), de tal forma a melhorar o tempo de resposta das consultas que obedecessem aos critérios definidos pelos padrões de acesso estabelecidos para este trabalho.

ES4 – Tipo de interface desejada.

Foi construída uma página JSP que contempla o tipo de interface esperada. Nesta interface, são disponibilizados os elementos das tabelas dimensões (loja e produto), o critério de agregação destas dimensões e uma "faixa" de datas que deve ser informada, correspondete ao intervalo das datas que compreendem os elementos das dimensões tempo.

ES5 – Documentação disponibilizada.

Foi disponibilizado um modelo projetado na ferramenta Rational Rose utilizando a notação UML. Este modelo provê uma visão das etapas realizadas na implementação do protótipo, permeando os requisitos elicitados, modelo de análise, modelo de projeto e uma visão da arquitetura do sistema.

ES6 – Oferecer uma melhoria de desempenho de consultas OLAP utilizando a abordagem MOLAP com relação à abordagem ROLAP via um SGBD-OR.

A seção seguinte apresenta detalhadamente o experimento realizado que possibilitou-nos observar um ganho de desempenho do protótipo em relação ao esquema estrela dentro do ambiente construído para realização dos testes.

Avaliação do experimento

A seguir, são descritos os procedimentos e métodos utilizados para comparar o desempenho das consultas OLAP realizadas entre o protótipo e o esquema estrela.

Configuração do Sistema: Os testes foram executados em um computador Pentium Intel III 866MHZ com 512MB de memória principal e um disco de 20GB Maxtor DiamondMax Plus 52049H4 ATA/100. O sistema operacional utilizado foi o Windows 2000 Server e o banco de dados utilizado foi o SGBD-OR Oracle 8i versão 8.1.7.

Esquema em Estrela: O esquema relacional utilizado segue abaixo:

- Tabela de fatos (CH_PRODUTO NUMBER(2), CH_LOJA NUMBER(2), CH_TEMPO NUMBER(8), QT_VENDIDA NUMBER(8), VL_VENDIDO NUMBER(8,2));
- Dimensão Tempo(CH_TEMPO NUMBER(8), NU_DIA NUMBER(2), NU_MES NUMBER(2), NU_ANO NUMBER(4), DT_TEMPO DATE);
- Dimensão Loja (CH_LOJA NUMBER(1), DS_LOJA VARCHAR2(30), DS_REGIAO VARCHAR2(30), CD_REGIAO NUMBER(2));
- Dimensão Produto (CH_PRODUTO NUMBER(2), DS_PRODUTO VARCHAR2(30), DS_CATEGORIA VARCHAR2(30), CD_CATEGORIA NUMBER(2));

Consultas OLAP: Foram definidos 10 (dez) tipos de consultas que foram executadas no Protótipo MOLAP e no Esquema Estrela. Foram realizadas 10 (dez) variações entre os elementos das dimensões para cada tipo de consulta. Estas variações disseram respeito a quantidade de dados e aos diferentes elementos das dimensões recuperados. No apêndice E é apresentada uma tabela com todos tempos de execução e as respectivas médias de cada consulta executada. Cada consulta foi executada três vezes tanto no Protótipo MOLAP quanto no Esquema Estrela, perfazendo um total de 600 execuções. A seguir são apresentados os tipos de consultas definidos.

a) Total de Vendas por Mês
b) Total de Vendas por Trimestre
c) Total de Vendas durante Quatro Meses
d) Total de Vendas por Semestre
e) Total de Vendas durante Sete Meses
f) Total de Vendas por Ano
g) Total de Vendas durante Dois Anos
h) Total de Vendas durante Quatro Anos
i) Média de Vendas por Mês agrupado por Ano e Produto
j) Média de Vendas por Trimestre agrupada por Ano e Produto

Conjunto de dados: Foram utilizados dados fictícios para analisar a performance. Foram gerados dados idênticos para o protótipo e para o esquema estrela. São 30 elementos para a dimensão Loja, 50 elementos para a dimensão Produto, e 2000 elementos para a dimensão tempo. Foram gerados 3000000 fatos que representam aproximadamente 57MB de dados armazenados.

A primeira consideração a ser fazer é mostrar que um chunk ocupa uma única página de disco. Como mencionado na seção 3.2.3, um chunk é formado por 400 registros e a página do Oracle está configurada para o tamanho de 8KB (parâmetro db_block_size). A página do sistema operacional também está configurada para o tamanho de 8KB. Para comprovar que um único chunk ocupa uma única página em disco foram utilizadas informações de uma visão do Oracle chamada V$CACHE. Esta visão contém informações sobre a quantidade de blocos ocupados por cada objeto na área de memória do SGBD. Nesta visão, tudo é visto com um objeto do Oracle, tabelas, índices, etc.

Objetos	Blocos	Objetos	Blocos
ACCESS$	61	I_DEPENDENCY2	19
ATTRCOL$	222	I_FILE#_BLOCK#	5
AURORA$CURRENT$DYN$REG	1	I_FILE2	1
AURORADYNREG	1	I_ICOL1	3
AURORA$STARTUP$CLASSES$	2	I_IDL_CHAR1	26
CCOL$	8	I_IDL_SB41	27
CDEF$	8	I_IDL_UB11	41
CLU$	222	I_IDL_UB21	27
COL$	222	I_JAVASNM1	38
COLTYPE$	222	I_JOB_NEXT	1
C_COBJ#	8	I_KOPM1	1
C_FILE#_BLOCK#	41	I_OBJ#	5
C_OBJ#	222	I_OBJ1	32
C_TS#	14	I_OBJ2	196
C_USER#	3	I_OBJAUTH1	6
DEPENDENCY$	622	I_OBJAUTH2	15
DUAL	2	I_PROCEDURE1	2
FET$	14	I_PROFILE	1
FILE$	1	I_SEQ1	1
ICOL$	222	I_SYN1	21
ICOLDEP$	222	I_SYSAUTH1	3
IDL_CHAR$	57	I_TRIGGER1	1
IDL_SB4$	65	I_TRIGGER2	1
IDL_UB1$	479	I_TRIGGERCOL2	1
IDL_UB2$	75	I_TRIGGERJAVAC	1
ID_GENS$	2	I_TRIGGERJAVAF	1
IND$	222	I_TRIGGERJAVAM	1
INDPART$	1	I_TRIGGERJAVAS	1
I_ACCESS1	64	I_TS#	1
I_CCOL2	4	I_UNDO1	1
I_CDEF3	1	I_USER#	1
I_COBJ#	1	I_USER1	1
I_DEPENDENCY1	85	I_VIEW1	3

continua ⇒

Objetos	Blocos	Objetos	Blocos
JAVA$POLICY$SHARED$TABLE	2	SEQ$	1
JAVASNM$	24	SNS$BINDINGS$	2
JOB$	1	SNS$INODE$	2
KOPM$	1	SNS$NODE_INDEX	1
LIBRARY$	222	SNS$REFADDR$	3
LINK$	1	SNS$REFADDR_INDEX	1
LOB$	222	SYN$	64
LOC$	1	SYSAUTH$	3
MIGRATE$	2	SYSTEM	3
NTAB$	222	SYS_C001012	1
OBJ$	278	TAB$	222
OBJAUTH$	13	TRIGGER$	2
PENDING_TRANS$	1	TRIGGERCOL$	1
PROCEDURE$	3	TRIGGERJAVAC$	2
PROFILE$	2	TRIGGERJAVAF$	2
PROPS$	2	TRIGGERJAVAM$	2
RBS0	3	TRIGGERJAVAS$	2
RBS1	2	TS$	14
RBS2	2	TSQ$	3
RBS3	2	TYPE_MISC$	222
RBS4	2	UET$	41
RBS5	2	UNDO$	1
RBS6	2	USER$	3
REFCON$	222	VIEW$	7
SEG$	41	VIEWTRCOL$	222

Tabela 4: Visão V$CACHE (a)

A tabela 4 ilustra as informações da visão V$CACHE antes de ser carregado um bloco para a memória. O comando SQL utilizado para prover essas informações foi:

- select name, count(*) from v$cache group by name order by 1;

O usuário para executar essa consulta deve possuir o privilégio SYS, que permite acessar informações do sistema.

A partir desse momento foi carregado um chunk para memória. O comando SQL executado foi:

1) select * from tb_molap_chunk_vendas where numero_chunk = 1;

Objetos	Blocos	Objetos	Blocos
ACCESS$	63	I_OBJ2	196
ATTRCOL$	222	I_OBJAUTH1	6
AURORA$CURRENT$DYN$REG	1	I_OBJAUTH2	15
AURORADYNREG	1	I_PROCEDURE1	2
AURORA$STARTUP$CLASSES$	2	I_PROFILE	1
CCOL$	8	I_SEQ1	1
CDEF$	8	I_SYN1	21
CLU$	222	I_SYSAUTH1	3
COL$	222	I_TRIGGER1	1
COLTYPE$	222	I_TRIGGER2	1
C_COBJ#	8	I_TRIGGERCOL2	1
C_FILE#_BLOCK#	50	I_TRIGGERJAVAC	1
C_OBJ#	222	I_TRIGGERJAVAF	1
C_TS#	14	I_TRIGGERJAVAM	1
C_USER#	3	I_TRIGGERJAVAS	1
DEPENDENCY$	622	I_TS#	1
DUAL	2	I_UNDO1	1
FET$	14	I_USER#	1

continua ⇒

Objetos	Blocos	Objetos	Blocos
FILE$	1	I_USER1	1
ICOL$	222	I_VIEW1	3
ICOLDEP$	222	JAVA$POLICY$SHARED$TABLE	2
IDL_CHAR$	62	JAVASNM$	24
IDL_SB4$	70	JOB$	1
IDL_UB1$	484	KOPM$	1
IDL_UB2$	80	LIBRARY$	222
ID_GENS$	2	LINK$	1
IND$	222	LOB$	222
INDPART$	1	LOC$	1
IX_VENDAS_CHUNK	3	MIGRATE$	2
I_ACCESS1	66	MON_MODS$	1
I_CCOL2	4	NTAB$	222
I_CDEF3	1	OBJ$	278
I_COBJ#	1	OBJAUTH$	13
I_CON1	2	PENDING_TRANS$	1
I_DEPENDENCY1	87	PROCEDURE$	3
I_DEPENDENCY2	19	PROFILE$	2
I_FILE#_BLOCK#	5	PROPS$	2
I_FILE2	1	RBS0	3
I_ICOL1	3	RBS1	2
I_IDL_CHAR1	27	RBS2	2
I_IDL_SB41	28	RBS3	2
I_IDL_UB11	41	RBS4	2
I_IDL_UB21	28	RBS5	2
I_JAVASNM1	38	RBS6	2
I_JOB_NEXT	1	REFCON$	222
I_KOPM1	1	SEG$	50
I_OBJ#	5	SEQ$	1
I_OBJ1	32	SNS$BINDINGS$	2
SNS$INODE$	2	TRIGGERJAVAC$	2
SNS$NODE_INDEX	1	TRIGGERJAVAF$	2
SNS$REFADDR$	3	TRIGGERJAVAM$	2
SNS$REFADDR_INDEX	1	TRIGGERJAVAS$	2
SYN$	64	TS$	14
SYSAUTH$	3	TSQ$	3
SYSTEM	3	TYPE_MISC$	222
SYS_C001012	1	UET$	50
TAB$	222	UNDO$	1
TB_MOLAP_CHUNK_VENDAS	**1**	USER$	3
TRIGGER$	2	VIEW$	10
TRIGGERCOL$	1	VIEWTRCOL$	222

Tabela 5: Visão V$CACHE (b)

Com a seleção de registros na tabela de fatos, é possível observar que a tabela TB_MOLAP_CHUNK_VENDAS passou a figurar na lista de objetos da visão V$CACHE.

O próximo passo foi selecionar um registro do chunk e constatar que não era carregado outro bloco para a memória, ou seja, a informação estava sendo recuperada direto da memória principal. O comando SQL utilizado para prover essas informações foi:

- select * from tb_molap_chunk_vendas where pk_cube = 1 and PK_OFFSET = 150;

Observe na tabela 6, que é utilizado o índice IX_VENDAS_CHUNK e que a tabela TB_MOLAP_CHUNK_VENDAS mantém a mesma página na memória.

Objetos	Blocos	Objetos	Blocos
ACCESS$	63	IDL_SB4$	70
ATTRCOL$	222	IDL_UB1$	484
AURORA$CURRENT$DYN$REG	1	IDL_UB2$	80
AURORADYNREG	1	ID_GENS$	2
AURORA$STARTUP$CLASSES$	2	IND$	222
CCOL$	8	INDPART$	1
CDEF$	8	**IX_VENDAS_CHUNK**	**1**
CLU$	222	I_ACCESS1	66
COL$	222	I_CCOL2	4
COLTYPE$	222	I_CDEF3	1
C_COBJ#	8	I_COBJ#	1
C_FILE#_BLOCK#	50	I_CON1	2
C_OBJ#	222	I_DEPENDENCY1	87
C_TS#	14	I_DEPENDENCY2	19
C_USER#	3	I_FILE#_BLOCK#	5
DEPENDENCY$	622	I_FILE2	1
DUAL	2	I_ICOL1	3
FET$	14	I_IDL_CHAR1	27
FILE$	1	I_IDL_SB41	28
ICOL$	222	I_IDL_UB11	41
ICOLDEP$	222	I_IDL_UB21	28
IDL_CHAR$	62	I_JAVASNM1	38
I_JOB_NEXT	1	PROCEDURE$	3
I_KOPM1	1	PROFILE$	2
I_OBJ#	5	PROPS$	2
I_OBJ1	32	RBS0	3
I_OBJ2	196	RBS1	2
I_OBJAUTH1	6	RBS2	2
I_OBJAUTH2	15	RBS3	2
I_PROCEDURE1	2	RBS4	2
I_PROFILE	1	RBS5	2
I_SEQ1	1	RBS6	2
I_SYN1	21	REFCON$	222
I_SYSAUTH1	3	SEG$	50
I_TRIGGER1	1	SEQ$	1
I_TRIGGER2	1	SNS$BINDINGS$	2
I_TRIGGERCOL2	1	SNS$INODE$	2
I_TRIGGERJAVAC	1	SNS$NODE_INDEX	1
I_TRIGGERJAVAF	1	SNS$REFADDR$	3
I_TRIGGERJAVAM	1	SNS$REFADDR_INDEX	1
I_TRIGGERJAVAS	1	SYN$	64
I_TS#	1	SYSAUTH$	3
I_UNDO1	1	SYSTEM	3
I_USER#	1	SYS_C001012	1
I_USER1	1	TAB$	222
I_VIEW1	3	**TB_MOLAP_CHUNK_VENDAS**	**1**
JAVA$POLICY$SHARED$TABLE	2	TRIGGER$	2
JAVASNM$	24	TRIGGERCOL$	1
JOB$	1	TRIGGERJAVAC$	2
KOPM$	1	TRIGGERJAVAF$	2
LIBRARY$	222	TRIGGERJAVAM$	2
LINK$	1	TRIGGERJAVAS$	2
LOB$	222	TS$	14
LOC$	1	TSQ$	3
MIGRATE$	2	TYPE_MISC$	222
MON_MODS$	1	UET$	50
NTAB$	222	UNDO$	1
OBJ$	278	USER$	3
OBJAUTH$	13	VIEW$	10
PENDING_TRANS$	1	VIEWTRCOL$	222

Tabela 6: Visão V$CACHE (c)

Com isso fica, provado que um chunk ocupa uma página de disco e que uma vez presente na memória principal, os seus dados também são acessados em memória principal.

Apresenta-se na figura 26, os gráficos com as médias dos tempos de execução, em segundos, de cada tipo de consulta submetida onde é possível analisar isoladamente estas consultas realizadas.

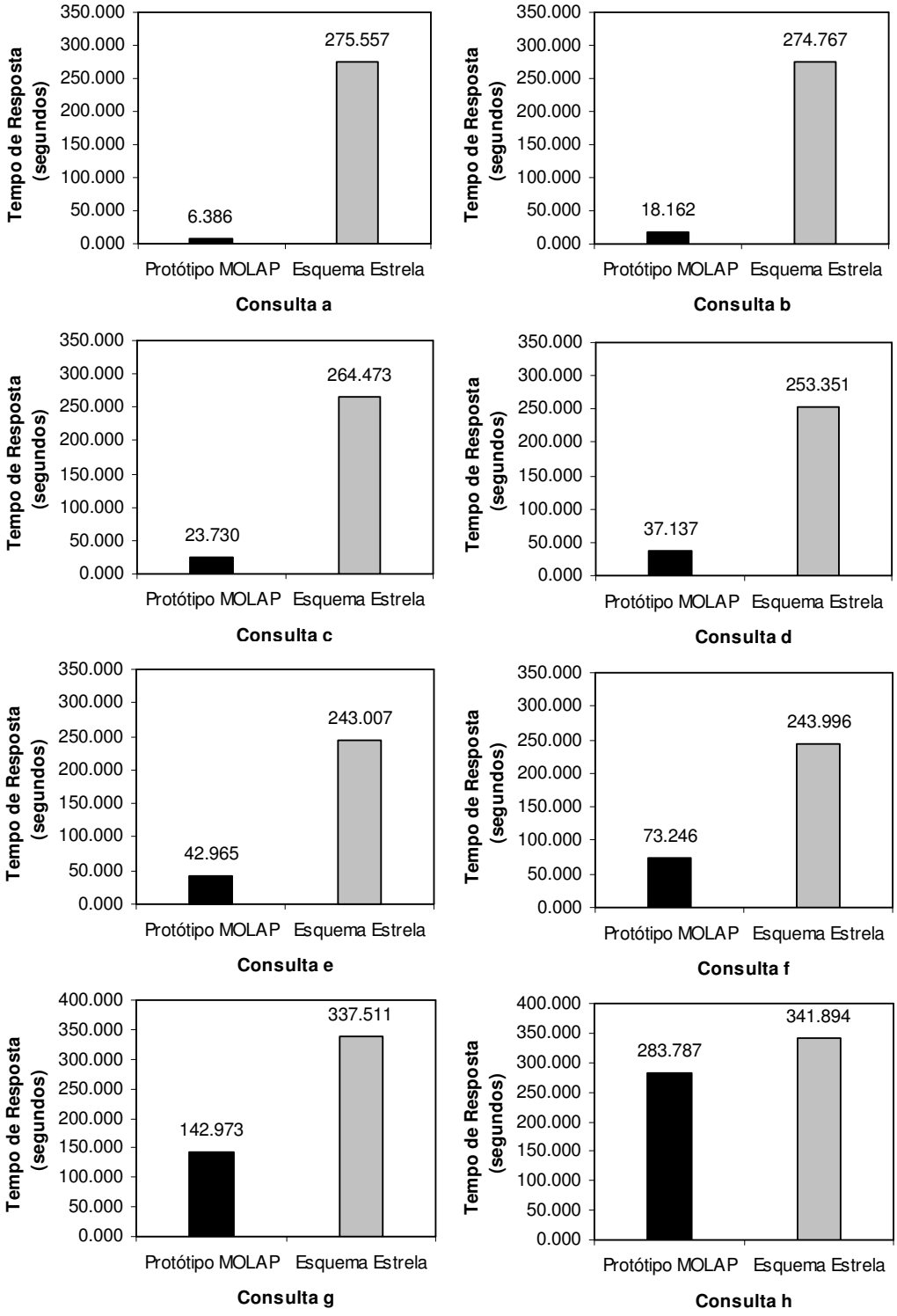

Figura 26: Média dos tempos de execução das consultas

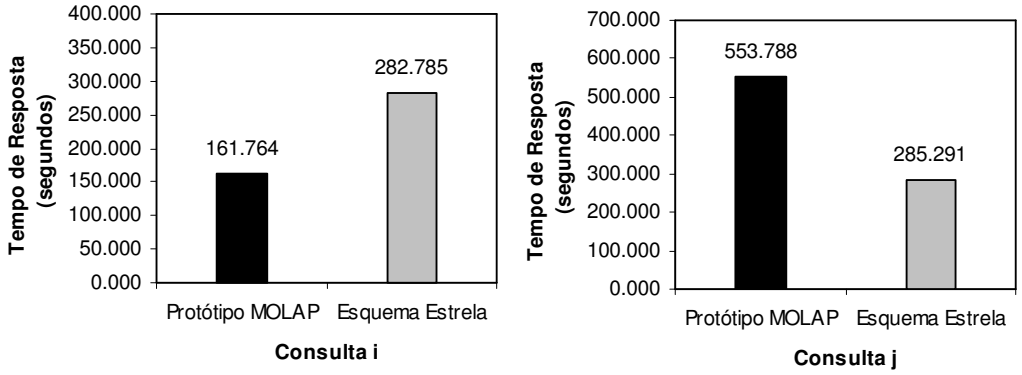

Figura 26: Média dos tempos de execução das

Apresenta-se na figura 27 um outro tipo de gráfico com as mesmas médias dos tempos de execução, também em segundos, das consultas submetidas onde é possível analisar o comportamento assumido pelo Protótipo MOLAP e pelo Esquema Estrela durante a execução das consultas.

Em seguida, é apresentada uma discussão desses resultados.

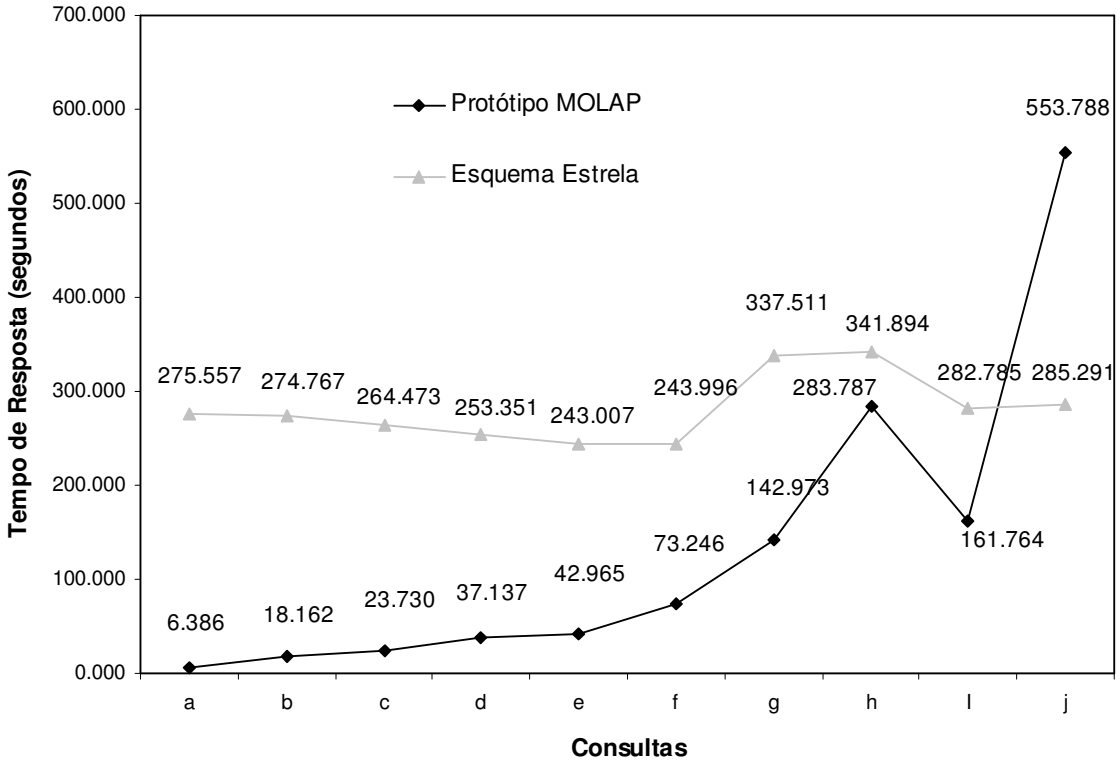

Figura 27: Comportamento do Protótipo MOLAP e do Esquema Estrela diante da execução das consultas

Analisando-se os resultados ilustrados na figura 26, é possível constatar que se a consulta executada obedece aos padrões de acesso definidos (consultas *a*, *b*, *d*, *f*), o protótipo MOLAP se mostra mais eficiente do que o Esquema em Estrela.

Ainda analisando-se os resultados ilustrados na figura 26, é possível constatar que se a consulta não obedece aos padrões de acesso definidos (consultas *c*, *e*), porém, sendo uma consulta que não apresente semântica muito diferente dos padrões de acesso, o protótipo MOLAP ainda se mostra mais eficiente do que o Esquema em Estrela, apesar das diferenças de tempos já se aproximarem.

Já os resultados ilustrados ainda na figura 26 mostram que se a consulta não obedece aos padrões de acesso definidos (consultas *g*, *h*) e apresenta semântica muito diferente dos padrões de acesso, o Protótipo MOLAP não se mostra tão eficiente em relação ao Esquema em Estrela. Apesar do tempo de resposta destas consultas ter sido inferior no Protótipo MOLAP, nota-se que o desempenho do protótipo não se mostra tão eficiente quanto apresentado nas consultas *a* até *f*.

Outra análise interessante de se fazer é sobre os resultados das consultas *i* e *j* ilustrados também na figura 26. A semântica destas consulta é muito diferente das demais consultas, pois é realizada uma operação de agrupamento. Neste contexto o Protótipo MOLAP não se mostra eficiente em relação ao Esquema em Estrela. Percebe-se que o tempo de resposta das consultas tem um aumento muito grande ao se comparar com o tempo de resposta das consultas realizadas sem esta operação (consultas *a* até *h*). Percebe-se também que o Esquema em Estrela mantém seu tempo de resposta relativamemete constante para consultas com esta semântica.

Analisando-se os resultados ilustrados na tabela apresentada no apêndice E, é possível verificar que tanto o Esquema em Estrela quanto o protótipo MOLAP, mostram-se mais eficientes em relação aos resultados da execução de suas consultas anteriores, ou seja, a medida que uma consulta é executada mais de uma vez, o seu tempo de resposta subseqüente vai diminuindo. Isto se deve ao fato das informações das consultas já estarem em "cache". Com

relação aos tempos de execução de ambos, não se percebe muita alteração dos quadros analisados nos parágrafos anteriores.

A não utilização da operação de "join" no algoritmo utilizado pelo protótipo para efetuar a recuperação dos dados da tabela de fatos contribuiu de forma decisiva para que os tempos de resposta das consultas realizadas no Protótipo MOLAP se mostrassem superiores ao Esquema em Estrela.. Como o algoritmo utilizado é baseado em valor (abscissas do cubo), ele não necessita efetuar esta operação entre as tabelas. Ao contrário, o Esquema em Estrela faz uso da operação de "join" entre os elementos das tabelas dimensões e os elementos da tabela de fatos. A operação "join" é uma das operações que mais oneram recursos de CPU para recuperação de registros em tabelas [4].

Outro fator que teve um papel crucial para o melhor desempenho do Protótipo MOLAP, foi a utilização dos padrões de acesso, como ilustrado na figura 26 (consultas *a*, *b*, *d*, *f*). O propósito, já discutido na seção 2.5.2, da utilização dos padrões de acesso mostrou-se eficiente conforme comprovam os resultados. A organização dos dados, favorecendo os tipos de consultas mais relevantes para o negócio, influenciou de maneira positiva os tempos de resposta. Enquanto o protótipo MOLAP possui os dados dispostos fisicamente de maneira a minimizar o tempo de resposta da classe de consultas mais relevantes para o negócio, o esquema em estrela, em princípio, dispõe seus dados de maneira seqüencial temporal, ou seja, à medida que os dados são inseridos na tabela de fatos, eles ficam dispostos de forma crescente em relação ao tempo, favorecendo a ordem de uma das dimensões. Sob este aspecto, vale ressaltar ainda que a utilização dos chunks permite que os dados sejam tratados de maneira uniforme, ou seja, não há privilégio de acesso para nenhuma dimensão.

Já os resultados ilustrados na figura 26 (consultas *g*, *h*, *i*, *j*) mostram uma diminuição na eficiência do protótipo. Um fator que contribuiu para isto foi a realização de consultas que não obedecem aos padrões de acesso e possuem semântica muito diferente dos padrões. Neste caso, como os dados estão dispostos segundo os padrões de acesso, então era de se esperar que o tempo de resposta das consultas realizadas no protótipo não fossem tão eficientes quanto ao tempo de resposta das consultas realizadas no Esquema em Estrela.

Os tempos, por sinal, foram muito superiores, ao comparar-se com o tempo das consultas *a* até *f*, o que nos reporta a outro questionamento. Como o protótipo utiliza cursores para o armazenamento dos elementos das dimensões das consultas e para realização do produto cartesiano, pode-se inferir também que quanto mais abrangente for a consulta, menos eficiente se torna a utilização do protótipo. Entretanto, para comprovar se essa tendência realmente se confirma, é necessária a realização de testes com um conjunto de dados maior e com a redefinição dos padrões de acesso para se ajustar a consultas com maior abrangência de informações a serem analisadas. Assim, realmente poderá ser detectado se a utilização de cursores é menos eficiente quando se trata de um conjunto maior de dados.

Esta inferência fica mais evidente analisando-se o gráfico ilustrado na figura 27. Como se pode perceber, os tempos de resposta do Esquema em Estrela mantém-se relativamente constantes, enquanto que no Protótipo MOLAP nota-se um aumento destes tempos de maneira proporcional a quantidade de registros recuperados pelas consultas.

A primeira conclusão que se pode tirar é que o algoritmo implementado no protótipo não apresenta técnicas de otimização para realização de consultas que necessitem da operação de agrupamento, face aos tempos de resposta apresentados nas consultas *i* e *j*. Enquanto o algoritmo de otimização de consultas do SGBD atende as consultas submetidas ao Esquema em Estrela, não se percebe o mesmo para as consultas submetidas ao Protótipo MOLAP, visto que, os tempos de resposta aumentam muito e de maneira proporcional a quantidade de registros recuperados pelas consultas.

A outra conclusão é que, com o conjunto de dados utilizado, os padrões de acesso definidos, a utilização dos conceitos de chunk e a realização de consultas que obedeçam os padrões de acesso sem a operação de agrupamento, o protótipo MOLAP mostra-se eficiente, em média, na realização de consultas OLAP.

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 Conclusões

Neste trabalho foi especificada e implementada uma solução que possibilita a realização de consultas OLAP, pautada na tecnologia MOLAP. Como repositório de dados foi utilizado o SGBD-OR Oracle 8i, que possui recursos da tecnologia objeto-relacional, o que possibilitou a implementação dos conceitos das técnicas de chunk, empregadas para simular o comportamento de um array multidimensional em SGBD's-OR.

A pesquisa tratou no que tange à construção de esquemas que utilizam a tecnologia MOLAP em um SGBD-OR de ampla aceitação no mercado. O desafio foi construir um protótipo MOLAP que, através de um esquema de metadados, estivesse apto a realizar consultas OLAP de forma eficiente.

Existem alguns estudos similares ao deste trabalho, entretanto, não é de nosso conhecimento que técnicas para simulação de arrays multidimensionais em SGBD's-OR tenham sido aplicadas no topo do SGBD-OR Oracle 8i.

Como primeiro resultado, temos como sugestão um conjunto de passos que podem ser seguidos para a construção de um esquema MOLAP no SGBD-OR Oracle 8i, que se encontram detalhados no capítulo 3.

Devido ao tamanho e à complexidade de prover todas as funcionalidades de um Servidor OLAP, delimitamos o escopo deste trabalho no que diz respeito à implementação de algumas funções para recuperação de dados no esquema MOLAP, visando atender a consultas gerenciais. As etapas de extração, transformação, carga e atualização, não são tratadas neste trabalho.

As técnicas utilizadas na concepção do protótipo foram estritamente focadas na redução do tempo de resposta das consultas OLAP submetidas. Isto influenciou diretamente na disposição física dos dados. Isto se baseia na identificação dos tipos de

consulta que agregam mais valor ao negócio cujo esquema está representando. Apoiando-nos nesta premissa, definimos alguns tipos de consultas neste trabalho, referenciadas como padrões de acesso, e atribuímo-lhes suas probabilidades de ocorrerem. Estas probabilidades representam o percentual de realização desses tipos de consulta para o negócio.

Com relação ao armazenamento dos dados enfrentamos um problema comum quando se trabalha com arrays multidimensionais, que é a esparsidade. Para contornar este problema utilizamos uma técnica de compressão que faz com que sejam armazenados apenas dados válidos, ou seja, todos os elementos do array *n*-dimensional realmente correspondem ao registro de um fato ocorrido no ambiente operacional, com isso eliminando o problema de esparsidade.

Com a adoção das técnicas de chunk, provemos um mecanismo de acesso uniforme aos dados do array. Queremos dizer que, não importa qual seja a ordem da elaboração da consulta, os dados são acessados sem privilégio de nenhuma dimensão, ou seja, o tratamento dispendido à realização das consultas é independente da ordem de combinação das dimensões nas consultas. O que passa a ser privilegiado são os tipos que consultas que atendem aos padrões de acesso estabelecidos.

A construção deste protótipo propiciou uma experiência prática no uso da tecnologia objeto-relacional em um ambiente de DWing. Podemos atestar que os recursos objeto-relacionais empregados na constução do protótipo não só facilitaram a implementação de um DW como tornaram possível essa construção.

Como constatação da viabiliadade da utilização do SGBD-OR Oracle 8i para projetos de DWs, utilizando-se a tecnologia MOLAP, foram realizadas vários tipos de consultas OLAP, sendo que o protótipo mostrou-se eficiente em relação ao tempo de resposta destas consultas ao ser comparado com a técnica do esquema em estrela, utilizada pela comunidade de DWing para projetar DWs.

Finalmente, a solução proposta conseguiu atender aos requistos elicitados de forma que os objetivos propostos foram alcançados. Sendo assim, o *protótipo MOLAP* criado pode vir a ser uma alternativa para projetos de DWs.

5.2 Trabalhos futuros

Para dar continuidade a este trabalho são sugeridos, com o intuito de aperfeiçoar tanto a especificação quanto a implementação do protótipo MOLAP, os seguintes estudos:

- Criação de uma linguagem com primitivas de consulta apoiadas em um modelo multidimensional conceitual e formal. Através do modelo formal, pode ser gerada uma interface comum para consultas OLAP no protótipo.
- Construção de uma interface gráfica mais amigável para realização das consultas. Trata-se de um requisito importante para os gerentes de negócio poderem realizar, de maneira fácil e intuitiva, as consultas para tomada de decisões.
- Estender o otimizador de consultas do Oracle 8i e instruí-lo a gerar planos de consultas mais eficientes para as funções implementadas no protótipo.
- Implementação de mais funções de agregação como “count”, “average”, etc. Com isso, estarão sendo incoporados mais “poder” de realização de consultas OLAP pelo protótipo.
- Implementação da técnica IndexToIndex [4], que permitirá que o protótipo possa navegar sobre os agregados do DW.
- Tratamento de atualizações no DW, não contempladas neste trabalho. É de suma importância que seja fornecido um mecanismo para a manutenção do DW utilizado no protótipo. Isso possibilitará consolidar o protótipo MOLAP como uma ferramenta de apoio à consultas OLAP.
- Reformulação dos chunks, pois isto não aparece de forma dinâmica. Os dados sempre são distribuídos de acordo a um histórico de consultas prévio à sua construção. Como aproveitar novamente o histórico de consultas para reconfigurar a distribuição dos dados?

Referências e Bibliografia

[1] CHAUDHURI, S.; DAYAL, U. An Overview of Data Warehousing and OLAP Technology. **ACM SIGMOD Record**, v. 26, n. 1, p. 65-74,1997.

[2] INMON, W.H. **Como Construir o Data Warehouse**. 2. ed. Rio de Janeiro: Campus, 1997. 387 p.

[3] STONEBRAKER, M.; BROWN, P. **Object Relacional DBMS:** Tracking The next Great Wave. 2nd ed. Morgan Kaufmann, 1999. 297 p.

[4] ZHAO, Y. et al. Array-Based Evaluation of Multi-Dimensional Queries in Object-Relational Database Systems. **IEEE Computer**, Orlando, p. 241-249, 1998. Proceedings of the Fourteenth International Conference on Data Engineering

[5] CODD, E.F.; CODD, S.B.; SALLEY, C.T. Beyond Decision Support. **Computerworld**, v. 27, n. 30, p. 87-89, 1993.

[6] PILOT SOFTWARE. **An Introdution OLAP.** Contém informações institucionais, técnicas, notícias, projetos, publicações e serviços. Disponível em: <http://www.pilotsw.com/r_and_t/whtpaper/olap/olap.html>. Acesso em: 15 dez. 2000.

[7] ORACLE CORPARATION. Oracle OLAP Products: Adding Value to the Data Warehouse, **Oracle White Paper**, n. C10281, p. 1-22, 1995, Disponível em: <http://otn.oracle.com/products/exp_server/pdf/olapwp.pdf>. Acesso em: 2 dez. 2000.

[8] SARAWAGI, S.; STONEBRAKER, M. Efficient Organization of Large Multidimensional Arrays. **IEEE Computer**, Washington, p. 328-336, 1994. Proceedings of the Tenth International Conference on Data Engineering

[9] WIEDERHOLD, G. Mediators in the Architecture of Future Information Systems. **IEEE Computer**, Los Alamitos, v. 25, n. 3, p.38-49, 1992.

[10] POE, V.; KLAUER, P.; BROBST, S. **Building a Data Warehouse for Decision Support**. 2nd ed. Prentice Hall PTR, 1997. 285 p.

[11] PARSAYE, K. The four Spaces of Decision Support. **DBMS Magazine**, p.25, 1996.

[12] KIMBALL, R. A Dimensional Modeling Manifesto. **DBMS Magazine**, 1997, Disponível em: <http://www.dbmsmag.com/9708d15.html>. Acesso em: 13 jul. 2001

[13] ZHAO, Y.; DESHPANDE, M. P.; NAUGHTON, N. J. An Array-Based Algorithm for Simultaneous Multidimensional Aggregates. **ACM Press**, New York, p. 159-170, 1997. Proceedings of the 1997 ACM SIGMOD international conference on Management of Data

[14] AGRAWAL R.; GUPTA A.; SARAWAGI S. Modeling Multidimensional Databases. **IEEE Computer**, Washington, p. 232-243, 1997. Proceedings of the Thirteenth International Conference on Data Engineering. Disponível em: <http://www.almaden.ibm.com/software/quest/Publications/papers/icde97_olap.pdf>. Acesso em: 15 jan. 2001.

[15] GRAND, M. **Patterns in Java:** A Catalog of Reusable Design Illustrated with UML. 2nd ed. Wisley, 2002. 544 p. 1v.

[16] FOWLER, M. et al. **Refactoring improving the design of existing code**. 1st ed. Addison-Wesley,1999. 431 p

[17] BECK, K.; GAMMA, E. Test infected: Programmers Love Writing Tests. Contém informações técnicas. Disponível em: <http://junit.sourceforge.net/doc/testinfected/testing.htm>. Acesso em: 23 nov. 2001

[18] BOOCH, G.; RUMBAUCH, J.; JACOBSON, I. **The Unified Modeling Language Reference Manual.** 1st ed. Addison Wesley, 1999. 550 p.

[19] FOWLER, M.; KENDALL, S. **UML Distilled:** A Brief Guide to the Standard Object Modeling Language. 2nd ed.Addison Wesley, 1999. 192 p

[20] BOOCH, G.; RUMBAUCH, J.; JACOBSON, I. **The Unified Software Development Process.** 1st. ed. Addison Wesley, 1999. 463 p.

[21] POLLICE, G. Using the IBM Rational Unified Process for Small Projects: Expanding Upon Extreme Programming. **IBM Rational** , 2000. Disponível em <http://www-106.ibm.com/developerworks/rational/library/409.html>. Acesso em: 18 ago. 2001

[22] ALMEIDA, R.R.; SOUZA, A.D.D. JConfig: Pacote de Configuração de Javabeans em XML, 2000. Disponível em: <//http://vulcano.dsc.ufpb.br/rodrigor/jconfig/index.html> . Acesso em 27 mar. 2001.

[23] OMG. OBJECT MANAGEMENT GROUP Unified Modeling Language Specification, v. 1.3, 1997-2001. Disponível em: <http://www.omg.org/technology/documents/formal/uml.htm>. Acesso em 05 jan. 2001.

[24] FURLAN, J. D. **Modelagem de Objetos Através de UML.** Pearson Education do Brasil, 1998. 344 p.

[25] GAMMA, E. et al. **Design Patterns:** Elements of Resusable Object-Oriented Software. 1st ed. Addison Wesley, 1995. 395 p.

[26] ORACLE. Oracle8i Reference Manual. v. 8.1.5, n. A67790-01, 1999.

[27] FIRESTONE, J. M. Evaluating OLAP Alternatives. White Paper, n. 4, 1997. Disponível em: <http://www.dkms.com/papers/olapalt.pdf>. Acesso em: 15 out. 2001.

[28] GUPTA, V.R. An Introduction to Data Warehousing. White Paper, System Services, 1997. Disponível em <http://system-services.com/dwintro.asp>. Acesso em: 08 jan. 2001.

[29] HOROWITZ, E.; SAHNI S. **Fundamentals of Data Structures.** 1st ed. New York: Computer Science Press, 1976

[30] KIMBALL, R.; ROSS M. **The Data Warehouse Toolkit.** 1st ed. Wiley, 1998. 464 p

[31] KIMBALL, R. et al. **The Data Warehouse Lifecycle Toolkit:** Experts Methods for Designing, Developing and Deploying Data Warehouses. Wiley , 1998. 800 p.

[32] SAMPAIO, M.C. Curso: Data Warehousing. In: SEMANA DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DA BAHIA, 1998, Salvador, **Cursos**, Salvador: 1998.

[33] THOMSEN, E. **OLAP Solutions:** Building Multidimensional Information Systems. Wisley, 1997. 688 p.

[34] UNIVERSIDADE FEDERAL DA PARAÍBA. SAUVÉ, J. P. Departamento de Sistemas e Computação. Notas de aula das disciplinas de Análise e Projeto OO e Métodos Avançados de Programação. COPIN, 2000. Disponível em: <http://vulcano.dsc.ufpb.br/jacques/cursos/1999.2/map/material/map2.htm>, <http://vulcano.dsc.ufpb.br/jacques/cursos/1999.1/apoo/material/apoo2.htm>. Acesso em 10 abr. 2000.

[35] UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. Biblioteca da Faculdade de Agronomia. Normas.doc: normas para elaboração de referências bibliográficas. Porto Alegre, 31 out. 2000. Disponível em: <http://wwwsr.unijui.tche.br/ambienteinteiro/Manual.pdf>. Acesso em 10 jan. 2002.

Apêndice A

A.1 Engenharia de Software: algumas técnicas modernas

Este capítulo apresenta algumas tecnologias que são utilizadas para o desenvolvimento do tema proposto e que são abordadas ao longo do documento: o Rational Unified Processo (RUP), que nos descreve um conjunto de passos para se desenvolver um bom software, juntamente com a "Unified Modeling Language" (UML), que nos possibilita modelar software orientado a objetos. São utilizadas técnicas avançadas de desenvolvimento e projeto de sistemas aqui representados pelo conceito Padrões de Projeto.

A.2 UML(Unified Modeling Language)

Quando o assunto em pauta é desenvolvimento de sistemas, vem à tona uma palavra-chave *cliente*. Satisfazer suas expectativas, procurar apresentar-lhes sempre boas soluções; fazer o que eles pedem no prazo com eles combinado, etc. Estes são apenas alguns aspectos que estão envolvidos na relação desenvolvedor de sistemas e cliente e está claro que a primitiva máxima é deixar o cliente satisfeito.

Para que a tarefa de desenvolver sistemas seja bem sucedida, é preciso que haja uma grande harmonia e eficiência na utilização dos recursos, recursos estes que envolvem tanto os aspectos materiais quanto os humanos.

Uma gama de tecnologias é disponibilizada ao desenvolvedor de sistemas para que ele maximize a qualidade de seus serviços. Uma dessas tecnologias é a UML. Mas antes de discorrer sobre a UML, é importante discutir um aspecto que se faz necessário: antes de entender o que é a UML, é importante que se saiba porque ela surgiu.

Um bom software é aquele que satisfaz e atende às necessidades do cliente, portanto o cliente não está interessado em modelos, em código fonte, em documentos, ele quer que seu software funcione. Entretanto, não é possível conceber um bom sistema sem que todos estes aspectos apresentados sejam muitos bem trabalhados. E é diante deste contexto que surge a modelagem de sistemas.

Então surge a questão: Por que modelar um sistema?

A finalidade de um modelo é prover um melhor entendimento do sistema que está sendo desenvolvido. Um modelo captura aspectos do mundo real e provê abstrações que simplificam o entendimento desta realidade.

Grady Booch, James Rumbauch e Ivar Jacobson em [18] definem que um modelo:

- ajuda a visualizar um sistema como ele é ou como nos queremos que ele seja;
- permite que especifiquemos a estrutura e comportamento do sistema;
- guia-nos durante a construção do sistema;
- Documenta as decisões que temos a tomar.

Em software, existem várias maneiras de se representar um modelo. As duas principais são: a perspectiva de algoritmos e a perspectiva de objetos, pautada na perspectiva de objetos na qual a UML está fundamentada.

A UML foi projetada para facilitar o entendimento das notações e conceitos da perspectiva de objetos e surgiu através da fusão de três grandes notações de modelagem. São elas: notação de Booch, a "Object Management Group(OMT)" ou notação de Rumbaugh e a Jacbson's "OO Software Engineering(OOSE)".

Visto o porquê da UML, podemos agora definir melhor o que ela vem a ser.

Segundo [18]: "A UML é uma linguagem padrão para modelar sistemas". Ela pode ser usada para visualizar, especificar, construir e documentar as particularidades do sistema.

A UML é chamada de linguagem de modelagem, não processo de desenvolvimento. Linguagem de modelagem é um tipo de notação, principalmente gráfica, que serve para produzir abstrações do mundo real. O processo de desenvolvimento define que passos o projeto seguirá.

Os três pesquisadores [18] também criaram um processo de desenvolvimento que ficou conhecido como Rational Unified Process(RUP). UML e RUP são

totalmente distintos, podendo ser usados separadamente. Entretanto, procurando extrair mais benefícios da UML, este projeto fará uso do RUP[20].

O Rational Unified Process é composto das seguintes partes:

- "Use case driven";
- "Architecture-centric";
- "Interative and Incremental".

Use case driven: significa que os use são usados empregados estabelecer o comportamento do sistema, verificar e validar a arquitetura do sistema. Os use case existem durante todas as fases do projeto. Na fase de análise os use cases capturam toda a funcionalidade do sistema e são validados junto ao usuário. Nas fases de projeto e teste os use case são realizados e verificam o sistema respectivamente.

Architecture-centric: significa que o sistema é desenvolvido a partir de uma arquitetura. É importante definir a arquitetura cedo e refiná-la com o tempo. A arquitetura vai definir as diferentes partes do sistema, bem como seus relacionamentos, e as interações.

Interactive and Incremental: em um processo iterativo o desenvolvimento é uma seqüência de etapas, cada etapa adicionando mais funcionalidades, detalhes. Toda iteração é varrida por todas as fases do processo(análise, projeto, implementação e testes). Cada iteração produz uma versão mais funcional do sistema. É importante que se defina cada incremento com a menor funcionalidade possível e, paulatinamente, vá-se adicionando funcionalidades em cada incremento.

A.3 Design Patterns

A Engenharia de Software é uma área da computação que tem como atividade fim a construção de software. Este processo de construção é concebido devido a um conjunto de procedimentos, métodos e ferramentas capazes de especificar, documentar, projetar, desenvolver e realizar manutenção sobre o software.

Apesar de existir uma área voltada apenas para o desenvolvimento de software, construir softwares de qualidade é uma tarefa difícil, lenta e cara. Fazer com que as soluções nele empregadas se tornem reusáveis é mais difícil ainda.

A solução mais freqüentemente apontada para sanar esse problema, é o reuso. Contudo, o reuso não acontece automaticamente, pois conceber boas idéias, boas abstrações, soluções genéricas, não é nada fácil. O que se verificou ao longo dos anos foi que a idéia do reuso ainda não estava sedimentada o suficiente no meio dos desenvolvedores de sistema. No entanto, na metade da década de 90 as lições bem sucedidas e a idéia do reuso se cristalizaram com o aparecimento e catalogação de soluções de projeto dos melhores projetista de software, o que ficou conhecido como Padrões de Projeto.

Uma definição genérica para o que vem a ser um padrão segundo Cristopher Alexander em [25] é: "Cada padrão descreve um problema que ocorre freqüentemente e então descreve o cerne da solução ao problema de forma a poder reusar a solução milhões de vezes em situações diferentes"

Com os Padrões de Projeto foi possível constatar que o reuso de idéias acontecia, não de código, desta forma uma mesma solução poderia ser reutilizada em sistemas que vislumbrassem mundos totalmente distintos.

Os Padrões de Projeto consistem de micro-arquiteturas de classes, objetos, seus papéis e suas colaborações e são formados basicamente por quatro elementos [25]:

- um nome;
- o problema;
- a solução;
- as conseqüências

Nome: descreve o problema de projeto, suas soluções e conseqüências em poucas palavras; permite projetar num nível mais alto de abstração; permite falar com outros sobre soluções e documentar código, já que os nomes de padrões estão ficando muito difundidos.

O problema: descreve quando aplicar o padrão; descreve o problema e o contexto; pode descrever problemas específicos de projetos; Pode descrever estruturas de objetos ou classes que são sintomas de um projeto inflexível.

A solução: descreve os elementos constituintes do projeto, seus relacionamentos, responsabilidades e colaborações. A solução não descreve um projeto ou implementação concreta porque um padrão é um gabarito de solução para várias situações.

As conseqüências: os resultados e o custo/benefício da aplicação do padrão dizem respeito ao custo/benefício de espaço, tempo, flexibilidade, extensibilidade, portabilidade.

Apêndice B

B.1 Metadados do Protótipo: DTD e XML

Este anexo apresenta a definição dos arquivos DTD (estrutura) e XML (dados) que compõem o metadados do protótipo. O arquivo XML representa o esquema conceitual do DW Vendas apresentado no Capítulo 3, seção 3.5.

ConfigMolap.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT DW (dw)>
<!ELEMENT dw (nome,login, senha, ip, porta, sid, dimensoes, cubos)>
<!ELEMENT dimensoes (dimensao+)>
<!ELEMENT dimensao (nome, tabela, atributos, arvoreb, indextoindex)>
<!ELEMENT atributos (atributo+)>
<!ELEMENT atributo (nome, tipo, campo)>
<!ELEMENT arvoreb (nome, atributos)
<!ELEMENT indextoindex (nome, tabela, atributos)
<!ELEMENT cubos (cubo)>
<!ELEMENT cubo (nome, tabela, atributos)>
<!ELEMENT medida (#PCDATA)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT tipo (#PCDATA)>
<!ELEMENT campo (#PCDATA)>
<!ELEMENT tabela (#PCDATA)>
<!ELEMENT login (#PCDATA)>
<!ELEMENT senha (#PCDATA)>
<!ELEMENT ip (#PCDATA)>
<!ELEMENT porta (#PCDATA)>
<!ELEMENT sid (#PCDATA)>
```

ConfigDWMolap.xml

```
<ConfigDWVendas>
        <dw>
                <nome>Vendas</nome>
                <login>molap</login>
                <senha>molap</senha>
                <ip>127.0.0.1</ip>
                <porta>1521</porta>
                <sid>LSI</sid>
                <dimensoes>
                        <dimensao>
                                <nome>Tempo</nome>
                                <tabela>Molap.Tempo</tabela>
                                <atributos>
                                        <atributo>
                                                <nome>ID_Tempo</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>id_tempo</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>dia</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>dia</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>mes</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>mes</campo>
```

```
                                </atributo>
                                <atributo>
                                        <nome>Ano</nome>
                                        <tipo>NUMERIC</tipo>
                                        <campo>ano</campo>
                                </atributo>
                                <atributo>
                                        <nome>Data</nome>
                                        <tipo>TEXT</tipo>
                                        <campo>data</campo>
                                </atributo>
                                <atributo>
                                        <nome>MesAno</nome>
                                        <tipo>TEXT</tipo>
                                        <campo>mesano</campo>
                                </atributo>
                                <atributo>
                                        <nome>Entrada_cubo</nome>
                                        <tipo>NUMERIC</tipo>
                                        <campo>entrada_cubo</campo>
                                </atributo>
                        </atributos>
                        <arvoreb>
                                <nome>IDX4_Tempo</nome>
                                <atributos>
                                        <atributo>
                                                <nome>Id_tempo</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>id_tempo</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>Entrada_cubo</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>entrada_cubo</campo>
                                        </atributo>
                                </atributos>
                        </arvoreb>
                        <indextoindex>
                                <nome>IndexToIndexTempo</nome>
                                <tabela>Molap.IndexToIndexTempo</tabela>
                                <atributos>
                                        <atributo>
                                                <nome>ID_Tempo</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>id_tempo</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>Identificador_Mes</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>identificador_mes</campo>
                                        </atributo>
                                </atributos>
                        </indextoindex>
                </dimensao>
                <dimensao>
                        <nome>Produto</nome>
                        <tabela>Molap.Produto</tabela>
                        <atributos>
                                <atributo>
                                        <nome>ID_Produto</nome>
                                        <tipo>NUMERIC</tipo>
                                        <campo>id_produto</campo>
                                </atributo>
                                <atributo>
                                        <nome>Nome</nome>
                                        <tipo>TEXT</tipo>
                                        <campo>nome</campo>
                                </atributo>
                                <atributo>
                                        <nome>Categoria</nome>
                                        <tipo>TEXT</tipo>
                                        <campo>categoria</campo>
                                </atributo>
```

```
                                        <atributo>
                                                <nome>Familia</nome>
                                                <tipo>TEXT</tipo>
                                                <campo>familia</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>Entrada_cubo</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>entrada_cubo</campo>
                                        </atributo>
                                </atributos>
                                <arvoreb>
                                        <nome>IDX3_Produto</nome>
                                        <atributos>
                                                <atributo>
                                                        <nome>Id_produto</nome>
                                                        <tipo>NUMERIC</tipo>
                                                        <campo>id_produto</campo>
                                                </atributo>
                                                <atributo>
                                                        <nome>Entrada_cubo</nome>
                                                        <tipo>NUMERIC</tipo>
                                                        <campo>entrada_cubo</campo>
                                                </atributo>
                                        </atributos>
                                </arvoreb>
                                <indextoindex>
                                        <nome>IndexToIndexProduto</nome>
                                        <tabela>Molap.IndexToIndexProduto</tabela>
                                        <atributos>
                                                <atributo>
                                                        <nome>ID_Produto</nome>
                                                        <tipo>NUMERIC</tipo>
                                                        <campo>id_produto</campo>
                                                </atributo>
                                                <atributo>
                                                        <nome>Identificador_Categoria</nome>
                                                        <tipo>NUMERIC</tipo>
                                                <campo>identificador_categoria</campo>
                                                </atributo>
                                        </atributos>
                                </indextoindex>
                        </dimensao>
                        <dimensao>
                                <nome>Loja</nome>
                                <tabela>Molap.Loja</tabela>
                                <atributos>
                                        <atributo>
                                                <nome>ID_Loja</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>id_loja</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>Nome</nome>
                                                <tipo>TEXT</tipo>
                                                <campo>nome</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>Estado</nome>
                                                <tipo>TEXT</tipo>
                                                <campo>estado</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>Entrada_cubo</nome>
                                                <tipo>NUMERIC</tipo>
                                                <campo>entrada_cubo</campo>
                                        </atributo>
                                </atributos>
                                <arvoreb>
                                        <nome>IDX2_loja</nome>
                                        <atributos>
                                                <atributo>
                                                        <nome>Id_loja</nome>
                                                        <tipo>NUMERIC</tipo>
                                                        <campo>id_loja</campo>
```

```
                                                </atributo>
                                                <atributo>
                                                        <nome>Entrada_cubo</nome>

                                                        <tipo>NUMERIC</tipo>
                                                        <campo>entrada_cubo</campo>
                                                </atributo>
                                        </atributos>
                                </arvoreb>
                                <indextoindex>
                                        <nome>IndexToIndexLoja</nome>
                                        <tabela>Molap.IndexToIndexLoja</tabela>
                                        <atributos>
                                                <atributo>
                                                        <nome>ID_Loja</nome>
                                                        <tipo>NUMERIC</tipo>
                                                        <campo>id_loja</campo>
                                                </atributo>
                                                <atributo>
                                                        <nome>Identificador_Estado</nome>
                                                        <tipo>NUMERIC</tipo>
                                                        <campo>identificador_estado</campo>
                                                </atributo>
                                        </atributos>
                                </indextoindex>
                        </dimensao>
                </dimensoes>
                <cubos>
                        <cubo>
                                <nome>CuboVendas</nome>
                                <tabela>Molap.Vendas</tabela>
                                <atributos>
                                        <atributo>
                                                <nome>OffSetInChunck</nome>
                                                <tipo>NUMERIC</tipo>
                                                <medida>false</medida>
                                                <campo>offsetinchunck</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>NumeroChunck</nome>
                                                <tipo>NUMERIC</tipo>
                                                <medida>false</medida>
                                                <campo>numerochunck</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>Quantidade</nome>
                                                <tipo>NUMERIC</tipo>
                                                <medida>true</medida>
                                                <campo>quantidade</campo>
                                        </atributo>
                                        <atributo>
                                                <nome>Faturamento</nome>
                                                <tipo>NUMERIC</tipo>
                                                <medida>true</medida>
                                                <campo>faturamento</campo>
                                        </atributo>
                                </atributos>
                        </cubo>
                </cubos>
        </dw>
</ConfigDWVendas>
```

Apêndice C

C.1 Identificando o "gargalo" da aplicação

Este anexo apresenta o arquivo gerado através da ferramenta Hprof, durante a submissão de uma consulta ao protótipo. Esta ferramenta faz parte da especificação do JDK a partir da versão 1.0.2. Ela provê uma maneira precisa de se identificar exatamente onde se encontra o "gargalo" da aplicação, através de um arquivo gerado durante a execução da aplicação. Este arquivo permite que o desenvolvedor analise quais foram os métodos que mais consumiram recursos durante a execução da aplicação.

Foram retiradas deste arquivo as informações dos *traces,* gerados devido a sua extensão. No arquivo, é possível conferir o percentual de tempo gasto pelos métodos mais executados no protótipo.

A diretiva de compilação utilizada para geração deste arquivo foi:

java -Xrunhprof:cpu=samples,file=ProfingMolap.txt,depth=10

ProfingMola.txt

```
JAVA PROFILE 1.0.1, created Wed Jun 20 03:48:10 2001

Header for -Xhprof ASCII Output

Copyright 1998 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto,
California, 94303, U.S.A.  All Rights Reserved.

WARNING!  This file format is under development, and is subject to
change without notice.

This file contains the following types of records:

THREAD START
THREAD END      mark the lifetime of Java threads

TRACE:  represents a Java stack trace.  Each trace consists of a series of stack frames.  Other
records refer to TRACEs to identify (1) where object allocations have  taken place, (2) the
frames in which GC roots were  found, and (3) frequently executed methods.

HEAP DUMP:       is a complete snapshot of all live objects in the Java heap.  Following
distinctions are made:
        ROOT    root set as determined by GC
        CLS             classes
        OBJ             instances
        ARR             arrays


SITES   is a sorted list of allocation sites.  This identifies the most heavily allocated object
types, and the TRACE at which those allocations occurred.
```

```
CPU SAMPLES:     is a statistical profile of program execution. The VM periodically samples all
running threads, and assigns a quantum to active TRACEs in those threads.  Entries in this
record are TRACEs ranked by the percentage of total quanta they consumed; top-ranked TRACEs are
typically hot spots in the program.

CPU TIME is a profile of program execution obtained by measuring the time spent in individual
methods (excluding the time spent in callees), as well as by counting the number of times each
method is called. Entries in this record are TRACEs ranked by the percentage of total CPU time.
The "count" field indicates the number of times each TRACE is invoked.

MONITOR TIME:    is a profile of monitor contention obtained by measuring the time spent by a
thread waiting to enter a monitor. Entries in this record are TRACEs ranked by the percentage of
total monitor contention time and a brief description of the monitor.  The "count" field
indicates the number of times the monitor was contended at that TRACE.

MONITOR DUMP     is a complete snapshot of all the monitors and threads in the System.

HEAP DUMP, SITES, CPU SAMPLES|TIME and MONITOR DUMP|TIME records are generated at program exit.
They can also be obtained during program execution by typing Ctrl-\ (on Solaris) or by typing
Ctrl-Break (on Win32).

--------
.
.
.
--------

CPU SAMPLES BEGIN (total = 651) Wed Feb 20 03:49:32 2002
rank   self  accum   count trace method
   1 51.61% 51.61%     336   126 java.net.SocketInputStream.socketRead
   2  4.30% 55.91%      28    11 java.lang.ClassLoader.defineClass0
   3  4.15% 60.06%      27   145 java.net.SocketInputStream.socketRead
   4  3.07% 63.13%      20   146 java.net.SocketInputStream.socketRead
   5  2.61% 65.75%      17   142 oracle.jdbc.driver.OracleConnection.privatePrepareStatement
   6  2.15% 67.90%      14    78 java.net.SocketOutputStream.socketWrite
   7  1.84% 69.74%      12   112 oracle.jdbc.dbaccess.DBColumn.setDBColumnForV8
   8  1.38% 71.12%       9   123 oracle.jdbc.driver.OracleStatement.executeQuery
   9  1.38% 72.50%       9    27 java.io.Win32FileSystem.getBooleanAttributes
  10  1.23% 73.73%       8    61 oracle.jdbc.ttc7.TTCConversion.<init>
  11  1.23% 74.96%       8    91 oracle.jdbc.driver.OracleConnection.privateCreateStatement
  12  0.92% 75.88%       6    95 oracle.jdbc.dbaccess.DBAccess.createDBDataSet
  13  0.77% 76.65%       5    48 oracle.jdbc.ttc7.TTC7Protocol.connect
  14  0.77% 77.42%       5    75 oracle.jdbc.ttc7.TTCTypeRep.newTTCMsgObject
  15  0.61% 78.03%       4    41 java.lang.Class.forName0
  16  0.61% 78.65%       4    53 oracle.jdbc.ttc7.TTC7Protocol.connect
  17  0.61% 79.26%       4    81 oracle.jdbc.ttc7.TTC7Protocol.logon
  18  0.61% 79.88%       4   140 java.text.resources.LocaleElements.getContents
  19  0.61% 80.49%       4    83 java.net.InetAddressImpl.lookupAllHostAddr
  20  0.61% 81.11%       4    35 java.lang.Class.forName0
  21  0.46% 81.57%       3    25 oracle.net.ns.NetOutputStream.<init>
  22  0.46% 82.03%       3    33 java.lang.Class.forName0
  23  0.46% 82.49%       3    58 oracle.sql.CharacterSet.<clinit>
  24  0.46% 82.95%       3   107 oracle.jdbc.ttc7.v8TTIuds.<init>
  25  0.46% 83.41%       3    54 java.lang.ClassLoader.findBootstrapClass
  26  0.46% 83.87%       3    38 oracle.net.ano.Service.j
  27  0.46% 84.33%       3    51 oracle.jdbc.ttc7.TTC7Protocol.connect
  28  0.46% 84.79%       3   130 java.math.BigDecimal.<init>
  29  0.46% 85.25%       3   113 java.util.zip.ZipFile.getEntry
  30  0.31% 85.56%       2     9 oracle.net.ns.NSProtocol.<init>
  31  0.31% 85.87%       2   148 oracle.jdbc.dbaccess.DBConversion.CHARBytesToJavaChar
  32  0.31% 86.18%       2    13 java.lang.ClassLoader.findBootstrapClass
  33  0.31% 86.48%       2   124 CargaDW.main
  34  0.31% 86.79%       2   120 oracle.jdbc.ttc7.TTC7Protocol.createDBItem
  35  0.31% 87.10%       2    12 oracle.net.ns.NSProtocol.connect
  36  0.31% 87.40%       2    80 oracle.net.ns.DataPacket.getDataFromBuffer
  37  0.31% 87.71%       2    26 sun.misc.URLClassPath.getLoader
  38  0.31% 88.02%       2   121 oracle.jdbc.ttc7.TTCAdapter.createNonPlsqlTTCColumnArray
  39  0.31% 88.33%       2   133 CargaDW.main
  40  0.15% 88.48%       1    42 oracle.net.aso.MD5.<init>
  41  0.15% 88.63%       1    97 java.lang.Class.forName0
  42  0.15% 88.79%       1    55 oracle.jdbc.dbaccess.DBConversion.<init>
  43  0.15% 88.94%       1   139 sun.misc.SoftCache$ValueCell.access$300
  44  0.15% 89.09%       1   111 oracle.jdbc.ttc7.v8Odscrarr.fillupDBcols
  45  0.15% 89.25%       1   104 oracle.jdbc.ttc7.TTCAdapter.<init>
```

```
  46  0.15% 89.40%       1   34 java.lang.System.arraycopy
  47  0.15% 89.55%       1  138 java.lang.Class.newInstance0
  48  0.15% 89.71%       1   76 oracle.jdbc.ttc7.v8TTIdty.<init>
  49  0.15% 89.86%       1   73 oracle.jdbc.ttc7.TTC7Protocol.logon

  50  0.15% 90.02%       1   37 java.util.HashMap.get
  51  0.15% 90.17%       1   30 oracle.net.ano.Ano.init
  52  0.15% 90.32%       1   52 oracle.net.ns.NetInputStream.processPacket
  53  0.15% 90.48%       1  108 oracle.jdbc.ttc7.v8TTIuds.unmarshal
  54  0.15% 90.63%       1  118 oracle.jdbc.oracore.OracleType.<init>
  55  0.15% 90.78%       1  149 oracle.jdbc.dbaccess.DBDataSetImpl._setRowItems
  56  0.15% 90.94%       1   66 oracle.jdbc.ttc7.TTIoer.<init>
  57  0.15% 91.09%       1  137 java.lang.ClassLoader.findBootstrapClass
  58  0.15% 91.24%       1  147 oracle.jdbc.ttc7.Oall7.init
  59  0.15% 91.40%       1   28 java.lang.ClassLoader.findBootstrapClass
  60  0.15% 91.55%       1   39 java.lang.StringBuffer.append
  61  0.15% 91.71%       1   94 oracle.jdbc.ttc7.TTC7Protocol.open
  62  0.15% 91.86%       1   84 oracle.jdbc.ttc7.TTC7Protocol.logon
  63  0.15% 92.01%       1   14 java.io.File.<init>
  64  0.15% 92.17%       1  114 sun.misc.URLClassPath$JarLoader.getResource
  65  0.15% 92.32%       1  103 java.lang.StringBuffer.append
  66  0.15% 92.47%       1    7 java.net.URLClassLoader.access$100
  67  0.15% 92.63%       1   43 java.lang.ClassLoader$NativeLibrary.find
  68  0.15% 92.78%       1  102 oracle.jdbc.ttc7.TTC7Protocol.parseExecuteFetch
  69  0.15% 92.93%       1   59 oracle.sql.CharacterSetFactoryThin.make
  70  0.15% 93.09%       1  100 oracle.jdbc.ttc7.TTCAdapter.createNonPlsqlTTCDataSet
  71  0.15% 93.24%       1   72 java.lang.StringBuffer.expandCapacity
  72  0.15% 93.39%       1  134 java.lang.StringBuffer.append
  73  0.15% 93.55%       1   77 java.lang.System.arraycopy
  74  0.15% 93.70%       1   40 java.lang.String.<init>
  75  0.15% 93.86%       1   67 java.lang.String.startsWith
  76  0.15% 94.01%       1   32 sun.misc.URLClassPath.getLoader
  77  0.15% 94.16%       1   31 oracle.net.ano.Ano.e
  78  0.15% 94.32%       1   15 java.lang.StringBuffer.append
  79  0.15% 94.47%       1   70 java.lang.ClassLoader.findBootstrapClass
  80  0.15% 94.62%       1   23 java.net.InetAddressImpl.getLocalHostName
  81  0.15% 94.78%       1  116 java.lang.String.regionMatches
  82  0.15% 94.93%       1   79 oracle.net.ns.NetInputStream.read
  83  0.15% 95.08%       1   65 oracle.jdbc.ttc7.Oall7.<init>
  84  0.15% 95.24%       1   16 java.util.jar.JarFile.getInputStream
  85  0.15% 95.39%       1  106 oracle.jdbc.ttc7.v8Odscrarr.receive
  86  0.15% 95.55%       1  115 java.lang.ClassLoader.findBootstrapClass
  87  0.15% 95.70%       1   64 oracle.jdbc.ttc7.Oall7.<init>
  88  0.15% 95.85%       1   56 java.lang.Character.toUpperCase
  89  0.15% 96.01%       1  141 java.util.GregorianCalendar.timeToFields
  90  0.15% 96.16%       1  150 oracle.jdbc.ttc7.MAREngine.value2Buffer
  91  0.15% 96.31%       1  129 java.lang.ClassLoader.findBootstrapClass
  92  0.15% 96.47%       1  122 oracle.jdbc.ttc7.MAREngine.unmarshalCLRforREFS
  93  0.15% 96.62%       1  119 oracle.jdbc.dbaccess.DBDataSetImpl._definesRowCompleted
  94  0.15% 96.77%       1  101 oracle.jdbc.ttc7.TTCDataSet.<init>
  95  0.15% 96.93%       1   93 java.lang.ClassLoader.check
  96  0.15% 97.08%       1   36 java.lang.StringBuffer.<init>
  97  0.15% 97.24%       1  109 java.lang.String.substring
  98  0.15% 97.39%       1   24 oracle.net.ns.NSProtocol.establishConnection
  99  0.15% 97.54%       1  144 java.lang.String.replace
 100  0.15% 97.70%       1   92 java.lang.StringBuffer.toString
 101  0.15% 97.85%       1  132 oracle.jdbc.driver.OracleStatement.getStringValue
 102  0.15% 98.00%       1   68 oracle.jdbc.ttc7.TTIrid.<init>
 103  0.15% 98.16%       1   29 java.lang.Class.forName0
 104  0.15% 98.31%       1   60 java.security.AccessController.doPrivileged
 105  0.15% 98.46%       1   85 java.net.SocketInputStream.socketRead
 106  0.15% 98.62%       1   10 sun.net.www.protocol.file.Handler.parseURL
 107  0.15% 98.77%       1  110 java.util.zip.ZipFile$ZipFileInputStream.read
 108  0.15% 98.92%       1  105 oracle.jdbc.ttc7.v8Odscrarr.receive
 109  0.15% 99.08%       1  125 oracle.jdbc.ttc7.MAREngine.value2Buffer
 110  0.15% 99.23%       1   82 sun.net.InetAddressCachePolicy.<clinit>
 111  0.15% 99.39%       1   49 java.net.URLStreamHandler.parseURL
 112  0.15% 99.54%       1  131 java.lang.String.substring
 113  0.15% 99.69%       1   44 java.net.SocketInputStream.socketRead
 114  0.15% 99.85%       1   57 java.lang.ClassLoader.findLoadedClass
 115  0.15% 100.00%      1  117 java.security.AccessController.doPrivileged
CPU SAMPLES END
```

Nº Chunk: 1			20032	2 4 2	2 4 2	60024	4 3 4	4 3 4	100016	6 2 6	6 2 6	140008	8 1 8	8 1 8	160040	9 4 10	9 4 10
1	1 1 1	1 1 1	20033	2 4 3	2 4 3	60025	4 3 5	4 3 5	100017	6 2 7	6 2 7	140009	8 1 9	8 1 9	180001	10 1 1	10 1 1
2	1 1 2	1 1 2	20034	2 4 4	2 4 4	60026	4 3 6	4 3 6	100018	6 2 8	6 2 8	140010	8 1 10	8 1 10	180002	10 1 2	10 1 2
3	1 1 3	1 1 3	20035	2 4 5	2 4 5	60027	4 3 7	4 3 7	100019	6 2 9	6 2 9	140011	8 2 1	8 2 1	180003	10 1 3	10 1 3
4	1 1 4	1 1 4	20036	2 4 6	2 4 6	60028	4 3 8	4 3 8	100020	6 2 10	6 2 10	140012	8 2 2	8 2 2	180004	10 1 4	10 1 4
5	1 1 5	1 1 5	20037	2 4 7	2 4 7	60029	4 3 9	4 3 9	100021	6 3 1	6 3 1	140013	8 2 3	8 2 3	180005	10 1 5	10 1 5
6	1 1 6	1 1 6	20038	2 4 8	2 4 8	60030	4 3 10	4 3 10	100022	6 3 2	6 3 2	140014	8 2 4	8 2 4	180006	10 1 6	10 1 6
7	1 1 7	1 1 7	20039	2 4 9	2 4 9	60031	4 4 1	4 4 1	100023	6 3 3	6 3 3	140015	8 2 5	8 2 5	180007	10 1 7	10 1 7
8	1 1 8	1 1 8	20040	2 4 10	2 4 10	60032	4 4 2	4 4 2	100024	6 3 4	6 3 4	140016	8 2 6	8 2 6	180008	10 1 8	10 1 8
9	1 1 9	1 1 9	40001	3 1 1	3 1 1	60033	4 4 3	4 4 3	100025	6 3 5	6 3 5	140017	8 2 7	8 2 7	180009	10 1 9	10 1 9
10	1 1 10	1 1 10	40002	3 1 2	3 1 2	60034	4 4 4	4 4 4	100026	6 3 6	6 3 6	140018	8 2 8	8 2 8	180010	10 1 10	10 1 10
11	1 2 1	1 2 1	40003	3 1 3	3 1 3	60035	4 4 5	4 4 5	100027	6 3 7	6 3 7	140019	8 2 9	8 2 9	180011	10 2 1	10 2 1
12	1 2 2	1 2 2	40004	3 1 4	3 1 4	60036	4 4 6	4 4 6	100028	6 3 8	6 3 8	140020	8 2 10	8 2 10	180012	10 2 2	10 2 2
13	1 2 3	1 2 3	40005	3 1 5	3 1 5	60037	4 4 7	4 4 7	100029	6 3 9	6 3 9	140021	8 3 1	8 3 1	180013	10 2 3	10 2 3
14	1 2 4	1 2 4	40006	3 1 6	3 1 6	60038	4 4 8	4 4 8	100030	6 3 10	6 3 10	140022	8 3 2	8 3 2	180014	10 2 4	10 2 4
15	1 2 5	1 2 5	40007	3 1 7	3 1 7	60039	4 4 9	4 4 9	100031	6 4 1	6 4 1	140023	8 3 3	8 3 3	180015	10 2 5	10 2 5
16	1 2 6	1 2 6	40008	3 1 8	3 1 8	60040	4 4 10	4 4 10	100032	6 4 2	6 4 2	140024	8 3 4	8 3 4	180016	10 2 6	10 2 6
17	1 2 7	1 2 7	40009	3 1 9	3 1 9	80001	5 1 1	5 1 1	100033	6 4 3	6 4 3	140025	8 3 5	8 3 5	180017	10 2 7	10 2 7
18	1 2 8	1 2 8	40010	3 1 10	3 1 10	80002	5 1 2	5 1 2	100034	6 4 4	6 4 4	140026	8 3 6	8 3 6	180018	10 2 8	10 2 8
19	1 2 9	1 2 9	40011	3 2 1	3 2 1	80003	5 1 3	5 1 3	100035	6 4 5	6 4 5	140027	8 3 7	8 3 7	180019	10 2 9	10 2 9
20	1 2 10	1 2 10	40012	3 2 2	3 2 2	80004	5 1 4	5 1 4	100036	6 4 6	6 4 6	140028	8 3 8	8 3 8	180020	10 2 10	10 2 10
21	1 3 1	1 3 1	40013	3 2 3	3 2 3	80005	5 1 5	5 1 5	100037	6 4 7	6 4 7	140029	8 3 9	8 3 9	180021	10 3 1	10 3 1
22	1 3 2	1 3 2	40014	3 2 4	3 2 4	80006	5 1 6	5 1 6	100038	6 4 8	6 4 8	140030	8 3 10	8 3 10	180022	10 3 2	10 3 2
23	1 3 3	1 3 3	40015	3 2 5	3 2 5	80007	5 1 7	5 1 7	100039	6 4 9	6 4 9	140031	8 4 1	8 4 1	180023	10 3 3	10 3 3
24	1 3 4	1 3 4	40016	3 2 6	3 2 6	80008	5 1 8	5 1 8	100040	6 4 10	6 4 10	140032	8 4 2	8 4 2	180024	10 3 4	10 3 4
25	1 3 5	1 3 5	40017	3 2 7	3 2 7	80009	5 1 9	5 1 9	120001	7 1 1	7 1 1	140033	8 4 3	8 4 3	180025	10 3 5	10 3 5
26	1 3 6	1 3 6	40018	3 2 8	3 2 8	80010	5 1 10	5 1 10	120002	7 1 2	7 1 2	140034	8 4 4	8 4 4	180026	10 3 6	10 3 6
27	1 3 7	1 3 7	40019	3 2 9	3 2 9	80011	5 2 1	5 2 1	120003	7 1 3	7 1 3	140035	8 4 5	8 4 5	180027	10 3 7	10 3 7
28	1 3 8	1 3 8	40020	3 2 10	3 2 10	80012	5 2 2	5 2 2	120004	7 1 4	7 1 4	140036	8 4 6	8 4 6	180028	10 3 8	10 3 8
29	1 3 9	1 3 9	40021	3 3 1	3 3 1	80013	5 2 3	5 2 3	120005	7 1 5	7 1 5	140037	8 4 7	8 4 7	180029	10 3 9	10 3 9
30	1 3 10	1 3 10	40022	3 3 2	3 3 2	80014	5 2 4	5 2 4	120006	7 1 6	7 1 6	140038	8 4 8	8 4 8	180030	10 3 10	10 3 10
31	1 4 1	1 4 1	40023	3 3 3	3 3 3	80015	5 2 5	5 2 5	120007	7 1 7	7 1 7	140039	8 4 9	8 4 9	180031	10 4 1	10 4 1
32	1 4 2	1 4 2	40024	3 3 4	3 3 4	80016	5 2 6	5 2 6	120008	7 1 8	7 1 8	140040	8 4 10	8 4 10	180032	10 4 2	10 4 2
33	1 4 3	1 4 3	40025	3 3 5	3 3 5	80017	5 2 7	5 2 7	120009	7 1 9	7 1 9	160001	9 1 1	9 1 1	180033	10 4 3	10 4 3
34	1 4 4	1 4 4	40026	3 3 6	3 3 6	80018	5 2 8	5 2 8	120010	7 1 10	7 1 10	160002	9 1 2	9 1 2	180034	10 4 4	10 4 4
35	1 4 5	1 4 5	40027	3 3 7	3 3 7	80019	5 2 9	5 2 9	120011	7 2 1	7 2 1	160003	9 1 3	9 1 3	180035	10 4 5	10 4 5
36	1 4 6	1 4 6	40028	3 3 8	3 3 8	80020	5 2 10	5 2 10	120012	7 2 2	7 2 2	160004	9 1 4	9 1 4	180036	10 4 6	10 4 6
37	1 4 7	1 4 7	40029	3 3 9	3 3 9	80021	5 3 1	5 3 1	120013	7 2 3	7 2 3	160005	9 1 5	9 1 5	180037	10 4 7	10 4 7
38	1 4 8	1 4 8	40030	3 3 10	3 3 10	80022	5 3 2	5 3 2	120014	7 2 4	7 2 4	160006	9 1 6	9 1 6	180038	10 4 8	10 4 8
39	1 4 9	1 4 9	40031	3 4 1	3 4 1	80023	5 3 3	5 3 3	120015	7 2 5	7 2 5	160007	9 1 7	9 1 7	180039	10 4 9	10 4 9
40	1 4 10	1 4 10	40032	3 4 2	3 4 2	80024	5 3 4	5 3 4	120016	7 2 6	7 2 6	160008	9 1 8	9 1 8	180040	10 4 10	10 4 10
20001	2 1 1	2 1 1	40033	3 4 3	3 4 3	80025	5 3 5	5 3 5	120017	7 2 7	7 2 7	160009	9 1 9	9 1 9	**Nº Chunk: 2**		
20002	2 1 2	2 1 2	40034	3 4 4	3 4 4	80026	5 3 6	5 3 6	120018	7 2 8	7 2 8	160010	9 1 10	9 1 10	1	1 1 1	1 1 11
20003	2 1 3	2 1 3	40035	3 4 5	3 4 5	80027	5 3 7	5 3 7	120019	7 2 9	7 2 9	160011	9 2 1	9 2 1	2	1 1 2	1 1 12
20004	2 1 4	2 1 4	40036	3 4 6	3 4 6	80028	5 3 8	5 3 8	120020	7 2 10	7 2 10	160012	9 2 2	9 2 2	3	1 1 3	1 1 13
20005	2 1 5	2 1 5	40037	3 4 7	3 4 7	80029	5 3 9	5 3 9	120021	7 3 1	7 3 1	160013	9 2 3	9 2 3	4	1 1 4	1 1 14
20006	2 1 6	2 1 6	40038	3 4 8	3 4 8	80030	5 3 10	5 3 10	120022	7 3 2	7 3 2	160014	9 2 4	9 2 4	5	1 1 5	1 1 15
20007	2 1 7	2 1 7	40039	3 4 9	3 4 9	80031	5 4 1	5 4 1	120023	7 3 3	7 3 3	160015	9 2 5	9 2 5	6	1 1 6	1 1 16
20008	2 1 8	2 1 8	40040	3 4 10	3 4 10	80032	5 4 2	5 4 2	120024	7 3 4	7 3 4	160016	9 2 6	9 2 6	7	1 1 7	1 1 17
20009	2 1 9	2 1 9	60001	4 1 1	4 1 1	80033	5 4 3	5 4 3	120025	7 3 5	7 3 5	160017	9 2 7	9 2 7	8	1 1 8	1 1 18
20010	2 1 10	2 1 10	60002	4 1 2	4 1 2	80034	5 4 4	5 4 4	120026	7 3 6	7 3 6	160018	9 2 8	9 2 8	9	1 1 9	1 1 19
20011	2 2 1	2 2 1	60003	4 1 3	4 1 3	80035	5 4 5	5 4 5	120027	7 3 7	7 3 7	160019	9 2 9	9 2 9	10	1 1 10	1 1 20
20012	2 2 2	2 2 2	60004	4 1 4	4 1 4	80036	5 4 6	5 4 6	120028	7 3 8	7 3 8	160020	9 2 10	9 2 10	11	1 2 1	1 2 11
20013	2 2 3	2 2 3	60005	4 1 5	4 1 5	80037	5 4 7	5 4 7	120029	7 3 9	7 3 9	160021	9 3 1	9 3 1	12	1 2 2	1 2 12
20014	2 2 4	2 2 4	60006	4 1 6	4 1 6	80038	5 4 8	5 4 8	120030	7 3 10	7 3 10	160022	9 3 2	9 3 2	13	1 2 3	1 2 13
20015	2 2 5	2 2 5	60007	4 1 7	4 1 7	80039	5 4 9	5 4 9	120031	7 4 1	7 4 1	160023	9 3 3	9 3 3	14	1 2 4	1 2 14
20016	2 2 6	2 2 6	60008	4 1 8	4 1 8	80040	5 4 10	5 4 10	120032	7 4 2	7 4 2	160024	9 3 4	9 3 4	15	1 2 5	1 2 15
20017	2 2 7	2 2 7	60009	4 1 9	4 1 9	100001	6 1 1	6 1 1	120033	7 4 3	7 4 3	160025	9 3 5	9 3 5	16	1 2 6	1 2 16
20018	2 2 8	2 2 8	60010	4 1 10	4 1 10	100002	6 1 2	6 1 2	120034	7 4 4	7 4 4	160026	9 3 6	9 3 6	17	1 2 7	1 2 17
20019	2 2 9	2 2 9	60011	4 2 1	4 2 1	100003	6 1 3	6 1 3	120035	7 4 5	7 4 5	160027	9 3 7	9 3 7	18	1 2 8	1 2 18
20020	2 2 10	2 2 10	60012	4 2 2	4 2 2	100004	6 1 4	6 1 4	120036	7 4 6	7 4 6	160028	9 3 8	9 3 8	19	1 2 9	1 2 19
20021	2 3 1	2 3 1	60013	4 2 3	4 2 3	100005	6 1 5	6 1 5	120037	7 4 7	7 4 7	160029	9 3 9	9 3 9	20	1 2 10	1 2 20
20022	2 3 2	2 3 2	60014	4 2 4	4 2 4	100006	6 1 6	6 1 6	120038	7 4 8	7 4 8	160030	9 3 10	9 3 10	21	1 3 1	1 3 11
20023	2 3 3	2 3 3	60015	4 2 5	4 2 5	100007	6 1 7	6 1 7	120039	7 4 9	7 4 9	160031	9 4 1	9 4 1	22	1 3 2	1 3 12
20024	2 3 4	2 3 4	60016	4 2 6	4 2 6	100008	6 1 8	6 1 8	120040	7 4 10	7 4 10	160032	9 4 2	9 4 2	23	1 3 3	1 3 13
20025	2 3 5	2 3 5	60017	4 2 7	4 2 7	100009	6 1 9	6 1 9	140001	8 1 1	8 1 1	160033	9 4 3	9 4 3	24	1 3 4	1 3 14
20026	2 3 6	2 3 6	60018	4 2 8	4 2 8	100010	6 1 10	6 1 10	140002	8 1 2	8 1 2	160034	9 4 4	9 4 4	25	1 3 5	1 3 15
20027	2 3 7	2 3 7	60019	4 2 9	4 2 9	100011	6 2 1	6 2 1	140003	8 1 3	8 1 3	160035	9 4 5	9 4 5	26	1 3 6	1 3 16
20028	2 3 8	2 3 8	60020	4 2 10	4 2 10	100012	6 2 2	6 2 2	140004	8 1 4	8 1 4	160036	9 4 6	9 4 6	27	1 3 7	1 3 17
20029	2 3 9	2 3 9	60021	4 3 1	4 3 1	100013	6 2 3	6 2 3	140005	8 1 5	8 1 5	160037	9 4 7	9 4 7	28	1 3 8	1 3 18
20030	2 3 10	2 3 10	60022	4 3 2	4 3 2	100014	6 2 4	6 2 4	140006	8 1 6	8 1 6	160038	9 4 8	9 4 8	29	1 3 9	1 3 19
20031	2 4 1	2 4 1	60023	4 3 3	4 3 3	100015	6 2 5	6 2 5	140007	8 1 7	8 1 7	160039	9 4 9	9 4 9	30	1 3 10	1 3 20

31	1 4 1	1 4 11	40023	3 3 3	3 3 13	80015	5 2 5	5 2 15	120007	7 1 7	7 1 17	140039	8 4 9	8 4 19	180031	10 4 1	10 4 11
32	1 4 2	1 4 12	40024	3 3 4	3 3 14	80016	5 2 6	5 2 16	120008	7 1 8	7 1 18	140040	8 4 10	8 4 20	180032	10 4 2	10 4 12
33	1 4 3	1 4 13	40025	3 3 5	3 3 15	80017	5 2 7	5 2 17	120009	7 1 9	7 1 19	160001	9 1 1	9 1 11	180033	10 4 3	10 4 13
34	1 4 4	1 4 14	40026	3 3 6	3 3 16	80018	5 2 8	5 2 18	120010	7 1 10	7 1 20	160002	9 1 2	9 1 12	180034	10 4 4	10 4 14
35	1 4 5	1 4 15	40027	3 3 7	3 3 17	80019	5 2 9	5 2 19	120011	7 2 1	7 2 11	160003	9 1 3	9 1 13	180035	10 4 5	10 4 15
36	1 4 6	1 4 16	40028	3 3 8	3 3 18	80020	5 2 10	5 2 20	120012	7 2 2	7 2 12	160004	9 1 4	9 1 14	180036	10 4 6	10 4 16
37	1 4 7	1 4 17	40029	3 3 9	3 3 19	80021	5 3 1	5 3 11	120013	7 2 3	7 2 13	160005	9 1 5	9 1 15	180037	10 4 7	10 4 17
38	1 4 8	1 4 18	40030	3 3 10	3 3 20	80022	5 3 2	5 3 12	120014	7 2 4	7 2 14	160006	9 1 6	9 1 16	180038	10 4 8	10 4 18
39	1 4 9	1 4 19	40031	3 4 1	3 4 11	80023	5 3 3	5 3 13	120015	7 2 5	7 2 15	160007	9 1 7	9 1 17	180039	10 4 9	10 4 19
40	1 4 10	1 4 20	40032	3 4 2	3 4 12	80024	5 3 4	5 3 14	120016	7 2 6	7 2 16	160008	9 1 8	9 1 18	180040	10 4 10	10 4 204
20001	2 1 1	2 1 11	40033	3 4 3	3 4 13	80025	5 3 5	5 3 15	120017	7 2 7	7 2 17	160009	9 1 9	9 1 19			
20002	2 1 2	2 1 12	40034	3 4 4	3 4 14	80026	5 3 6	5 3 16	120018	7 2 8	7 2 18	160010	9 1 10	9 1 20		•	
20003	2 1 3	2 1 13	40035	3 4 5	3 4 15	80027	5 3 7	5 3 17	120019	7 2 9	7 2 19	160011	9 2 1	9 2 11		•	
20004	2 1 4	2 1 14	40036	3 4 6	3 4 16	80028	5 3 8	5 3 18	120020	7 2 10	7 2 20	160012	9 2 2	9 2 12		•	
20005	2 1 5	2 1 15	40037	3 4 7	3 4 17	80029	5 3 9	5 3 19	120021	7 3 1	7 3 11	160013	9 2 3	9 2 13			
20006	2 1 6	2 1 16	40038	3 4 8	3 4 18	80030	5 3 10	5 3 20	120022	7 3 2	7 3 12	160014	9 2 4	9 2 14			
20007	2 1 7	2 1 17	40039	3 4 9	3 4 19	80031	5 4 1	5 4 11	120023	7 3 3	7 3 13	160015	9 2 5	9 2 15			
20008	2 1 8	2 1 18	40040	3 4 10	3 4 20	80032	5 4 2	5 4 12	120024	7 3 4	7 3 14	160016	9 2 6	9 2 16			
20009	2 1 9	2 1 19	60001	4 1 1	4 1 11	80033	5 4 3	5 4 13	120025	7 3 5	7 3 15	160017	9 2 7	9 2 17			
20010	2 1 10	2 1 20	60002	4 1 2	4 1 12	80034	5 4 4	5 4 14	120026	7 3 6	7 3 16	160018	9 2 8	9 2 18			
20011	2 2 1	2 2 11	60003	4 1 3	4 1 13	80035	5 4 5	5 4 15	120027	7 3 7	7 3 17	160019	9 2 9	9 2 19			
20012	2 2 2	2 2 12	60004	4 1 4	4 1 14	80036	5 4 6	5 4 16	120028	7 3 8	7 3 18	160020	9 2 10	9 2 20			
20013	2 2 3	2 2 13	60005	4 1 5	4 1 15	80037	5 4 7	5 4 17	120029	7 3 9	7 3 19	160021	9 3 1	9 3 11			
20014	2 2 4	2 2 14	60006	4 1 6	4 1 16	80038	5 4 8	5 4 18	120030	7 3 10	7 3 20	160022	9 3 2	9 3 12			
20015	2 2 5	2 2 15	60007	4 1 7	4 1 17	80039	5 4 9	5 4 19	120031	7 4 1	7 4 11	160023	9 3 3	9 3 13			
20016	2 2 6	2 2 16	60008	4 1 8	4 1 18	80040	5 4 10	5 4 20	120032	7 4 2	7 4 12	160024	9 3 4	9 3 14			
20017	2 2 7	2 2 17	60009	4 1 9	4 1 19	100001	6 1 1	6 1 11	120033	7 4 3	7 4 13	160025	9 3 5	9 3 15			
20018	2 2 8	2 2 18	60010	4 1 10	4 1 20	100002	6 1 2	6 1 12	120034	7 4 4	7 4 14	160026	9 3 6	9 3 16			
20019	2 2 9	2 2 19	60011	4 2 1	4 2 11	100003	6 1 3	6 1 13	120035	7 4 5	7 4 15	160027	9 3 7	9 3 17			
20020	2 2 10	2 2 20	60012	4 2 2	4 2 12	100004	6 1 4	6 1 14	120036	7 4 6	7 4 16	160028	9 3 8	9 3 18			
20021	2 3 1	2 3 11	60013	4 2 3	4 2 13	100005	6 1 5	6 1 15	120037	7 4 7	7 4 17	160029	9 3 9	9 3 19			
20022	2 3 2	2 3 12	60014	4 2 4	4 2 14	100006	6 1 6	6 1 16	120038	7 4 8	7 4 18	160030	9 3 10	9 3 20			
20023	2 3 3	2 3 13	60015	4 2 5	4 2 15	100007	6 1 7	6 1 17	120039	7 4 9	7 4 19	160031	9 4 1	9 4 11			
20024	2 3 4	2 3 14	60016	4 2 6	4 2 16	100008	6 1 8	6 1 18	120040	7 4 10	7 4 20	160032	9 4 2	9 4 12			
20025	2 3 5	2 3 15	60017	4 2 7	4 2 17	100009	6 1 9	6 1 19	140001	8 1 1	8 1 11	160033	9 4 3	9 4 13			
20026	2 3 6	2 3 16	60018	4 2 8	4 2 18	100010	6 1 10	6 1 20	140002	8 1 2	8 1 12	160034	9 4 4	9 4 14			
20027	2 3 7	2 3 17	60019	4 2 9	4 2 19	100011	6 2 1	6 2 11	140003	8 1 3	8 1 13	160035	9 4 5	9 4 15			
20028	2 3 8	2 3 18	60020	4 2 10	4 2 20	100012	6 2 2	6 2 12	140004	8 1 4	8 1 14	160036	9 4 6	9 4 16			
20029	2 3 9	2 3 19	60021	4 3 1	4 3 11	100013	6 2 3	6 2 13	140005	8 1 5	8 1 15	160037	9 4 7	9 4 17			
20030	2 3 10	2 3 20	60022	4 3 2	4 3 12	100014	6 2 4	6 2 14	140006	8 1 6	8 1 16	160038	9 4 8	9 4 18			
20031	2 4 1	2 4 11	60023	4 3 3	4 3 13	100015	6 2 5	6 2 15	140007	8 1 7	8 1 17	160039	9 4 9	9 4 19			
20032	2 4 2	2 4 12	60024	4 3 4	4 3 14	100016	6 2 6	6 2 16	140008	8 1 8	8 1 18	160040	9 4 10	9 4 20			
20033	2 4 3	2 4 13	60025	4 3 5	4 3 15	100017	6 2 7	6 2 17	140009	8 1 9	8 1 19	180001	10 1 1	10 1 11			
20034	2 4 4	2 4 14	60026	4 3 6	4 3 16	100018	6 2 8	6 2 18	140010	8 1 10	8 1 20	180002	10 1 2	10 1 12			
20035	2 4 5	2 4 15	60027	4 3 7	4 3 17	100019	6 2 9	6 2 19	140011	8 2 1	8 2 11	180003	10 1 3	10 1 13			
20036	2 4 6	2 4 16	60028	4 3 8	4 3 18	100020	6 2 10	6 2 20	140012	8 2 2	8 2 12	180004	10 1 4	10 1 14			
20037	2 4 7	2 4 17	60029	4 3 9	4 3 19	100021	6 3 1	6 3 11	140013	8 2 3	8 2 13	180005	10 1 5	10 1 15			
20038	2 4 8	2 4 18	60030	4 3 10	4 3 20	100022	6 3 2	6 3 12	140014	8 2 4	8 2 14	180006	10 1 6	10 1 16			
20039	2 4 9	2 4 19	60031	4 4 1	4 4 11	100023	6 3 3	6 3 13	140015	8 2 5	8 2 15	180007	10 1 7	10 1 17			
20040	2 4 10	2 4 20	60032	4 4 2	4 4 12	100024	6 3 4	6 3 14	140016	8 2 6	8 2 16	180008	10 1 8	10 1 18			
40001	3 1 1	3 1 11	60033	4 4 3	4 4 13	100025	6 3 5	6 3 15	140017	8 2 7	8 2 17	180009	10 1 9	10 1 19			
40002	3 1 2	3 1 12	60034	4 4 4	4 4 14	100026	6 3 6	6 3 16	140018	8 2 8	8 2 18	180010	10 1 10	10 1 20			
40003	3 1 3	3 1 13	60035	4 4 5	4 4 15	100027	6 3 7	6 3 17	140019	8 2 9	8 2 19	180011	10 2 1	10 2 11			
40004	3 1 4	3 1 14	60036	4 4 6	4 4 16	100028	6 3 8	6 3 18	140020	8 2 10	8 2 20	180012	10 2 2	10 2 12			
40005	3 1 5	3 1 15	60037	4 4 7	4 4 17	100029	6 3 9	6 3 19	140021	8 3 1	8 3 11	180013	10 2 3	10 2 13			
40006	3 1 6	3 1 16	60038	4 4 8	4 4 18	100030	6 3 10	6 3 20	140022	8 3 2	8 3 12	180014	10 2 4	10 2 14			
40007	3 1 7	3 1 17	60039	4 4 9	4 4 19	100031	6 4 1	6 4 11	140023	8 3 3	8 3 13	180015	10 2 5	10 2 15			
40008	3 1 8	3 1 18	60040	4 4 10	4 4 20	100032	6 4 2	6 4 12	140024	8 3 4	8 3 14	180016	10 2 6	10 2 16			
40009	3 1 9	3 1 19	80001	5 1 1	5 1 11	100033	6 4 3	6 4 13	140025	8 3 5	8 3 15	180017	10 2 7	10 2 17			
40010	3 1 10	3 1 20	80002	5 1 2	5 1 12	100034	6 4 4	6 4 14	140026	8 3 6	8 3 16	180018	10 2 8	10 2 18			
40011	3 2 1	3 2 11	80003	5 1 3	5 1 13	100035	6 4 5	6 4 15	140027	8 3 7	8 3 17	180019	10 2 9	10 2 19			
40012	3 2 2	3 2 12	80004	5 1 4	5 1 14	100036	6 4 6	6 4 16	140028	8 3 8	8 3 18	180020	10 2 10	10 2 20			
40013	3 2 3	3 2 13	80005	5 1 5	5 1 15	100037	6 4 7	6 4 17	140029	8 3 9	8 3 19	180021	10 3 1	10 3 11			
40014	3 2 4	3 2 14	80006	5 1 6	5 1 16	100038	6 4 8	6 4 18	140030	8 3 10	8 3 20	180022	10 3 2	10 3 12			
40015	3 2 5	3 2 15	80007	5 1 7	5 1 17	100039	6 4 9	6 4 19	140031	8 4 1	8 4 11	180023	10 3 3	10 3 13			
40016	3 2 6	3 2 16	80008	5 1 8	5 1 18	100040	6 4 10	6 4 20	140032	8 4 2	8 4 12	180024	10 3 4	10 3 14			
40017	3 2 7	3 2 17	80009	5 1 9	5 1 19	120001	7 1 1	7 1 11	140033	8 4 3	8 4 13	180025	10 3 5	10 3 15			
40018	3 2 8	3 2 18	80010	5 1 10	5 1 20	120002	7 1 2	7 1 12	140034	8 4 4	8 4 14	180026	10 3 6	10 3 16			
40019	3 2 9	3 2 19	80011	5 2 1	5 2 11	120003	7 1 3	7 1 13	140035	8 4 5	8 4 15	180027	10 3 7	10 3 17			
40020	3 2 10	3 2 20	80012	5 2 2	5 2 12	120004	7 1 4	7 1 14	140036	8 4 6	8 4 16	180028	10 3 8	10 3 18			
40021	3 3 1	3 3 11	80013	5 2 3	5 2 13	120005	7 1 5	7 1 15	140037	8 4 7	8 4 17	180029	10 3 9	10 3 19			
40022	3 3 2	3 3 12	80014	5 2 4	5 2 14	120006	7 1 6	7 1 16	140038	8 4 8	8 4 18	180030	10 3 10	10 3 20			

Nº do Chunk: 1500

599601	1 1 1	1 21 1997
1	1 1 1	1 1997 21
2	1 1 2	1 1997 22
3	1 1 3	1 1997 23
4	1 1 4	1 1997 24
5	1 1 5	1 1997 25
6	1 1 6	1 1997 26
7	1 1 7	1 1997 27
8	1 1 8	1 1997 28
9	1 1 9	1 1997 29
10	1 1 10	1 1997 30
11	1 2 1	1 1998 21
12	1 2 2	1 1998 22
13	1 2 3	1 1998 23
14	1 2 4	1 1998 24
15	1 2 5	1 1998 25
16	1 2 6	1 1998 26
17	1 2 7	1 1998 27
18	1 2 8	1 1998 28
19	1 2 9	1 1998 29
20	1 2 10	1 1998 30
21	1 3 1	1 1999 21
22	1 3 2	1 1999 22
23	1 3 3	1 1999 23
24	1 3 4	1 1999 24
25	1 3 5	1 1999 25
26	1 3 6	1 1999 26
27	1 3 7	1 1999 27
28	1 3 8	1 1999 28
29	1 3 9	1 1999 29
30	1 3 10	1 1999 30
31	1 4 1	1 2000 21
32	1 4 2	1 2000 22
33	1 4 3	1 2000 23
34	1 4 4	1 2000 24
35	1 4 5	1 2000 25
36	1 4 6	1 2000 26
37	1 4 7	1 2000 27
38	1 4 8	1 2000 28
39	1 4 9	1 2000 29
40	1 4 10	1 2000 30
20001	2 1 1	2 1997 21
20002	2 1 2	2 1997 22
20003	2 1 3	2 1997 23
20004	2 1 4	2 1997 24
20005	2 1 5	2 1997 25
20006	2 1 6	2 1997 26
20007	2 1 7	2 1997 27
20008	2 1 8	2 1997 28
20009	2 1 9	2 1997 29
20010	2 1 10	2 1997 30
20011	2 2 1	2 1998 21
20012	2 2 2	2 1998 22
20013	2 2 3	2 1998 23
20014	2 2 4	2 1998 24

20015	2 2 5	2 1998 25	60007	4 1 7	4 1997 27	80039	5 4 9	5 2000 29	120031	7 4 1	7 2000 21	160023	9 3 3	9 1999 23
20016	2 2 6	2 1998 26	60008	4 1 8	4 1997 28	80040	5 4 10	5 2000 30	120032	7 4 2	7 2000 22	160024	9 3 4	9 1999 24
20017	2 2 7	2 1998 27	60009	4 1 9	4 1997 29	100001	6 1 1	6 1997 21	120033	7 4 3	7 2000 23	160025	9 3 5	9 1999 25
20018	2 2 8	2 1998 28	60010	4 1 10	4 1997 30	100002	6 1 2	6 1997 22	120034	7 4 4	7 2000 24	160026	9 3 6	9 1999 26
20019	2 2 9	2 1998 29	60011	4 2 1	4 1998 21	100003	6 1 3	6 1997 23	120035	7 4 5	7 2000 25	160027	9 3 7	9 1999 27
20020	2 2 10	2 1998 30	60012	4 2 2	4 1998 22	100004	6 1 4	6 1997 24	120036	7 4 6	7 2000 26	160028	9 3 8	9 1999 28
20021	2 3 1	2 1999 21	60013	4 2 3	4 1998 23	100005	6 1 5	6 1997 25	120037	7 4 7	7 2000 27	160029	9 3 9	9 1999 29
20022	2 3 2	2 1999 22	60014	4 2 4	4 1998 24	100006	6 1 6	6 1997 26	120038	7 4 8	7 2000 28	160030	9 3 10	9 1999 30
20023	2 3 3	2 1999 23	60015	4 2 5	4 1998 25	100007	6 1 7	6 1997 27	120039	7 4 9	7 2000 29	160031	9 4 1	9 2000 21
20024	2 3 4	2 1999 24	60016	4 2 6	4 1998 26	100008	6 1 8	6 1997 28	120040	7 4 10	7 2000 30	160032	9 4 2	9 2000 22
20025	2 3 5	2 1999 25	60017	4 2 7	4 1998 27	100009	6 1 9	6 1997 29	140001	8 1 1	8 1997 21	160033	9 4 3	9 2000 23
20026	2 3 6	2 1999 26	60018	4 2 8	4 1998 28	100010	6 1 10	6 1997 30	140002	8 1 2	8 1997 22	160034	9 4 4	9 2000 24
20027	2 3 7	2 1999 27	60019	4 2 9	4 1998 29	100011	6 2 1	6 1998 21	140003	8 1 3	8 1997 23	160035	9 4 5	9 2000 25
20028	2 3 8	2 1999 28	60020	4 2 10	4 1998 30	100012	6 2 2	6 1998 22	140004	8 1 4	8 1997 24	160036	9 4 6	9 2000 26
20029	2 3 9	2 1999 29	60021	4 3 1	4 1999 21	100013	6 2 3	6 1998 23	140005	8 1 5	8 1997 25	160037	9 4 7	9 2000 27
20030	2 3 10	2 1999 30	60022	4 3 2	4 1999 22	100014	6 2 4	6 1998 24	140006	8 1 6	8 1997 26	160038	9 4 8	9 2000 28
20031	2 4 1	2 2000 21	60023	4 3 3	4 1999 23	100015	6 2 5	6 1998 25	140007	8 1 7	8 1997 27	160039	9 4 9	9 2000 29
20032	2 4 2	2 2000 22	60024	4 3 4	4 1999 24	100016	6 2 6	6 1998 26	140008	8 1 8	8 1997 28	160040	9 4 10	9 2000 30
20033	2 4 3	2 2000 23	60025	4 3 5	4 1999 25	100017	6 2 7	6 1998 27	140009	8 1 9	8 1997 29	180001	10 1 1	10 1997 21
20034	2 4 4	2 2000 24	60026	4 3 6	4 1999 26	100018	6 2 8	6 1998 28	140010	8 1 10	8 1997 30	180002	10 1 2	10 1997 22
20035	2 4 5	2 2000 25	60027	4 3 7	4 1999 27	100019	6 2 9	6 1998 29	140011	8 2 1	8 1998 21	180003	10 1 3	10 1997 23
20036	2 4 6	2 2000 26	60028	4 3 8	4 1999 28	100020	6 2 10	6 1998 30	140012	8 2 2	8 1998 22	180004	10 1 4	10 1997 24
20037	2 4 7	2 2000 27	60029	4 3 9	4 1999 29	100021	6 3 1	6 1999 21	140013	8 2 3	8 1998 23	180005	10 1 5	10 1997 25
20038	2 4 8	2 2000 28	60030	4 3 10	4 1999 30	100022	6 3 2	6 1999 22	140014	8 2 4	8 1998 24	180006	10 1 6	10 1997 26
20039	2 4 9	2 2000 29	60031	4 4 1	4 2000 21	100023	6 3 3	6 1999 23	140015	8 2 5	8 1998 25	180007	10 1 7	10 1997 27
20040	2 4 10	2 2000 30	60032	4 4 2	4 2000 22	100024	6 3 4	6 1999 24	140016	8 2 6	8 1998 26	180008	10 1 8	10 1997 28
40001	3 1 1	3 1997 21	60033	4 4 3	4 2000 23	100025	6 3 5	6 1999 25	140017	8 2 7	8 1998 27	180009	10 1 9	10 1997 29
40002	3 1 2	3 1997 22	60034	4 4 4	4 2000 24	100026	6 3 6	6 1999 26	140018	8 2 8	8 1998 28	180010	10 1 10	10 1997 30
40003	3 1 3	3 1997 23	60035	4 4 5	4 2000 25	100027	6 3 7	6 1999 27	140019	8 2 9	8 1998 29	180011	10 2 1	10 1998 21
40004	3 1 4	3 1997 24	60036	4 4 6	4 2000 26	100028	6 3 8	6 1999 28	140020	8 2 10	8 1998 30	180012	10 2 2	10 1998 22
40005	3 1 5	3 1997 25	60037	4 4 7	4 2000 27	100029	6 3 9	6 1999 29	140021	8 3 1	8 1999 21	180013	10 2 3	10 1998 23
40006	3 1 6	3 1997 26	60038	4 4 8	4 2000 28	100030	6 3 10	6 1999 30	140022	8 3 2	8 1999 22	180014	10 2 4	10 1998 24
40007	3 1 7	3 1997 27	60039	4 4 9	4 2000 29	100031	6 4 1	6 2000 21	140023	8 3 3	8 1999 23	180015	10 2 5	10 1998 25
40008	3 1 8	3 1997 28	60040	4 4 10	4 2000 30	100032	6 4 2	6 2000 22	140024	8 3 4	8 1999 24	180016	10 2 6	10 1998 26
40009	3 1 9	3 1997 29	80001	5 1 1	5 1997 21	100033	6 4 3	6 2000 23	140025	8 3 5	8 1999 25	180017	10 2 7	10 1998 27
40010	3 1 10	3 1997 30	80002	5 1 2	5 1997 22	100034	6 4 4	6 2000 24	140026	8 3 6	8 1999 26	180018	10 2 8	10 1998 28
40011	3 2 1	3 1998 21	80003	5 1 3	5 1997 23	100035	6 4 5	6 2000 25	140027	8 3 7	8 1999 27	180019	10 2 9	10 1998 29
40012	3 2 2	3 1998 22	80004	5 1 4	5 1997 24	100036	6 4 6	6 2000 26	140028	8 3 8	8 1999 28	180020	10 2 10	10 1998 30
40013	3 2 3	3 1998 23	80005	5 1 5	5 1997 25	100037	6 4 7	6 2000 27	140029	8 3 9	8 1999 29	180021	10 3 1	10 1999 21
40014	3 2 4	3 1998 24	80006	5 1 6	5 1997 26	100038	6 4 8	6 2000 28	140030	8 3 10	8 1999 30	180022	10 3 2	10 1999 22
40015	3 2 5	3 1998 25	80007	5 1 7	5 1997 27	100039	6 4 9	6 2000 29	140031	8 4 1	8 2000 21	180023	10 3 3	10 1999 23
40016	3 2 6	3 1998 26	80008	5 1 8	5 1997 28	100040	6 4 10	6 2000 30	140032	8 4 2	8 2000 22	180024	10 3 4	10 1999 24
40017	3 2 7	3 1998 27	80009	5 1 9	5 1997 29	120001	7 1 1	7 1997 21	140033	8 4 3	8 2000 23	180025	10 3 5	10 1999 25
40018	3 2 8	3 1998 28	80010	5 1 10	5 1997 30	120002	7 1 2	7 1997 22	140034	8 4 4	8 2000 24	180026	10 3 6	10 1999 26
40019	3 2 9	3 1998 29	80011	5 2 1	5 1998 21	120003	7 1 3	7 1997 23	140035	8 4 5	8 2000 25	180027	10 3 7	10 1999 27
40020	3 2 10	3 1998 30	80012	5 2 2	5 1998 22	120004	7 1 4	7 1997 24	140036	8 4 6	8 2000 26	180028	10 3 8	10 1999 28
40021	3 3 1	3 1999 21	80013	5 2 3	5 1998 23	120005	7 1 5	7 1997 25	140037	8 4 7	8 2000 27	180029	10 3 9	10 1999 29
40022	3 3 2	3 1999 22	80014	5 2 4	5 1998 24	120006	7 1 6	7 1997 26	140038	8 4 8	8 2000 28	180030	10 3 10	10 1999 30
40023	3 3 3	3 1999 23	80015	5 2 5	5 1998 25	120007	7 1 7	7 1997 27	140039	8 4 9	8 2000 29	180031	10 4 1	10 2000 21
40024	3 3 4	3 1999 24	80016	5 2 6	5 1998 26	120008	7 1 8	7 1997 28	140040	8 4 10	8 2000 30	180032	10 4 2	10 2000 22
40025	3 3 5	3 1999 25	80017	5 2 7	5 1998 27	120009	7 1 9	7 1997 29	160001	9 1 1	9 1997 21	180033	10 4 3	10 2000 23
40026	3 3 6	3 1999 26	80018	5 2 8	5 1998 28	120010	7 1 10	7 1997 30	160002	9 1 2	9 1997 22	180034	10 4 4	10 2000 24
40027	3 3 7	3 1999 27	80019	5 2 9	5 1998 29	120011	7 2 1	7 1998 21	160003	9 1 3	9 1997 23	180035	10 4 5	10 2000 25
40028	3 3 8	3 1999 28	80020	5 2 10	5 1998 30	120012	7 2 2	7 1998 22	160004	9 1 4	9 1997 24	180036	10 4 6	10 2000 26
40029	3 3 9	3 1999 29	80021	5 3 1	5 1999 21	120013	7 2 3	7 1998 23	160005	9 1 5	9 1997 25	180037	10 4 7	10 2000 27
40030	3 3 10	3 1999 30	80022	5 3 2	5 1999 22	120014	7 2 4	7 1998 24	160006	9 1 6	9 1997 26	180038	10 4 8	10 2000 28
40031	3 4 1	3 2000 21	80023	5 3 3	5 1999 23	120015	7 2 5	7 1998 25	160007	9 1 7	9 1997 27	180039	10 4 9	10 2000 29
40032	3 4 2	3 2000 22	80024	5 3 4	5 1999 24	120016	7 2 6	7 1998 26	160008	9 1 8	9 1997 28	180040	10 4 10	10 2000 30
40033	3 4 3	3 2000 23	80025	5 3 5	5 1999 25	120017	7 2 7	7 1998 27	160009	9 1 9	9 1997 29			
40034	3 4 4	3 2000 24	80026	5 3 6	5 1999 26	120018	7 2 8	7 1998 28	160010	9 1 10	9 1997 30			
40035	3 4 5	3 2000 25	80027	5 3 7	5 1999 27	120019	7 2 9	7 1998 29	160011	9 2 1	9 1998 21			
40036	3 4 6	3 2000 26	80028	5 3 8	5 1999 28	120020	7 2 10	7 1998 30	160012	9 2 2	9 1998 22			
40037	3 4 7	3 2000 27	80029	5 3 9	5 1999 29	120021	7 3 1	7 1999 21	160013	9 2 3	9 1998 23			
40038	3 4 8	3 2000 28	80030	5 3 10	5 1999 30	120022	7 3 2	7 1999 22	160014	9 2 4	9 1998 24			
40039	3 4 9	3 2000 29	80031	5 4 1	5 2000 21	120023	7 3 3	7 1999 23	160015	9 2 5	9 1998 25			
40040	3 4 10	3 2000 30	80032	5 4 2	5 2000 22	120024	7 3 4	7 1999 24	160016	9 2 6	9 1998 26			
60001	4 1 1	4 1997 21	80033	5 4 3	5 2000 23	120025	7 3 5	7 1999 25	160017	9 2 7	9 1998 27			
60002	4 1 2	4 1997 22	80034	5 4 4	5 2000 24	120026	7 3 6	7 1999 26	160018	9 2 8	9 1998 28			
60003	4 1 3	4 1997 23	80035	5 4 5	5 2000 25	120027	7 3 7	7 1999 27	160019	9 2 9	9 1998 29			
60004	4 1 4	4 1997 24	80036	5 4 6	5 2000 26	120028	7 3 8	7 1999 28	160020	9 2 10	9 1998 30			
60005	4 1 5	4 1997 25	80037	5 4 7	5 2000 27	120029	7 3 9	7 1999 29	160021	9 3 1	9 1999 21			
60006	4 1 6	4 1997 26	80038	5 4 8	5 2000 28	120030	7 3 10	7 1999 30	160022	9 3 2	9 1999 22			

TOTAL DE VENDAS (VOLUME DE DADOS 3.000.000)

	Descrição da Consulta	Tempo Medido em Segundos							
		PROTÓTIPO MOLAP				ESQUEMA ESTRELA			
		Número de Execuções da Consulta							
		1ª	2ª	3ª	Média	1ª	2ª	3ª	Média
Total de Vendas por Mês	1) Produto (01-10) / Loja (01-05) / Tempo (Jan-2001)	0.640	0.541	0.439	**0.591**	108.356	99.673	99.571	**102.533**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mai-2001)	1.793	1.743	1.641	**1.768**	209.25	212.636	212.533	**211.473**
	3) Produto (15-45) / Loja (01-10) / Tempo (Ago-2001)	3.405	3.175	3.074	**3.290**	316.225	317.296	317.192	**316.904**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mar-2002)	6.259	5.748	5.649	**6.004**	434.886	437.449	437.344	**436.560**
	5) Produto (01-50) / Loja (15-30) / Tempo (Out-2002)	8.732	7.911	7.809	**8.322**	535.069	531.404	531.3	**532.591**
	6) Produto (10-20) / Loja (11-25) / Tempo (Dez-2002)	3.775	1.762	1.664	**2.769**	116.618	116.687	116.584	**116.630**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mar-2003)	5.198	4.446	4.344	**4.822**	232.845	226.045	226.048	**228.313**
	8) Produto (20-50) / Loja (03-27) / Tempo (Ago-2003)	9.133	7.791	7.686	**8.462**	344.716	352.407	352.302	**349.808**
	9) Produto (10-50) / Loja (02-29) / Tempo (Mai-2004)	12.318	11.596	11.494	**11.957**	455.014	462.074	462.073	**459.720**
	10) Produto (1-50) / Loja (01-30) / Tempo (Out-2005)	15.753	16.003	16.012	**15.878**	1.372	0.921	0.817	**1.037**
	Média das Vendas por Mês	**6.701**	**6.072**	**5.981**	**6.386**	**275.435**	**275.659**	**275.576**	**275.557**
Total de Vendas por Trimestre	1) Produto (01-10) / Loja (01-05) / Tempo (Jan-2001 à Mar-2001)	1.753	1.562	1.459	**1.658**	98.712	98.261	98.157	**98.377**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mai-2001 à Jul-2001)	5.488	5.067	5.063	**5.278**	203.983	205.966	205.870	**205.273**
	3) Produto (15-45) / Loja (01-10) / Tempo (Ago-2001 à Out-2001)	10.185	9.253	9.155	**9.719**	308.203	312.730	312.628	**311.187**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mar-2002 à Mai-2002)	18.046	17.084	17.089	**17.565**	428.396	420.475	420.371	**423.081**
	5) Produto (01-50) / Loja (15-30) / Tempo (Out-2002 à Dez-2002)	24.365	23.734	23.631	**24.050**	529.502	529.000	528.008	**528.837**
	6) Produto (10-20) / Loja (11-25) / Tempo (Dez-2002 à Fev-2003)	6.099	5.127	5.025	**5.613**	120.023	114.304	114.202	**116.176**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mar-2003 à Mai-2003)	14.691	13.359	13.257	**14.025**	225.675	260.084	260.081	**248.613**
	8) Produto (20-50) / Loja (03-27) / Tempo (Ago-2003 à Out-2003)	23.203	23.313	23.211	**23.258**	396.080	340.340	340.243	**358.888**
	9) Produto (10-50) / Loja (2-29) / Tempo (Mai-2004 à Jul-2004)	35.732	33.999	33.898	**34.866**	452.701	457.588	457.485	**455.925**
	10) Produto (1-50) / Loja (1-30) / Tempo (Out-2005 à Dez-2005)	46.827	44.344	44.243	**45.586**	1.712	1.161	1.065	**1.313**
	Média das Vendas por Trimeste	**18.639**	**17.684**	**17.603**	**18.162**	**276.499**	**273.991**	**273.811**	**274.767**

TOTAL DE VENDAS (VOLUME DE DADOS 3.000.000)

	Descrição da Consulta	Tempo Medido em Segundos							
		PROTÓTIPO MOLAP				ESQUEMA ESTRELA			
		Número de Execuções da Consulta							
		1ª	2ª	3ª	Média	1ª	2ª	3ª	Média
Total de Vendas durante quatro Meses	1) Produto (01-10) / Loja (01-05) / Tempo (Jan-2001 à Abr-2001)	2.413	2.043	2.041	**2.228**	94.606	90.831	90.730	**92.056**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mai-2001 à Ago-2001)	8.082	6.810	6.713	**7.446**	196.322	194.960	194.856	**195.379**
	3) Produto (15-45) / Loja (01-10) / Tempo (Ago-2001 à Nov-2001)	12.898	12.268	12.164	**12.583**	290.527	293.942	293.840	**292.770**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mar-2002 à Jun-2002)	24.485	22.693	22.590	**23.589**	400.105	405.734	405.631	**403.823**
	5) Produto (01-50) / Loja (15-30) / Tempo (Out-2002 à Jan-2003)	30.684	28.521	28.422	**29.603**	497.265	500.960	500.855	**499.693**
	6) Produto (10-20) / Loja (11-25) / Tempo (Dez-2002 à Mar-2003)	7.631	6.900	6.810	**7.266**	108.556	109.818	109.714	**109.363**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mar-2003 à Jun-2003)	18.276	17.626	17.524	**17.951**	256.900	219.145	219.042	**231.696**
	8) Produto (20-50) / Loja (03-27) / Tempo (Ago-2003 à Nov-2003)	31.826	30.724	30.622	**31.275**	362.060	379.225	379.127	**373.471**
	9) Produto (10-50) / Loja (02-29) / Tempo (Mai-2004 à Ago-2004)	46.317	45.476	45.373	**45.897**	447.890	443.633	443.531	**445.018**
	10) Produto (1-50) / Loja (01-30) / Tempo (Set-2005 à Dez-2005)	60.046	58.875	58.774	**59.461**	1.933	1.282	1.178	**1.464**
	Média das Vendas durante quatro Meses	**24.266**	**23.194**	**23.103**	**23.730**	**265.616**	**263.953**	**263.850**	**264.473**
Total de Vendas por Semestre	1) Produto (01-10) / Loja (01-05) / Tempo (Jan-2001 à Jun-2001)	3.305	3.135	3.030	**3.220**	93.835	90.740	90.637	**91.737**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mai-2001 à Out-2001)	12.328	10.365	10.261	**11.347**	193.498	207.909	207.802	**203.070**
	3) Produto (15-45) / Loja (01-10) / Tempo (Ago-2001 à Jan-2002)	20.259	18.767	18.668	**19.513**	285.291	290.107	290.004	**288.467**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mar-2002 à Ago-2002)	36.112	34.409	34.310	**35.261**	391.683	396.210	396.113	**394.669**
	5) Produto (01-50) / Loja (15-30) / Tempo (Out-2002 à Fev-2003)	50.152	46.898	46.794	**48.525**	486.780	494.180	494.077	**491.679**
	6) Produto (10-20) / Loja (11-25) / Tempo (Dez-2002 à Mai-2003)	11.577	10.394	10.291	**10.986**	115.816	107.695	107.592	**110.368**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mar-2003 à Ago-2003)	30.143	27.339	27.236	**28.741**	211.985	213.166	213.061	**212.737**
	8) Produto (20-50) / Loja (03-27) / Tempo (Ago-2003 à Jan-2004)	51.494	48.660	48.560	**50.077**	318.979	314.853	314.755	**316.196**
	9) Produto (10-50) / Loja (02-29) / Tempo (Mai-2004 à Out-2004)	73.225	69.730	69.625	**71.478**	425.822	421.516	421.413	**422.917**
	10) Produto (1-50) / Loja (01-30) / Tempo (Jul-2005 à Dez-2005)	93.945	90.510	90.410	**92.228**	1.923	1.492	1.590	**1.668**
	Média das Vendas por Semestre	**38.254**	**36.021**	**35.919**	**37.137**	**252.561**	**253.787**	**253.704**	**253.351**

TOTAL DE VENDAS (VOLUME DE DADOS 3.000.000)

	Descrição da Consulta	Tempo Medido em Segundos							
		PROTÓTIPO MOLAP				ESQUEMA ESTRELA			
		Número de Execuções da Consulta							
		1ª	2ª	3ª	Média	1ª	2ª	3ª	Média
Total de Vendas durante Sete Meses	1) Produto (01-10) / Loja (01-05) / Tempo (Jan-2001 à Jul-2001)	4.777	3.826	3.724	**4.302**	87.356	86.745	86.642	**86.914**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mai-2001 à Nov-2001)	13.719	12.198	12.096	**12.959**	187.109	185.156	185.052	**185.772**
	3) Produto (15-45) / Loja (01-10) / Tempo (Ago-2001 à Fev-2002)	23.374	21.791	21.694	**22.583**	278.952	277.459	277.354	**277.922**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mar-2002 à Set-2002)	42.300	40.508	40.409	**41.404**	380.347	383.882	383.785	**382.671**
	5) Produto (01-50) / Loja (15-30) / Tempo (Out-2002 à Mar-2003)	59.025	56.181	56.077	**57.603**	468.614	473.972	473.870	**472.152**
	6) Produto (10-20) / Loja (11-25) / Tempo (Dez-2002 à Jun-2003)	13.389	12.699	12.597	**13.044**	102.477	103.058	103.051	**102.862**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mar-2003 à Set-2003)	32.657	31.886	31.780	**32.272**	205.155	202.571	202.473	**203.400**
	8) Produto (20-50) / Loja (03-27) / Tempo (Ago-2003 à Fev-2004)	57.904	54.528	54.421	**56.216**	306.811	306.631	306.529	**306.657**
	9) Produto (10-50) / Loja (02-29) / Tempo (Mai-2004 à Nov-2004)	83.921	81.137	81.032	**82.529**	410.820	409.699	409.595	**410.038**
	10) Produto (1-50) / Loja (1-30) / Tempo (Jun-2005 à Dez-2005)	108.366	105.111	105.010	**106.739**	1.912	1.622	1.520	**1.685**
	Média das Vendas durante sete Meses	**43.943**	**41.987**	**41.884**	**42.965**	**242.955**	**243.080**	**242.987**	**243.007**
Total de Vendas por Ano	1) Produto (01-10) / Loja (01-05) / Tempo (Jan-2001 à Dez-2001)	7.881	6.369	6.264	**7.125**	94.947	86.415	86.311	**89.224**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mai-2001 à Abr-2002)	22.732	20.339	20.235	**21.536**	188.831	184.726	184.622	**186.060**
	3) Produto (15-45) / Loja (01-10) / Tempo (Ago-2001 à Jul-2002)	39.828	37.834	37.731	**38.831**	275.837	270.809	270.704	**272.450**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mar-2002 à Fev-2003)	71.633	68.589	68.486	**70.111**	380.878	378.344	378.247	**379.156**
	5) Produto (01-50) / Loja (15-30) / Tempo (Out-2002 à Set-2003)	99.393	94.916	94.810	**97.155**	469.555	459.491	459.390	**462.812**
	6) Produto (10-20) / Loja (11-25) / Tempo (Dez-2002 à Nov-2003)	22.222	21.290	21.191	**21.756**	103.910	115.416	115.312	**111.546**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mar-2003 à Fev-2004)	57.673	54.448	54.344	**56.061**	212.917	213.477	213.373	**213.256**
	8) Produto (20-50) / Loja (03-27) / Tempo (Ago-2003 à Jul-2004)	98.302	94.586	94.483	**96.444**	311.367	310.196	310.095	**310.553**
	9) Produto (10-50) / Loja (02-29) / Tempo (Mai-2004 à Abr-2005)	143.686	139.630	139.520	**141.658**	409.519	414.066	414.060	**412.548**
	10) Produto (1-50) / Loja (1-30) / Tempo (Jan-2005 à Dez-2005)	183.594	179.969	179.862	**181.782**	2.847	2.153	2.052	**2.351**
	Média das Vendas por Ano	**74.694**	**71.797**	**71.693**	**73.246**	**245.061**	**243.509**	**243.417**	**243.996**

TOTAL DE VENDAS (VOLUME DE DADOS 3.000.000)

	Descrição da Consulta	Tempo Medido em Segundos							
		PROTÓTIPO MOLAP				ESQUEMA ESTRELA			
		Número de Execuções da Consulta							
		1ª	2ª	3ª	Média	1ª	2ª	3ª	Média
Total de Vendas durante dois Anos	1) Produto (01-10) / Loja (01-05) / Tempo (Jan-2001 à Dez-2002)	14.591	12.829	12.721	**13.710**	100.104	98.762	98.660	**99.175**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mai-2001 à Abr-2003)	45.235	40.318	40.213	**42.777**	212.035	215.850	215.753	**214.546**
	3) Produto (15-45) / Loja (01-10) / Tempo (Ago-2001 à Jul-2003)	76.610	73.956	73.855	**75.283**	322.453	325.088	325.082	**324.208**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mar-2002 à FEB-2004)	138.720	136.777	136.671	**137.749**	439.692	442.016	442.013	**441.240**
	5) Produto (01-50) / Loja (15-30) / Tempo (Out-2002 à SEP-2004)	194.630	190.063	190.060	**192.347**	542.881	547.787	547.684	**546.117**
	6) Produto (10-20) / Loja (11-25) / Tempo (Dez-2002 à Nov-2004)	42.531	40.259	40.155	**41.395**	117.489	119.181	119.080	**118.583**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mar-2003 à Fev-2005)	110.970	108.596	108.495	**109.783**	232.134	232.624	232.522	**232.427**
	8) Produto (20-50) / Loja (03-27) / Tempo (Ago-2003 à Jul-2005)	190.524	185.767	185.669	**188.146**	352.046	354.049	354.048	**353.381**
	9) Produto (10-50) / Loja (02-29) / Tempo (Mai-2003 à Abr-2005)	273.864	272.453	272.351	**273.159**	468.504	470.556	470.455	**469.838**
	10) Produto (1-50) / Loja (1-30) / Tempo (Jan-2003 à Dez-2004)	356.072	354.690	354.584	**355.381**	571.772	577.560	577.456	**575.596**
	Média das Vendas durante dois Anos	**144.375**	**141.571**	**141.477**	**142.973**	**335.911**	**338.347**	**338.275**	**337.511**
Total de Vendas durante Quatro Anos	1) Produto (01-10) / Loja (01-5) / Tempo (Jan-2001 à Dez-2004)	28.741	22.216	22.112	**25.479**	101.766	100.564	100.460	**100.930**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mai-2001 à Abr-2004)	89.470	83.553	83.451	**86.512**	271.363	218.634	218.531	**236.176**
	3) Produto (15-45) / Loja (01-10) / Tempo (Ago-2001 à Jul-2004)	149.220	145.566	145.463	**147.393**	327.070	322.333	322.230	**323.878**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mar-2002 à FEB-2005)	273.440	268.497	268.399	**270.969**	445.791	440.824	440.722	**442.446**
	5) Produto (01-50) / Loja (15-30) / Tempo (Out-2002 à Set-2005)	381.260	378.693	378.591	**379.977**	549.099	549.671	549.569	**549.446**
	6) Produto (10-20) / Loja (11-25) / Tempo (Dez-2002 à Nov-2005)	84.062	82.790	82.685	**83.426**	117.910	115.876	115.775	**116.520**
	7) Produto (25-45) / Loja (05-25) / Tempo (Jan-2001 à Dez-2004)	218.940	214.566	214.461	**216.753**	234.377	235.448	235.341	**235.055**
	8) Produto (20-50) / Loja (03-27) / Tempo (Mai-2001 à Abr-2004)	379.048	376.291	376.193	**377.670**	356.933	357.634	357.533	**357.367**
	9) Produto (10-50) / Loja (02-29) / Tempo (Ago-2001 à Jul-2004)	545.728	541.317	541.211	**543.523**	473.771	475.134	475.030	**474.645**
	10) Produto (1-50) / Loja (1-30) / Tempo (Mar-2002 à Fev-2005)	710.920	701.430	701.327	**706.175**	583.088	582.277	582.073	**582.479**
	Média das Vendas durante dois Anos	**286.083**	**281.492**	**281.389**	**283.787**	**346.117**	**339.840**	**339.726**	**341.894**

MÉDIA DE VENDAS AGRUPADA POR PRODUTO E ANO (VOLUME DE DADOS 3.000.000)

	Descrição da Consulta	Tempo Medido em Segundos							
		PROTÓTIPO MOLAP				ESQUEMA ESTRELA			
		Número de Execuções da Consulta							
		1ª	2ª	3ª	Média	1ª	2ª	3ª	Média
Média de Vendas por Mês	1) Produto (01-10) / Loja (01-05) / Tempo (Mês Jan e Ano 2001-2005)	16.403	15.422	15.320	**15.715**	79.505	80.836	80.730	**80.357**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mês Fev e Ano 2001-2005)	45.065	44.133	44.028	**44.409**	179.248	175.993	175.891	**177.044**
	3) Produto (15-45) / Loja (1-10) / Tempo (Mês Mar e Ano 2001-2005)	91.161	88.367	88.261	**89.263**	266.183	262.227	262.123	**263.511**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mês Abr e Ano 2001-2005)	156.304	152.209	152.102	**153.538**	359.447	358.836	358.734	**359.006**
	5) Produto (01-50) / Loja (15-30) / Tempo (Mês Mai e Ano 2001-2005)	225.935	219.926	219.823	**221.895**	449.476	450.507	450.409	**450.131**
	6) Produto (10-20) / Loja (11-25) / Tempo (Mês Jun e Ano 2001-2005)	47.428	44.113	44.010	**45.184**	97.460	97.871	97.765	**97.699**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mês Jul e Ano 2001-2005)	129.807	120.473	120.369	**123.550**	199.597	196.633	196.529	**197.586**
	8) Produto (20-50) / Loja (03-27) / Tempo (Mês Ago e Ano 2001-2005)	221.228	208.240	208.138	**212.535**	299.651	300.041	300.036	**299.909**
	9) Produto (10-50) / Loja (02-29) / Tempo (Mês Set e Ano 2001-2005)	319.199	301.483	301.385	**307.356**	403.931	404.392	404.289	**404.204**
	10) Produto (1-50) / Loja (01-30) / Tempo (Mês Out e Ano 2001-2005)	412.954	399.865	399.764	**404.194**	495.062	500.069	500.072	**498.401**
	Média das Vendas Agrupadas (produto e ano) por Mês	**166.548**	**159.423**	**159.320**	**161.764**	**282.956**	**282.741**	**282.658**	**282.785**
Média de Vendas por Trimestre	1) Produto (01-10) / Loja (01-05) / Tempo (Mês Jan-Mar e Ano 2001-2005)	47.388	44.964	44.860	**45.737**	80.435	80.696	80.591	**80.574**
	2) Produto (20-40) / Loja (13-20) / Tempo (Mês Abr-Jun e Ano 2001-2005)	146.721	141.794	141.691	**143.402**	175.262	178.737	178.633	**177.544**
	3) Produto (15-45) / Loja (01-10) / Tempo (Mês Jul-Set e Ano 2001-2005)	268.546	261.185	261.083	**263.605**	274.735	268.066	268.064	**270.288**
	4) Produto (05-45) / Loja (07-20) / Tempo (Mês Out-Dez e Ano 2001-2005)	479.890	477.523	477.419	**478.277**	369.221	363.773	363.670	**365.555**
	5) Produto (01-50) / Loja (15-30) / Tempo (Mês Fev-Abr e Ano 2001-2005)	657.796	630.116	630.010	**639.307**	454.954	454.474	454.367	**454.598**
	6) Produto (10-20) / Loja (11-25) / Tempo (Mês Mai-Jul e ano 2001-2005)	155.694	135.374	135.267	**142.112**	97.430	98.242	98.139	**97.937**
	7) Produto (25-45) / Loja (05-25) / Tempo (Mês Ago-Out e Ano 2001-2005)	391.934	361.412	361.315	**371.554**	197.855	197.914	197.811	**197.860**
	8) Produto (20-50) / Loja (03-27) / Tempo (Mês Mar-Mai e Ano 2001-2005)	649.474	619.796	619.699	**629.656**	300.612	300.702	300.599	**300.638**
	9) Produto (10-50) / Loja (02-29) / Tempo (Mês Jun-Ago e Ano 2001-2005)	1,173.327	1,166.968	1,066.962	**1,135.752**	403.861	406.744	406.641	**405.749**
	10) Produto (1-50) / Loja (1-30) / Tempo (Mês Set-Nov e Ano 2001-2005)	1,726.392	1,719.527	1,619.523	**1,688.481**	501.501	502.553	502.450	**502.168**
	Média das Vendas Agrupadas (produto e ano) por Trimeste	**569.716**	**555.866**	**535.783**	**553.788**	**285.587**	**285.190**	**285.097**	**285.291**