Classificação dos domínios transacionais e analíticos para avaliação de alternativas de sincronismo em projetos de data warehouse

Isabel Cristina Italiano

Dissertação apresentada ao
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
da
UNIVERSIDADE DE SÃO PAULO
para obtenção do grau de mestre em Ciência da Computação

Área de Concentração: **Bancos de Dados**

Orientador: **Prof. Dr. João Eduardo Ferreira**

São Paulo - Junho de 2002

Classificação dos domínios transacionais e analíticos
para avaliação de alternativas de sincronismo
em projetos de data warehouse

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Isabel Cristina Italiano
e aprovada pela comissão julgadora.

São Paulo, junho de 2002.

Banca examinadora:

- Prof. Dr. João Eduardo Ferreira (orientador) (IME-USP)
- Prof. Dr. Roberto Marcondes Cesar Junior (IME-USP)
- Prof. Dr. Jorge Rady de Almeida Junior (Escola Politécnica - USP)

Este trabalho é dedicado ao meu pai (in memoriam) e à minha mãe.

Abstract

The importance and use of the data warehouse has been growing rapidly in the last few years. It plays a relevant role as a base for decision support systems, through analytical applications deployment, whether in business or academic scene. The synchronism pattern between transactional and analytical environment, usually implemented through periodical loads is not adequate for some business areas that require more and more decision making processes with fast and accurate responses. This requirement needs a more dynamic synchronism option, that involves keeping these environments synchronized in real time or smaller time intervals than those implemented by static loads performed periodically.

The purpose of our present work is to define the most suitable synchronism option for an specific business model, using a parameter set. The parameter set definition was based on the analysis of the characteristics related to data warehouse information portions that take part in transactional and analytical environment. The work also defines a function in order to evaluate those parameters implemented in decision trees. The function chooses the most suitable synchronism option to an specific information portion.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Warehousing[2] é uma técnica utilizada para recuperação e integração de dados a partir de fontes distribuídas, autônomas e, possivelmente, heterogêneas [ZGMHW94].

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

Os data warehouses vêm sendo muito utilizados pelas empresas, já que proporcionam um alicerce sólido de integração de dados corporativos e históricos para a realização de análises gerenciais. Sua construção e implementação são feitas de uma maneira passo a passo, organizando e armazenado os dados sob uma perspectiva de longo prazo. Assim, partindo-se dos dados históricos básicos, pode-se realizar análises de tendências [IH97].

Por sua característica básica, que é a integração de dados provenientes de várias fontes diferentes, a etapa mais complexa na implementação de um data warehouse é o processo de carga. Por várias fontes de dados entendemos as bases de dados de produção, que contêm os dados utilizados pelos vários sistemas transacionais de uma empresa. Neste processo, os dados distribuídos pelos vários ambientes operacionais devem ser selecionados, trabalhados com o objetivo de padronização e limpeza, transferidos para o novo

[2] O termo *Warehousing* não possui tradução adequada para o português.

ambiente e finalmente carregados, sempre atendendo ao padrão da modelagem utilizada para o data warehouse. Este processo é feito periodicamente, sendo que sua freqüência depende de vários fatores relacionados ao modelo de negócios utilizado pela empresa e, normalmente, não é menor que 24 horas. Desta forma, podemos dizer que os dados armazenados no data warehouse são, para todos os propósitos práticos, uma longa série de fotografias, tiradas ao longo do tempo. Uma vez que os dados são armazenados no data warehouse, eles não mais sofrem atualizações, sendo, portanto, um ambiente apenas de carga e acesso.

Após sua criação e primeira carga, o data warehouse passa a sofrer cargas incrementais que devem refletir o ambiente operacional ao longo do tempo tornando-o uma imensa base de dados para os sistemas de apoio à decisão.

Torna-se claro, portanto, que o data warehouse tem características de um ambiente estático, que só reflete as alterações ocorridas no ambiente operacional após períodos pré-definidos. Este fato em si não é tão grave, já que em várias áreas de negócio, as análises são feitas baseadas em resumos mensais, por exemplo. O que é um fator crítico é a alta complexidade do processo de carga que se transforma em um ponto muito suscetível à introdução de erros, que podem levar ao colapso de todo o processo de tomada de decisão.

Como o data warehouse sofre periodicamente novas cargas, aumentando constantemente seu tamanho, surge mais um grande problema em sua utilização, que é a dificuldade em responder às consultas dos usuários de forma rápida e eficiente. As otimizações nos componentes que envolvem este processo, visando agilizar as respostas às consultas, vêm sendo projetadas em vários níveis:

1. mudanças nas estruturas físicas que compõem a base de dados;
2. ferramentas de análise mais eficientes e
3. melhorias no modelo de dados implementado no data warehouse;

Os itens 1 e 2 acima têm sido tratados pelos fornecedores de bancos de dados e de ferramentas de análise, respectivamente e, portanto, devemos nos concentrar no item 3.

Os problemas citados anteriormente, ou seja, a alta complexidade do processo de carga aliada às características estáticas do data warehouse e a necessidade de melhorar o modelo de dados implementado, são a motivação para este trabalho.

Com o objetivo de conferir ao data warehouse uma característica mais dinâmica e otimizar o acesso aos dados, vários estudos têm sido feitos indicando o uso de *visões materializadas*[3] como alternativa viável.

Uma visão é uma relação derivada, definida em termos de relações base, que é computada todas as vezes em que uma referência a ela é feita. Uma visão é dita materializada quando ela é realmente armazenada na base de dados em vez de ser computada a partir das relações base em resposta a consultas [QGMW97]. Uma visão materializada pode ser vista como um *cache*[4] - uma cópia dos dados que pode ser acessada rapidamente. Os data warehouses podem, portanto, armazenar estas visões materializadas com o objetivo de possibilitar acesso rápido à informação que está integrada a partir de diferentes fontes de dados distribuídas [DEB95].

Existem duas abordagens diferentes no que se refere à utilização de visões materializadas nos data warehouses. A primeira delas define o próprio data warehouse como um conjunto de visões materializadas baseadas nos dados dos ambientes operacionais [QGMW97, Gup97]. A segunda abordagem propõe um conjunto compartilhado de visões materializadas definidas a partir de uma análise das consultas mais freqüentes executadas no data warehouse. Estas visões, por estarem materializadas, agilizariam o acesso aos dados necessários para a realização da consulta [YKL96].

O uso de visões materializadas nas abordagens citadas acima traz vários aspectos que serão cuidadosamente analisados, entre eles:

1. a seleção das visões mais adequadas levando-se em conta uma análise dos custos de manutenção e os benefícios de cada visão e os algoritmos disponíveis;

2. os mecanismos que permitem a propagação correta quando da atualização das fontes

[3] A expressão visões materializadas foi traduzida do inglês materialized views.

[4] Estrutura de dados em memória que visa manter disponível dados com grande possibilidade de reutilização, minimizando repetidos acessos à mesma informação armazenada em disco

de dados base para vários tipos de visões, preferencialmente de forma online;

3. a manutenção das visões de forma dinâmica e preferencialmente autônoma utilizando o conceito de *self maintainable views*[5].

Um data warehouse, que tem por objetivo atender a área de tomada de decisões, envolve informações provenientes de um conjunto de aplicações transacionais que têm características operacionais e requisitos de negócios bem distintos entre si. Fica claro que existe uma heterogeneidade no grau de atualização das informações e, portanto, o data warehouse não pode implementar apenas um modelo de sincronismo, estático ou dinâmico.

Os trabalhos existentes, relacionados a este tema, até a presente data, não levam em conta a possibilidade de se implementar um modelo de sincronismo híbrido, já que partes ou porções do data warehouse têm comportamentos e requisitos diferentes. Além disso, em nenhum dos trabalhos pesquisados foram encontradas diretrizes para a escolha do modelo de sincronismo mais adequado.

Este trabalho parte, portanto, da ausência dessas diretrizes e propõe a existência dos modelos híbridos, em função das características dos ambientes transacionais que geram cada porção de informações do data warehouse e das formas de utilização destas porções pelas aplicações analíticas. Assim, o trabalho sugere a opção mais adequada para um determinado conjunto de informações, baseada nos diferentes níveis de sincronismo entre os ambientes transacional e analítico. A escolha do modelo de sincronismo é baseada em uma função de análise e alguns parâmetros, definidos neste trabalho, que avaliam as características tanto do ambiente transacional quanto do analítico relacionadas a cada porção do data warehouse.

A estrutura do trabalho é apresentada a seguir, sendo que, no capítulo 2 estão os conceitos básicos sobre o data warehouse e a modelagem multidimensional, típica do ambiente analítico. Aspectos detalhados desta modelagem e dos componentes do data warehouse,

[5]A expressão self maintainable views, que se refere a visões que possuem a capacidade de auto manutenção, será utilizada em sua forma original, em inglês.

bem como o modelo de sincronismo estático, estão descritos no capítulo 3. A abordagem dinâmica utilizando visões materializadas está descrita no capítulo 4 que também apresenta os aspectos e os problemas mais importantes que envolvem o sincronismo dinâmico entre o data warehouse e as aplicações transacionais.

No capítulo 5, as seções 5.2 e 5.3 avaliam as características dos ambientes transacional e analítico respectivamente, levando aos parâmetros propostos para a avaliação do modelo de sincronismo. Na seção 5.5 é descrita uma função de análise que é aplicada sobre cada conjunto de informações e no capítulo 6 tem-se o protótipo de tal função, juntamente com um estudo de caso, para validar a função e os parâmetros definidos neste trabalho.

Assim, o trabalho aqui desenvolvido apresenta grande relevância para a definição e implementação de um data warehouse, fornecendo diretrizes para a escolha da opção de sincronismo mais adequada a um determinado modelo de negócios.

Capítulo 2

Conceitos

2.1 O que é o Data Warehouse

De acordo com [AV98], o data warehouse existe para responder as questões que as pessoas têm sobre os negócios. Esta função contrasta fortemente com o propósito dos sistemas transacionais que as empresas utilizam e requer que o desenho ou o modelo de dados do data warehouse siga princípios completamente diferentes. As técnicas de modelagem dimensional, se aplicadas corretamente, garantem que o desenho do data warehouse reflita a forma de pensar dos gerentes de negócio e possa ser utilizado para responder suas questões.

Em todas as empresas, o processo de geração de negócios é composto por uma série de eventos que caracterizam suas principais atividades. A natureza e freqüência destes eventos variam conforme o tipo de negócio em que a empresa está envolvida: um produto é manufaturado, uma conta é creditada enquanto outra é debitada, um assento é reservado, um pedido é incluído etc. O controle e processamento corretos destes eventos são críticos para a empresa, sendo que estas atividades contribuem para seu sucesso ou fracasso. Para isso, a maioria das organizações possui um conjunto de sistemas conhecidos como *sistemas transacionais* ou *sistemas OLTP* (OnLine Transaction Processing) que capturam os eventos de forma individual e todos os detalhes associados a eles. Cada um destes

sistemas está encarregado de um tipo diferente de atividades e trata das transações de negócios segundo um conjunto de regras que garanta sua consistência e o armazenamento de todos os detalhes associados. Para um sistema OLTP, os princípios de desenho, como normalização e consistência de transações, são extremamente críticos. Porém, por tratar as transações de forma individual, os sistemas OLTP falham no que se refere a questões sobre o processo do negócio, como "quais foram os produtos mais vendidos durante o mês passado?"ou "quais produtos estão perdendo *market share*?"ou ainda "quais são nossos clientes mais fiéis?".

O data warehouse é desenhado para suprir estas falhas. Ele é construído para responder questões que não estão limitadas às transações individuais, porém tratam do processo de forma completa. Para isso, o desenho do data warehouse deve refletir a forma com que os especialistas enxergam o negócio e este é o ponto chave que distingue um data warehouse de um sistema OLTP. A técnica utilizada para se obter um modelo para o data warehouse que identifique e represente a informação importante para o modelo de negócios é a *modelagem dimensional.* Quando bem definido, o modelo dimensional pode ser uma ajuda de valor incalculável para as áreas de negócio, apoiando e otimizando todo o processo de tomada de decisões.

Conforme [Kel94], um data warehouse corporativo pode ser definido em termos de seis características básicas que o diferenciam dos outros sistemas na empresa. Os dados em um data warehouse são:

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

6. *Acessível* aos usuários que tenham um conhecimento limitado de sistemas computacionais ou estruturas de dados.

Além destas características, podemos citar sua *não-volatilidade*, ou seja, uma vez que o dado foi carregado no data warehouse, não deve mais sofrer alterações.

2.2 O modelo dimensional e suas implementações [AV98]

A modelagem de um data warehouse normalmente utiliza uma abordagem diferente da modelagem entidade-relacionamento, que é a maneira convencional para desenhar uma base de dados relacional contemplando toda a teoria de normalização dos dados.

O modelo dimensional (também chamado de multidimensional), utilizado para definir um data warehouse, representa os indicadores importantes para uma área de negócios, que são chamados de fatos ou métricas e os parâmetros, chamados dimensões, através dos quais estas métricas são vistas. A figura 2.1 mostra um exemplo de modelo dimensional para um processo de pedidos. As métricas definidas estão no quadro central e as dimensões estão representadas nos quadros ao redor das métricas. As métricas são sumariadas ou detalhadas de acordo com o interesse da análise a ser feita sobre os dados. Este modelo é fácil de ser entendido por uma pessoa da área de negócios, já que "as coisas que eu avalio"estão na parte central do diagrama e "as formas de se olhar para elas"estão nos quadros em volta.

Fica fácil perceber que estes quadros facilmente se transformarão em tabelas, que podem ser utilizadas para armazenar toda a informação. Um modelo como este não muda muito ao ser implementado em um banco de dados relacional. Cada quadro com os atributos de uma dimensão se torna uma tabela, chamada de *tabela dimensão*, na base de dados e o quadro central se torna uma grande tabela, chamada *tabela fato*, que contém, por vezes, milhões ou bilhões de linhas.

Porém, os modelos dimensionais nem sempre são implementados em bases de dados relacionais. Existem no mercado os *bancos de dados multidimensionais*, ou *MDDBs*, que armazenam as informações em um formato diferente, freqüentemente chamados de *cubos*.

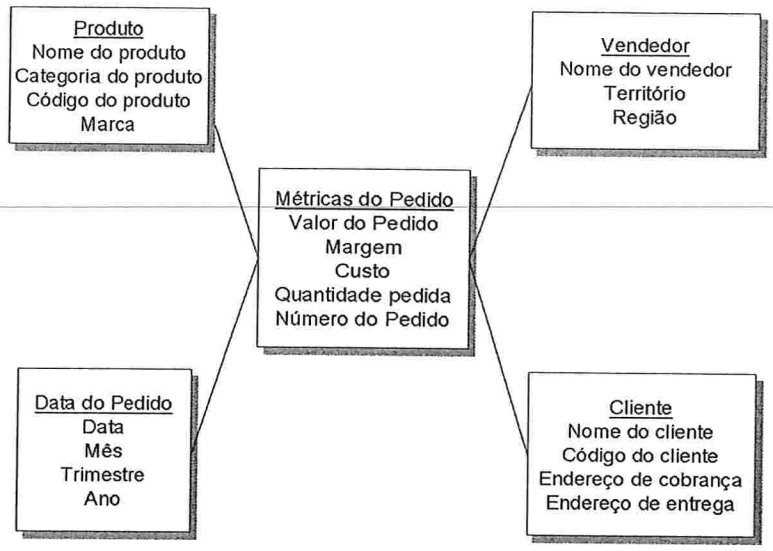


Figura 2.1: Modelo dimensional para um processo de pedidos

Os cubos são construídos de tal forma que, cada combinação de atributos das dimensões com uma métrica ou é precalculado ou é calculado muito rapidamente. Entretanto, a natureza de uma base de dados multidimensional também significa que não é possível manipular volumes de dados extremamente grandes, já que uma transação de análise dos dados, com uma ferramenta OLAP e que envolva um grande volume de dados, vai consumir grande quantidade de memória ou simplesmente não se efetua. Além disso, o número de atributos dimensionais armazenados em um cubo pode impactar o tamanho e o desempenho do cubo.

Uma das alternativas para solucionar estes problemas pode ser a implementação do modelo dimensional em um banco de dados relacional e, após isto, utilizá-lo como fonte para os cubos. Esta abordagem é muito utilizada em empresas que querem executar análises em pequenos subconjuntos de um grande conjunto de dados armazenados em um data warehouse. Quando esta abordagem é implementada, o data warehouse fica armazenado no banco de dados relacional, enquanto que os cubos contêm partes ou segmentos do data warehouse, são chamados de *data marts*.

Outra alternativa é utilizar uma ferramenta de consulta acessando o data warehouse

no banco relacional, transformando o resultado da consulta em um cubo que permita a análise rápida dos dados. Estes cubos podem ser gerados no computador do usuário ou em um servidor de aplicações. Esta abordagem é interessante para os usuários, já que não requer a administração centralizada destes cubos. É importante notar, porém, que o usuário fica sujeito aos limites de capacidade de processamento de seu PC ou do servidor de aplicações.

[SMKK98] apresenta uma arquitetura para o Data Warehouse, representada na figura 2.2. Nesta arquitetura, os dados são provenientes dos sistemas operacionais. Estas fontes são conectadas a *wrappers* ou monitores que efetuam o processo de seleção, transformação e limpeza dos dados. Além disso, monitoram as alterações nas fontes de dados propagando-as a um componente integrador, que combina os dados selecionados a partir das diferentes fontes operacionais. Estes dados, já consistentes, são propagados para o warehouse.

O metadados contém informações relevantes sobre a criação, gerenciamento e uso do data warehouse e funciona como uma ponte entre os usuários do data warehouse e os dados nele contidos.

O data warehouse pode ser acessado através de um servidor OLAP, que tem como objetivo apresentar as informações multidimensionais para as ferramentas de acesso, análise, geradores de relatórios, planilhas e ferramentas de mineração de dados (*Data Mining tools*). Basicamente, o servidor OLAP interpreta as consultas dos usuários convertendo-as em instruções adequadas, muitas vezes complexas, para o acesso ao data warehouse. Atualmente existem dois tipos de solução para implementação para acesso ao repositório de data warehouse. Uma é a utilização de modelos relacionais associados a tecnologias de buscas multidimensionais em cubos pré-construídos (MOLAP). Outra é a utilização de gerenciadores relacionais incrementados com tecnologias de índices bitmap e recuperação de dados com listas invertidas(ROLAP).

Na figura 2.2, pode-se identificar as várias camadas que compõem tal arquitetura:

1. Ferramentas de acesso para os usuários;

2. Servidor OLAP(MOLAP-ROLAP);

3. Sistema de gerenciamento do data warehouse e ferramentas para selecionar, transformar, limpar, integrar e copiar os dados;

4. Dados dos sistemas operacionais.

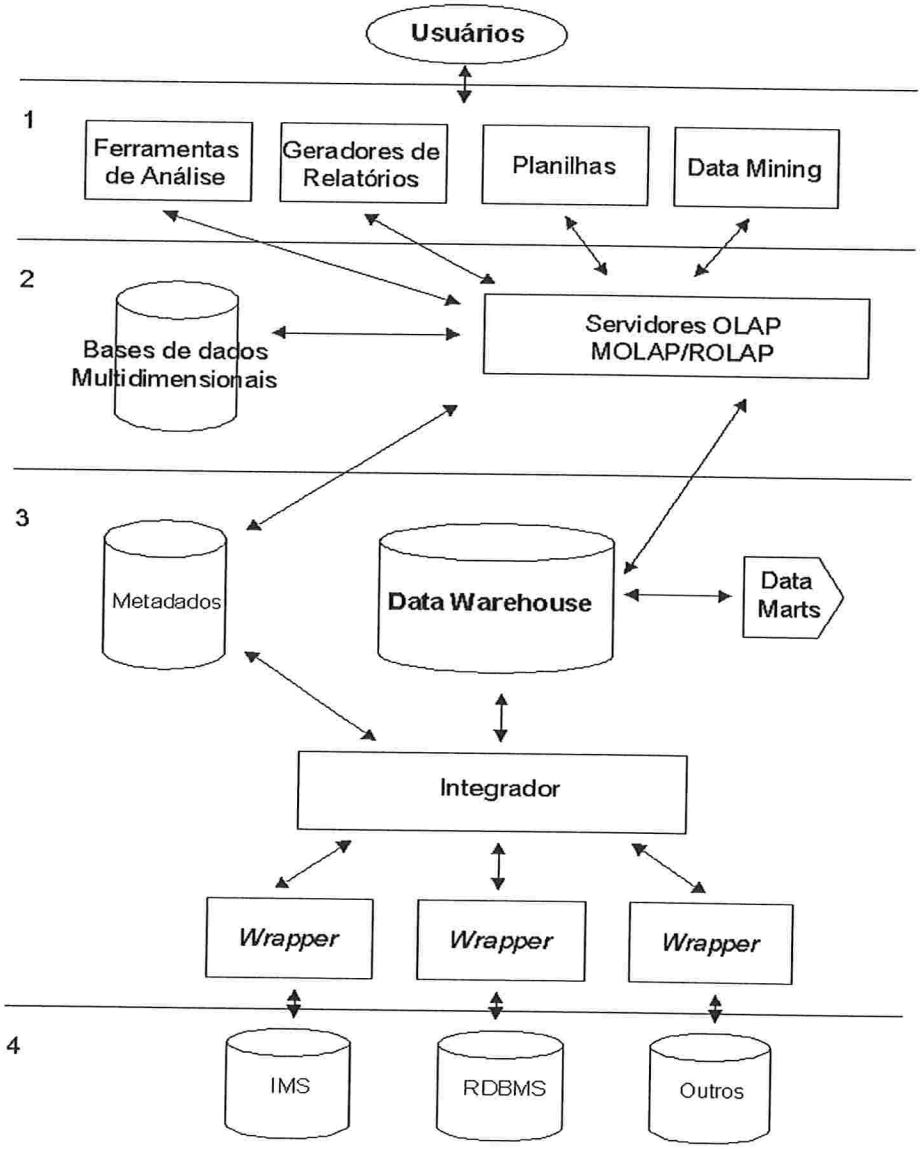


Figura 2.2: Arquitetura de data warehouse proposta por [SMKK98]

2.3 O modelo formal da base de dados multidimensional

[BPT97] define formalmente o modelo multidimensional implementado em uma base de dados relacional, apresentado nesta seção.

A estrutura básica deste modelo pode ser representada por um diagrama entidade-relacionamento como na figura 2.3.

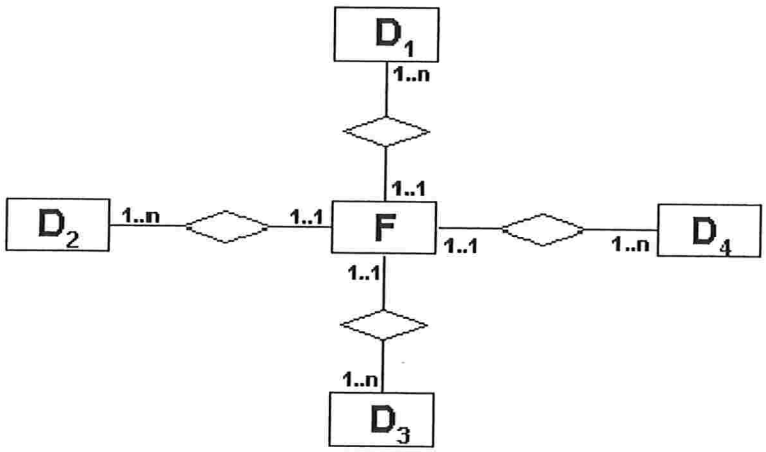


Figura 2.3: Representação Entidade-Relacionamento de uma base de dados multidimensional

Definição 2.3.1 *Uma* **base de dados multidimensional** *é uma coleção de relações* $D_1,\ldots, D_n$, F, *onde:*

- *Cada* D_i *é uma* **tabela dimensão**, *isto é, uma relação caracterizada por um identificador que identifica unicamente cada tupla (*d_i *é a chave primária de* D_i*).*
- F *é uma* **tabela fato**, *isto é, uma relação que conecta todas as tabelas* $D_1,\ldots, D_n$ *e o identificador de* F *é composto pelas chaves estrangeiras* $d_i,\ldots, d_n$ *de todas as tabelas dimensão conectadas. O esquema de* F *contém um conjunto de atributos adicionais* V *(que representam os valores sobre os quais serão aplicadas as funções de agregação).*

As tabelas dimensão podem conter hierarquias.

Definição 2.3.2 *Seja D uma tabela dimensão com o identificador d. Uma* **hierarquia de atributos** *em D é um conjunto de dependências funcionais* $FD_D = \{fd_0, fd_1, ..., fd_n\}$, *onde cada* fd_i *é caracterizado por dois conjuntos de atributos* $A_i^l \subset Attr(D)$ *e* $A_i^r \subset Attr(D)$ *(chamados respectivamente de lado esquerdo e lado direito da dependência); a dependência é representada por* $fd_i : A_i^l \rightarrow A_i^r$.

Cada dependência funcional fd_i é uma restrição ao conteúdo da tabela dimensão D, sendo que para cada par de tuplas t_1 e $t_2 \in D$, $t_1[A_i^l] = t_2[A_i^l] \Longrightarrow t_1[A_i^l] = t_2[A_i^r]$. Uma dependência fd_0 com $A_0^l = \{d\}$ e $A_0^r = \{Attr(D)$ - $d\}$ estará sempre presente em FD_D. As dependências funcionais devem ser acíclicas, isto é, o grafo obtido pelo desenho de um arco partindo de a_x e chegando em a_y deve ser acíclico, se $\exists\, fd_i \in FD_D \mid a_x \in A_i^l \wedge a_y \in A_i^r$.

Exemplo: Um exemplo prático, retirado de [Kim96] representa a base de dados multidimensional para uma grande cadeia de lojas de varejo. Esta cadeia possui um grande número de lojas, sendo cada uma delas um supermercado que vende uma ampla variedade de diferentes produtos. A base de dados multidimensional armazena informação sobre cada venda, por loja, por dia e considera também as promoções em cada produto vendido.

Pode-se identificar as seguintes dimensões: Product[6], que caracteriza cada produto vendido, Store, que caracteriza cada ponto de venda, Time, para o tempo e Promotion, que descreve as características das promoções dos produtos. A tabela fato fornece as informações sobre as vendas, sobre as quais serão realizadas as análises financeiras. Inclui identificadores para todas as dimensões e vários atributos descrevendo as vendas (como valor da venda, quantidade vendida, etc.)

Ao se considerar a dimensão Store, do exemplo acima, com chave s e restrita a um conjunto de atributos {z, c, s_t, n}, que representam respectivamente zip code, county, state e number of sale clerks, percebe-se que a dimensão tem a seguinte hierarquia de atributos:

$$\{fd_0 : s \rightarrow \{z, c, s_t, n\}, fd_1 : z \rightarrow c, fd_2 : c \rightarrow s_t\}$$

[6]Por razões didáticas, os nomes de tabelas e atributos serão mantidos no idioma original, inglês.

Definição 2.3.3 *Uma* **hierarquia de atributos da base de dados multidimensional** FD_{DB} *é a união das hierarquias de atributos* DF_{D_j} *de todas as dimensões* D_j *existentes na base de dados multidimensional.*

Capítulo 3

A abordagem estática do data warehouse atual

3.1 Características

Conforme já comentado, o modelo dimensional é poderoso, pois reflete a maneira de pensar dos especialistas de negócios e responde às suas necessidades de informações. A tecnologia relacional de bancos de dados possibilita ao data warehouse ser utilizado para responder as questões de forma rápida e precisa. Para isso, são necessários três componentes essenciais, a saber:

1. Os dados provenientes das várias fontes distribuídas pela empresa e armazenados em um único local;
2. Ferramentas que possibilitem a análise das informações armazenadas de forma rápida, flexível com alta qualidade de apresentação e
3. O conhecimento do especialista de negócios.

Existem inúmeras ferramentas, como as citadas no item 2 acima, disponíveis no mercado e são chamadas de OLAP (Online Analytical Processing). Estas ferramentas permitem ao usuário visualizar os vários níveis de detalhamento da informação, sob as

visões das diferentes dimensões definidas no modelo e têm sido alvo de vários trabalhos acadêmicos. O conhecimento do especialista de negócios é outro componente essencial, já que apenas ele pode tirar as conclusões adequadas das informações apresentadas, com o objetivo de tomada de decisões da corporação.

Em linhas gerais, o processo de implementação de um data warehouse está dividido nas seguintes etapas:

1. Levantamento do processo de negócio a modelar;
2. Definição dos modelos conceitual, lógico e físico;
3. Definição do processo de carga.

Dentre todas as etapas citadas acima, a que envolve o processo de carga é de longe a mais complexa. Além de trazer os dados de vários sistemas transacionais diferentes, o processo engloba atividades de verificação, padronização, limpeza e transformação dos dados antes da carga. A definição da periodicidade da carga depende da natureza do negócio que compõe o data warehouse e do tipo de informação armazenada. Algumas áreas de negócios requerem carga diária, enquanto que outras necessitam apenas de cargas mensais. Seja qual for a periodicidade escolhida, fica claro que o data warehouse não está sincronizado em tempo real com os sistemas transacionais. Para algumas aplicações, esta característica estática do data warehouse não compromete o resultado das análises porém, para outras, um data warehouse dinâmico, ou seja, sincronizado com os sistemas transacionais, é essencial. Para implementar um data warehouse dinâmico é necessário definir uma arquitetura que permita propagar as atualizações nas bases transacionais no instante em que elas ocorrem.

Atualmente, pode-se dizer que os data warehouses implantados são de natureza estática, tendo processos de carga de alta complexidade e que requerem um grande tempo de processamento.

3.2 Conceitos da modelagem [Tan97]

A modelagem dimensional combina tabelas fato que armazenam dados históricos temporais (normalmente numéricos), indexadas por chaves dimensionais que estão descritas nas tabelas dimensão correspondentes. As tabelas dimensão contêm informações como, por exemplo: períodos de tempo, produtos, mercados, organizações, contas, vendedores e clientes e inclui as descrições e os atributos destas dimensões. Além disso, as tabelas dimensão contemplam toda a estrutura da dimensão, como os agrupamentos dos produtos em marcas e em categorias, as cidades em estados, em regiões e em países e assim por diante.

As tabelas fato contêm as métricas ou os fatos a serem analisados dentro do modelo de negócios. Cada um destes fatos está diretamente relacionado às dimensões, que descrevem suas condições de ocorrência. Todo o relacionamento entre a tabela fato e as tabelas dimensão é feito através de chaves.

Uma consulta executada neste modelo dimensional, geralmente se inicia por uma pesquisa nas tabelas dimensão, aplicando-se os filtros de valores e obtendo-se como resultado um conjunto de chaves. Após isso, o acesso é feito à tabela fato, garantindo assim a precisão no acesso aos dados através de uma estrutura completa de chaves, eliminando-se um *table scan* e, com isso, obtendo-se o melhor desempenho possível de uma tecnologia relacional.

O conceito de armazenamento das dimensões separadamente garante que a base de dados trate os vetores esparsos de maneira eficaz, isto é, sem armazenar vazios, assegurando o mais eficiente acesso possível.

O modelo dimensional pode ser implementado utilizando vários tipos de esquemas diferentes. Provavelmente, o *Star schema*, apresentado por R. Kimball em [Kim96], seja o primeiro esquema utilizado para representar o data warehouse implementado em um banco de dados relacional. Apesar de ser o mais conhecido, o *Star schema* não é o único. De fato, existe uma série de variações que também são analisadas neste trabalho. Porém, antes de se iniciar a modelagem em um esquema particular, é necessário definir o

processo de negócio a ser modelado. A figura 3.1 mostra um exemplo de esquema para um processo de pedidos. A tabela central é a tabela fato e as tabelas em torno da fato são as dimensões. Neste exemplo, estão representadas as dimensões Produto, Tempo (data), Cliente e Vendedor (incluindo informações de ordem geográfica). Os fatos representados são: valor do pedido, margem, custo, quantidade pedida e número do pedido.

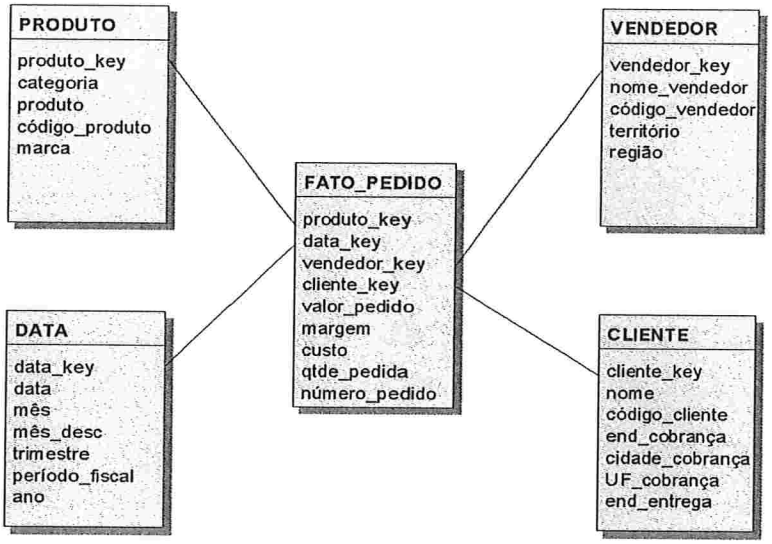


Figura 3.1: Esquema para um processo de pedidos

Após definido o escopo do data warehouse, os próximos passos são: definir a granularidade das informações representadas na tabela fato, identificar as dimensões e enumerar as métricas ou fatos. Na próxima seção é elaborada uma descrição sucinta sobre cada uma destas etapas e, na seqüência, os tipos de esquemas básicos e suas variações.

3.2.1 A granularidade das informações [AV98]

Como já citado, uma das primeiras decisões feita para modelar o data warehouse está relacionada com o nível de detalhe das métricas a serem armazenadas. Este nível de detalhamento é conhecido como granularidade da tabela fato. É muito importante que todas as linhas na tabela fato sejam armazenadas com informações exatamente no mesmo nível de detalhes. Como exemplo, um processo de pedidos teria sua granularidade definida

no nível de detalhe da linha individual do pedido. Se fossem armazenados diferentes níveis de detalhamento na tabela fato, a utilização da base de informações ficaria bastante prejudicada. Para ilustrar esta situação, se supõe que a maioria das informações esteja armazenada no nível da linha de detalhe do pedido e que, entretanto, algumas informações foram armazenadas no nível do cabeçalho do pedido. No nível de detalhe da linha do pedido existe um relacionamento com cliente, produto, representante comercial e tempo. Porém, no nível do cabeçalho do pedido, o relacionamento com os produtos específicos que foram pedidos desaparece. As informações relevantes que estejam em diferentes níveis de granularidade deveriam ser armazenadas em tabelas fato diferentes.

Em geral, pode-se dizer que, a aderência de uma tabela fato a uma certa granularidade requer que as chaves que relacionam as linhas da tabela fato às linhas das dimensões nunca sejam nulas. Um relacionamento opcional a uma dimensão geralmente indica um problema de granularidade.

3.2.2 As dimensões

Uma vez que o nível de granularidade tenha sido definido, a escolha das dimensões a serem utilizadas deve ser o próximo passo. O ideal seria definir uma dimensão com um grande número de atributos, representando um rico conjunto de detalhes sobres o processo de negócio. É comum que as dimensões apresentem 100 ou até mesmo 200 atributos em uma única tabela dimensão [AV98].

As tabelas dimensão contêm informações sobre as dimensões dos dados, ou seja, tempo, produto, mercado, contas e assim por diante e devem ser desenhadas a partir da perspectiva do usuário. Por esta razão, os atributos e suas descrições devem ser definidos de forma significativa para o usuário e adequados para a posterior exibição em relatórios. Uma das principais funções da tabela dimensão é reunir os atributos que serão utilizados para qualificar as consultas e cujos valores serão utilizados para agrupar e sumariar as métricas [AV98].

Cada tabela dimensão deve ter múltiplos atributos que contenham valores ou textos

que possam ajudar a descrever a chave. Estas colunas de atributos são utilizadas para filtrar o conteúdo da dimensão. Além disso, a utilização de atributos do tipo inteiro, quando apropriados, favorecem as operações de filtragem como *maior que*, *menor que* e *entre*, de acordo com [Tan97].

Os atributos de uma dimensão podem compor uma hierarquia ou ser apenas descritivos. Por exemplo, em uma dimensão produto, a hierarquia pode ser composta pelos atributos item, marca, tipo e divisão. A hierarquia definida na dimensão é requisito básico para as funções de agregação das métricas contidas na tabela fato. Através dos agrupamentos dos elementos na hierarquia, os usuários podem analisar os fatos em um nível maior ou menor de detalhes, conforme sua necessidade específica. Este conceito de hierarquia, que permite a implementação de agregações das métricas, é muito importante também para a dimensão tempo e, vale ressaltar que, conforme [Rad96], é muito raro um modelo dimensional que não inclua a dimensão tempo como uma dimensão fundamental.

Deve-se resistir ao impulso de aplicar as regras de normalização nas tabelas dimensão [AV98]. O processo de normalização, separando os atributos em várias tabelas diferentes faz com que as consultas fiquem bem mais complexas, o banco de dados leve mais tempo para recuperar os dados e, por conseqüência, os usuários esperem mais tempo pelas respostas. Este custo é muito alto como resultado da economia de apenas uns poucos bytes em uma tabela dimensão que, em comparação com uma tabela fato, é minúscula.

Manter as tabelas dimensão desnormalizadas faz com que as hierarquias naturais contidas nos dados fiquem bem definidas. No exemplo de pedidos, pode-se notar como a dimensão tempo (data) inclui os atributos ano, trimestre, mês e data (que inclui o dia) juntamente com outros atributos relacionados a uma data específica. Estes componentes não foram colocados em uma série de tabelas progressivamente mais detalhadas. Em vez disso estão em uma única tabela que deixa clara a hierarquia. Enquanto que o ano de 1998 pode aparecer na tabela 365 vezes, isto ficará invisível para o usuário. Quando as métricas forem sumariadas por ano, apenas uma ocorrência de 1998 aparecerá no relatório.

Um atributo muito importante da tabela dimensão é sua chave. A chave primária de uma tabela dimensão dever ser sempre um atributo único e definido pelo sistema com

um valor genérico. Por questões de desempenho, não se utilizam chaves compostas por várias partes nem tampouco chaves concatenadas. Também não são utilizadas chaves ou identificadores provenientes de outros sistemas como, por exemplo, código do cliente ou código do produto. Existem várias razões para se utilizar chaves genéricas em vez de chaves com valores significativos. Em caso de alterações de atributos de um cliente como, por exemplo, seu endereço, teremos que dar um tratamento especial e inserir este mesmo cliente, com seu novo endereço, com outra chave. Se estivermos utilizando o código do cliente como chave primária, isto não será possível. Os tratamentos de alterações são analisados neste trabalho, em seções posteriores.

3.2.3 Os fatos

Após identificar a granularidade e as dimensões, é o momento de focalizar a tabela fato. Para isso, inicia-se definindo quais as métricas ou fatos que serão avaliadas no data warehouse. Estes fatos são os números que serão analisados através das diferentes dimensões [AV98]. De acordo com [Rad96], a seleção dos fatos para compor o modelo do data warehouse é relativamente simples: uma vez que a área de negócios esteja definida, a lista de fatos a serem utilizados responde a questão: *O que estamos avaliando?*.

Pode-se ver, no exemplo representado nas figuras 2.1 e 3.1, em suas respectivas tabelas centrais, quais são os valores relevantes para a análise do processo de pedidos.

Existem três tipos de métricas ou fatos que são discutidos na próxima seção. Os outros elementos de uma tabela fato, são analisados na seção seguinte.

Os três tipos de métricas

As métricas mais comuns são as *completamente aditivas*. Uma métrica é completamente aditiva, quando faz sentido sumariá-la, adicionando seus valores ao longo de qualquer dimensão. No exemplo de pedidos apresentado, o valor do pedido, a margem, o custo e a quantidade pedida são todas métricas completamente aditivas. Apesar de serem armazenadas na tabela fato para um determinado produto, um cliente, um vendedor e

uma data específica, pode-se facilmente agregá-las da maneira que interesse. Para verificar os pedidos de um determinado mês, basta adicionar os valores dos pedidos de todas as datas daquele mês. O mesmo ocorre se for necessário verificar a margem obtida para uma determinada categoria de produtos. Basta adicionar os valores de margem para todos os produtos de uma determinada categoria.

As métricas completamente aditivas são bastante úteis e poderosas, já que não existem limitações em sua utilização. Podem ser facilmente agregadas para qualquer combinação de valores das dimensões.

Em contraste total com este tipo de métrica, temos as métricas *não aditivas*. As métricas não aditivas não podem ser adicionadas ao longo dos valores das dimensões. Considerando a métrica margem, expressa como uma porcentagem de vendas e designada *margem percentual*, pode-se dizer que esta métrica representa a margem como percentual e não mais como valor expresso na moeda corrente. Seria muito simples adicionar esta métrica à tabela fato, porém esta métrica teria pouca utilização, já que não é possível sumariá-la de acordo com a dimensão que interesse. Por exemplo, em uma determinada data, um vendedor vende a um cliente 4 tipos diferentes de produtos, cada um deles com uma margem percentual de 25%. Não faz sentido adicionar os quatro valores de margem percentual para calcular a margem total para este pedido.

Aparentemente conclui-se que este tipo de métrica não pode ser utilizado em nossa tabela fato. Na verdade, esta métrica é derivada de outras duas métricas que são aditivas: margem e valor do pedido. A solução é, portanto, armazenar na tabela fato apenas seus componentes, que são completamente aditivos, sendo que o cálculo para expressar a margem percentual deverá ser feito pela aplicação, que neste caso é a divisão da margem pelo valor.

O terceiro tipo de métrica é a *semi-aditiva*. A métrica semi-aditiva pode ser sumariada ao longo de determinadas dimensões, porem não todas. Ao considerar como exemplo o gerenciamento de saldo das contas de um banco, o saldo é armazenado no final de cada dia, para cada cliente, por conta ao longo do tempo. Em alguns casos este saldo é aditivo. Se um cliente tem uma conta corrente e uma conta poupança, podemos adicionar os

saldos de cada conta no final de um dia e obter resultado significativo. É possível também adicionar os saldos de uma determinada agência para obter um panorama da situação geral de cada localidade. Entretanto, não faz o menor sentido adicionar o saldo de um cliente ao longo do tempo. Por esta razão, a métrica saldo é considerada semi-aditiva, já que é aditiva para algumas dimensões, porém não o é para outras..

Segundo [GMR98], este tipo de atributo não aditivo ou semi-aditivo pode ser agregado ou sumariado utilizando-se outros operadores como média, máximo ou mínimo. É o caso da métrica temperatura, que é considerada não aditiva, já que adicionar temperaturas dificilmente faz sentido.

Outros elementos da tabela fato

Além das métricas, cada tabela fato tem uma chave primária. Esta chave primária é composta por várias colunas, sendo que cada uma delas corresponde logicamente a uma chave na tabela dimensão. Cada elemento componente da chave deve, portanto, estar representado e descrito em uma tabela dimensão correspondente, que logicamente se une a uma ou mais tabelas fato através de colunas idênticas. Pode-se dizer, então, que a chave primária de uma tabela fato é uma chave composta por um subconjunto de chaves estrangeiras para as tabelas dimensão. A dimensão tempo é sempre representada como parte da chave primária.

A figura 3.2 ilustra uma tabela fato que contém o valor das vendas e a quantidade de unidades vendidas, logicamente unida a uma tabela dimensão de mercado correspondente. Na figura não estão representadas as outras dimensões produto e tempo.

Ocasionalmente, não é necessário armazenar nenhuma métrica em uma tabela fato. Conforme [AV98], às vezes, o simples armazenamento de um relacionamento entre as dimensões em um certo ponto do tempo é tudo que uma área de negócios necessita como métrica. Este tipo de tabela fato é a tabela sem fato, ou seja, do original *factless fact table*. O exemplo mais típico deste tipo de tabela fato é o controle de freqüência de alunos em determinadas disciplinas. Não existe uma métrica associada e apenas a existência do relacionamento entre as dimensões já é um indicativo suficiente para o processo de análise.

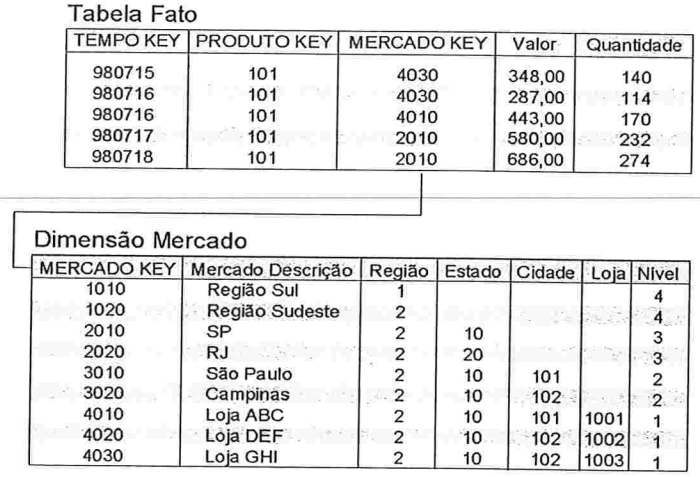


Tabela Fato

TEMPO KEY	PRODUTO KEY	MERCADO KEY	Valor	Quantidade
980715	101	4030	348,00	140
980716	101	4030	287,00	114
980716	101	4010	443,00	170
980717	101	2010	580,00	232
980718	101	2010	686,00	274

Dimensão Mercado

MERCADO KEY	Mercado Descrição	Região	Estado	Cidade	Loja	Nível
1010	Região Sul	1				4
1020	Região Sudeste	2				4
2010	SP	2	10			3
2020	RJ	2	20			3
3010	São Paulo	2	10	101		2
3020	Campinas	2	10	102		2
4010	Loja ABC	2	10	101	1001	1
4020	Loja DEF	2	10	102	1002	1
4030	Loja GHI	2	10	102	1003	1

Figura 3.2: Junção lógica entre tabelas fato e dimensão

Este tipo de tabela encontra-se ilustrado na figura 3.3.

Pode ocorrer que a mesma tabela fato tenha múltiplas chaves estrangeiras, como parte de sua chave primária, que se relacionam com a mesma dimensão. Por exemplo, pode-se ter o mesmo fato com múltiplas datas associadas a ele, como no caso da tabela fato de remessa de produtos, onde duas chaves estrangeiras, data da remessa e data do pedido, estão associadas a uma mesma tabela dimensão tempo. A figura 3.4 ilustra esta situação.

Neste caso, não é necessária a criação de duas tabelas dimensão data. Em vez disso, utiliza-se visões ou sinônimos para referenciar duas cópias virtuais da mesma tabela dimensão em uma única instrução SQL, no momento da consulta.

3.3 Do MER para o modelo do Data Warehouse

A criação de um sistema de informações, que seja bem documentado e que ao mesmo tempo responda aos requerimentos especificados pelos usuários, deve ter como base um cuidadoso desenho conceitual. Apesar disso, a maior parte da literatura sobre data warehouse trata da modelagem lógica e física, sem ao menos mencionar os aspectos relacionados à sua modelagem conceitual.

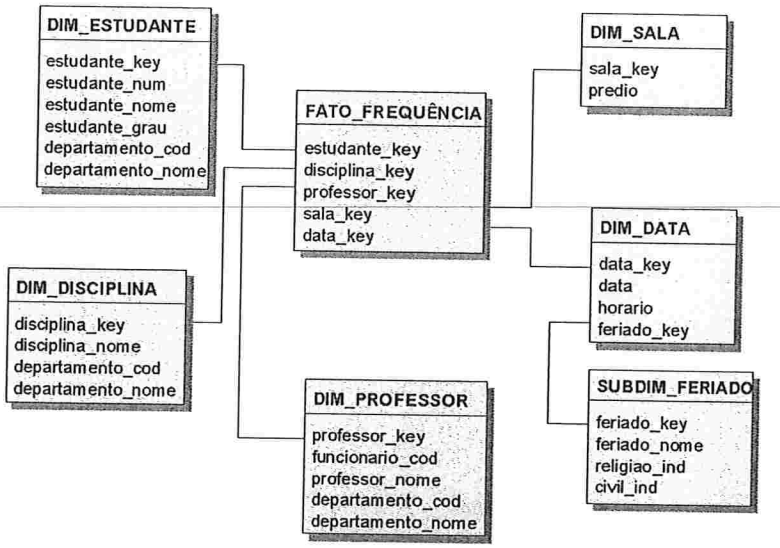


Figura 3.3: Um exemplo de tabela fato do tipo *factless*

A falta de interesse em assuntos relacionados ao desenho conceitual deve-se, principalmente, a dois fatores, segundo [GMR98]:

1. O uso do data warehouse começou no mundo industrial como resultado das solicitações de usuários que, tipicamente, não davam importância aos aspectos conceituais;

2. O desenho lógico e físico afeta mais a otimização do desempenho dos sistemas, que é o objetivo principal nas aplicações de data warehouse, que o desenho conceitual.

Portanto, é comum encontrar aspectos do desenho conceitual inseridos no desenho lógico do data warehouse.

O modelo Entidade Relacionamento (E/R) é amplamente utilizado como um formalismo conceitual que fornece documentação padronizada para os sistemas relacionais de informação e, por esta razão, muitos esforços têm sido feitos com o objetivo de utilizá-los como entrada para o desenho de bases de dados não relacionais.

[SBHD99] apresenta uma extensão ao modelo Entidade Relacionamento, argumentando que este tipo de modelo básico não é apropriado para a modelagem multidimensional conceitual já que a semântica das caraterísticas principais desta modelagem não

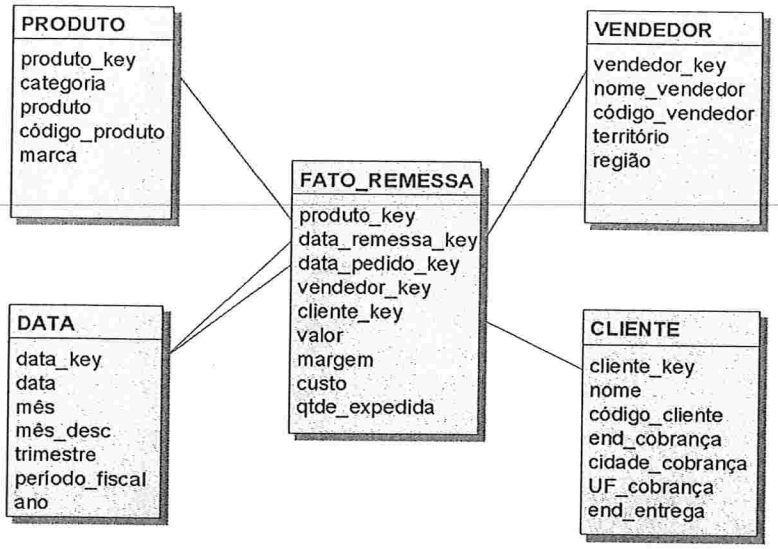


Figura 3.4: Relacionamento múltiplo entre uma tabela fato e uma dimensão

pode ser adequadamente representada. Em [SBHD99] é apresentada uma especialização do modelo E/R, chamado Modelo Entidade Relacionamento Multidimensional (ME/R), sendo que, para representar a estrutura multidimensional, são definidos conjuntos especializados de relacionamentos e entidades, estendendo assim o modelo ER básico.

[GMR98] também propõe um modelo conceitual, chamado *dimensional fact schema* (DF). Eles utilizam uma notação gráfica e uma metodologia que resulta em um modelo DF, partindo-se de um modelo ER dos dados que será a fonte para a definição do data warehouse. As técnicas utilizadas, apesar de apoiarem conceitos semanticamente ricos, não estão baseadas em um modelo de dados formal (o modelo ER é utilizado apenas como entrada para o processo). Sua argumentação, para a não utilização do modelo ER, baseia-se, inicialmente, na seguinte citação de [Kim96]:

"Os modelos de dados Entidade Relacionamento... não podem ser entendidos pelos usuários e não podem ser navegados de forma proveitosa pelo software de banco de dados. Os modelos Entidade Relacionamento não podem ser utilizados como base para os data warehouses das empresas".

Na próxima seção são apresentados o modelo conceitual gráfico para data warehouse,

o *dimensional fact schema* e uma metodologia semi-automática para construir o modelo DF a partir de um modelo E/R já existente e que descreva um sistema de informações operacional, ambos baseados no trabalho desenvolvido por [GMR98]. Em alguns casos, a documentaçao do E/R não está disponível, porém, a metodologia apresentada pode ser aplicada a partir da base de dados relacional existente, desde que as multiplicidades (-para-um ou -para-muitos) das associações lógicas estabelecidas pelas restrições das chaves estrangeiras sejam conhecidas.

3.3.1 O modelo conceitual para data warehouses

Dentro do escopo desta abordagem, o modelo conceitual para um data warehouse é composto por um conjunto de esquemas de fatos estruturados em árvores. Estes esquemas de fatos têm como componentes os fatos, as dimensões e as hierarquias. Como já discutido anteriormente, um fato é um objeto de interesse da empresa, que será o foco das análises; uma dimensão determina a granularidade adotada para representar os fatos e a hierarquia determina como as instâncias dos fatos podem ser agregadas e selecionadas para apoiar o processo de tomada de decisão.

A estrutura em árvore utilizada para representar estes esquemas tem como raiz um fato. Este fato é representado com uma caixa que indica seu nome e tipicamente, um ou mais atributos numéricos que "medem"o fato a partir de diferentes pontos de vista. A figura 3.5 representa um esquema de fatos para o fato SALE em uma cadeia de lojas. Os atributos fato quantity sold e returns contêm os valores que serão utilizados para a análise.

Conforme representado na figura 3.5, cada vértice diretamente conectado ao fato é uma dimensão. As sub-árvores, que têm como raiz uma dimensão, são as hierarquias. Os atributos, representados por círculos nos vértices, podem assumir um conjunto discreto de valores, sendo que, as linhas que conectam um par de atributos representam um relacionamento do tipo -para-um. Na raiz de cada hierarquia está representado o nível mais detalhado de granularidade da agregação, sendo que, os vértices ao longo da hierarquia definem progressivamente níveis menos detalhados.

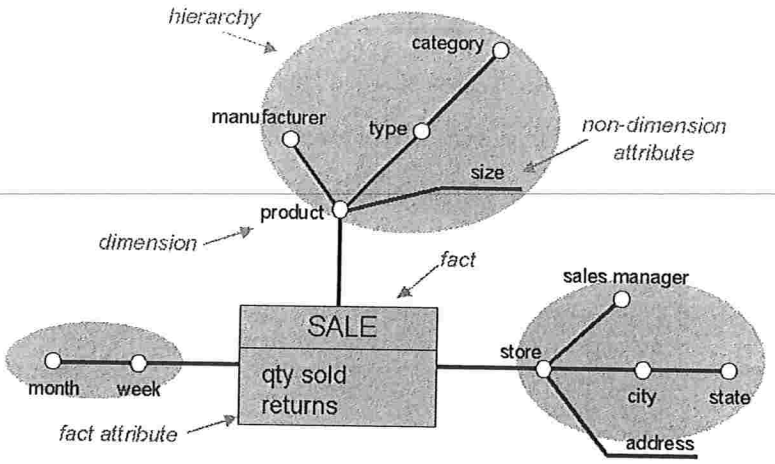


Figura 3.5: Um exemplo de esquema de fato com três dimensões para uma cadeia de lojas

Pode-se ver, portanto, que o esquema da figura 3.5 tem três dimensões: week, product e store. Alguns vértices terminais do esquema de fatos podem ser representados por linhas em vez de círculos, como no caso de size e address na figura 3.5. Estes vértices correspondem a atributos não-dimensionais, ou seja, são atributos que contêm informações adicionais sobre um atributo da hierarquia e está conectado a este por um relacionamento do tipo -para-um. Diferentemente dos atributos da hierarquia, os atributos não-dimensionais não podem ser utilizados para agregação.

A opcionalidade nos relacionamentos pode ser representada neste esquema, bastando acrescentar um sinal "/"na linha que conecta um par de atributos.

Um fato expressa um relacionamento muitos-para-muitos entre as dimensões. Cada combinação de valores das dimensões define uma instância do fato caracterizada por exatamente um valor para cada atributo. Estas instâncias são as informações básicas representadas em um data warehouse. Na figura 3.5, uma instância do fato descreve a quantidade de um produto vendida durante uma semana em uma loja e seu correspondente valor de lucro total.

Um esquema de fatos pode não ter nenhum atributo fato. Neste caso, cada instância do fato armazena a ocorrência de um evento. Os esquemas de fatos sem atributos fato correspondem, no modelo lógico, a um tabela de fatos do tipo *factless*, tipicamente utilizada

para acompanhamento de eventos.

3.3.2 A representação da aditividade no modelo

As pesquisas em um data warehouse são tipicamente baseadas na extração de dados agregados, apresentados na forma de relatórios estruturados, analisados durante os processos de tomada de decisão. Apesar da possibilidade de apresentação dos dados em seu nível mais detalhado, as análises são freqüentemente realizadas em diferentes níveis de abstração e, para isso, as instâncias dos fatos devem ser agregadas. Estas agregações requerem a definição de um operador, para que seja possível compor os valores dos atributos, caracterizando cada nível de sumário. Em geral, utiliza-se o operador soma para agregar estes valores ao longo da hierarquia, já que a maioria dos atributos fato é aditiva. Um exemplo de atributo aditivo, no exemplo da figura 3.5, é a quantidade vendida (qty sold), ou seja, a quantidade vendida para um determinado gerente de vendas é a soma das quantidades vendidas em todas as lojas gerenciadas por este gerente.

Os conceitos de semi-aditividade e não-aditividade, já apresentados anteriormente também são representados no modelo, lembrando que estes tipos de atributos podem ser agregados utilizando outros operadores como média, valor máximo e mínimo.

Em um modelo DF, os atributos são aditivos em todas as dimensões por *default*. A semi-aditividade e não-aditividade são representadas explicitamente indicando em quais dimensões o atributo não pode ser somado. Se algum outro operador, que não a soma, for utilizado, deve ser indicado explicitamente no modelo. A figura 3.6 exemplifica este tipo de atributo e sua representação no modelo.

3.3.3 A representação dos modelos de pesquisa em um esquema de fatos

Os operadores básicos das ferramentas OLAP, para a formulação de pesquisas nos data warehouses são: *roll-up*, *drill down*, *drill across* e *slice-and-dice* utilizados, respectivamente, para agregar atributos fato com o objetivo de visualizar os dados em um nível

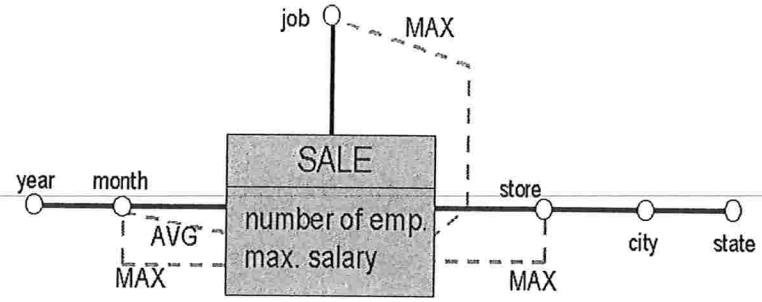


Figura 3.6: Representação da semi-aditividade no esquema de fatos

maior de abstração, desagregar atributos fato para aumentar o nível de detalhamento, relacionar e comparar fatos distintos, selecionar e projetar fatos para reduzir sua dimensionalidade.

Em um esquema de fatos, uma pesquisa pode ser representada por um *query pattern*, composto por um conjunto de marcadores (representados por círculos preenchidos) colocados nos atributos das dimensões. Estes marcadores podem ser colocados dentro de cada hierarquia, para indicar quais os níveis de agregação das instâncias de fatos. Uma dimensão que não possui marcadores indica que nenhum de seus atributos está envolvido na pesquisa. Os atributos não-dimensionais não precisam ser mostrados em um *query pattern*. A figura 3.7 mostra o *query pattern* representando a seguinte pesquisa: "a quantidade total vendida e a média de lucro por unidade vendida para cada semana e para cada tipo de produto". A média de lucro por unidade vendida é a razão entre o total de lucro e a quantidade vendida (returns / qty sold).

Caracterizando os processos de consulta, [MF01] descreve as operações típicas do ambiente OLAP:

- *Pivoteamento* - orienta a visão multidimensional dos dados: dadas as dimensões (a e b, por exemplo) temos que, cada ponto (x,y) corresponde ao valor agregado do par de dimensões escolhidas;

- *Rollup* - aumenta o nível de agregação: diminui o detalhe sobre uma ou mais hierarquias de dimensões;

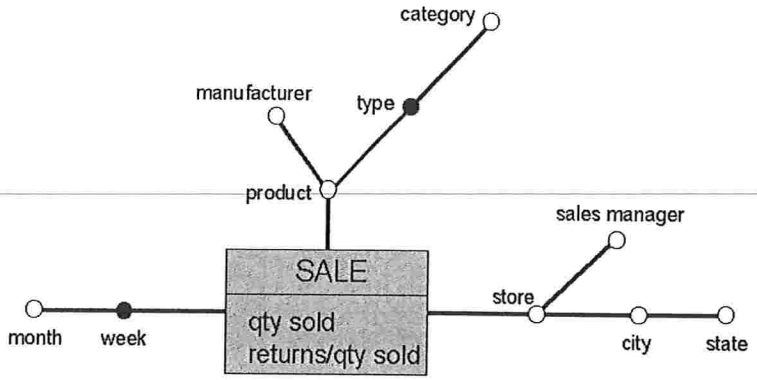


Figura 3.7: Representação de um modelo de pesquisa (*query pattern*)

- *Drill-down* - decresce o nível de agregação: aumenta o detalhe sobre uma ou mais hierarquias de dimensões;

- *Slice-and-dice* - seleção e projeção: corresponde a reduzir/trocar a dimensão, tomando como critério de seleção um determinado valor de uma dimensão e projetando tal resultado como critério de seleção de valores de outras dimensões. Um exemplo é apresentado na consulta 4.

Segue abaixo um exemplo, partindo-se do modelo representado pela figura 3.8, supondo a necessidade de se conhecer a quantidade de cada um dos produtos vendidos por mês, no sistema representado. Deve-se, então, pivotear as seguintes dimensões: Tabela de Fatos (Fatos), Produto e Data, sendo necessário, para isto, o seguinte comando SQL:

Consulta 1 - Pivoteamento:

```
Select Sum(F.Quantidade) as quantidade_vendida, P.nome, D.mes
From Fatos as F, Produto as P, Data as D
Where F.id_produto = P.id_produto
                    and F.id_data = D.id_data
Group by P.nome, D.mes;
```

Desta maneira é possível obter a resposta para o problema apresentado.

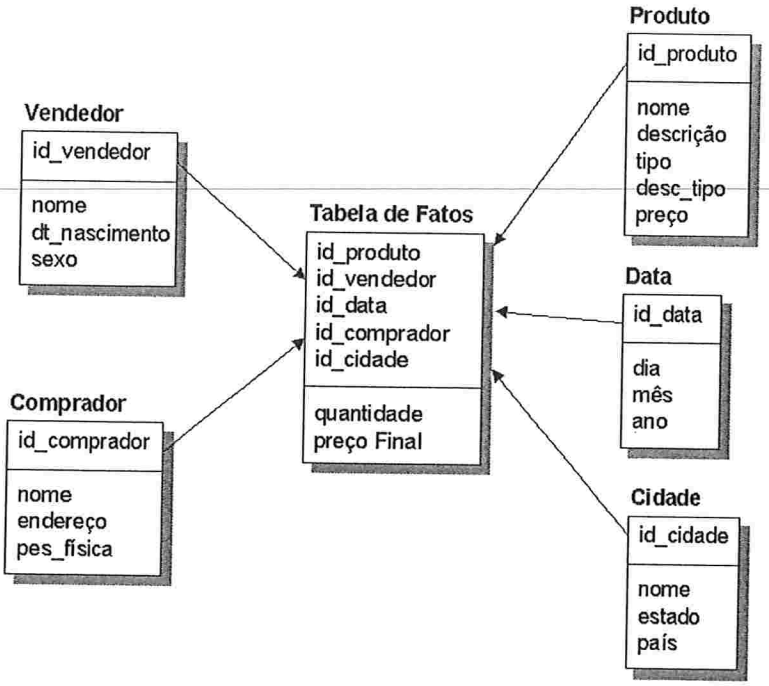


Figura 3.8: Modelo para exemplos de consultas OLAP

Outras operações relacionadas com o pivoteamento são *rollup* e *drill-down*. Se o *rollup* for executado sobre a consulta anterior significa que o nível de agregação aumenta, ou seja, a análise da venda de produtos passa a ser feita por ano (próximo nível de agregação em relação ao mês). *Drill-down* é o processo inverso de *rollup*, ou seja, o nível de agregação decresce, passando a analisar as vendas de produtos por dia e não mais por mês. É importante notar que, para as operações de *rollup* e *drill-down* é necessário que haja uma hierarquia. No exemplo apresentado, tais operações são permitidas para as dimensões de Produto, Data e Cidade. Os respectivos comandos SQL para *rollup* e *drill-down* seriam:

Consulta 2 - Rollup:

```
Select Sum(F.Quantidade) as quantidade_vendida, P.nome, D.ano
From Fatos as F, Produto as P, Data as D
Where F.id_produto = P.id_produto
                    and F.id_data = D.id_data
Group by P.nome, D.ano;
```

Consulta 3 - Drill-down:

```
Select Sum(F.Quantidade) as quantidade_vendida, P.nome, D.dia
From Fatos as F, Produto as P, Data as D
Where F.id_produto = P.id_produto
                    and F.id_data = D.id_data
Group by P.nome, D.dia;
```

Supondo agora que, após tal análise (consulta 1), seja necessário saber a quantidade vendida de determinado produto (tênis Bical, por exemplo), para cada cidade durante todo o período. Neste caso, a operação realizada é *slice-and-dice*, sendo necessárias a tabela de Fatos e as dimensões Região e Produto. O comando SQL correspondente seria:

Consulta 4 - Slice-and-dice:

```
Select Sum(F.Quantidade) as quantidade_vendida, C.nome
From Fatos as F, Cidade as C, Produto as P
Where F.id_produto = P.id_produto
                    and F.id_cidade = Cidade.id_cidade
                    and P.nome = 'Tênis Bical'
Group by C.nome;
```

A consulta 4 não se utiliza das dimensões Data e Vendedor, ao contrário da consulta 1. Porém, a consulta 4 se utiliza da dimensão Cidade, que não foi utilizada pela consulta 1, e ainda seleciona um determinado valor para a dimensão de Produto ('Tênis Bical').

3.3.4 Do E/R aos esquemas de fatos

Nas últimas décadas, a maioria dos sistemas de informação foi documentado utilizando os esquemas E/R (Entidade/Relacionamento), portanto, parece natural que o modelo conceitual de um data warehouse tenha como base um esquema E/R existente. A metodologia para geração do DF, partindo de um esquema E/R, contempla os seguintes passos:

1. Definição dos fatos;

2. Para cada fato:

(a) Criação da árvore de atributos;

(b) Adequação da árvore de atributos ("podar"e "enxertar"a árvore);

(c) Definição das dimensões;

(d) Definição dos atributos fato;

(e) Definição das hierarquias.

Cada etapa apresentada acima é detalhada nas próximas seções, utilizando como exemplo um esquema E/R simplificado conforme mostrado na figura 3.9. Cada instância do relacionamento Sale representa um item que se referencia a um único produto em um ticket de venda. O atributo unitPrice é colocado em Sale em vez de em Product, já que o preço dos produtos pode variar com o tempo.

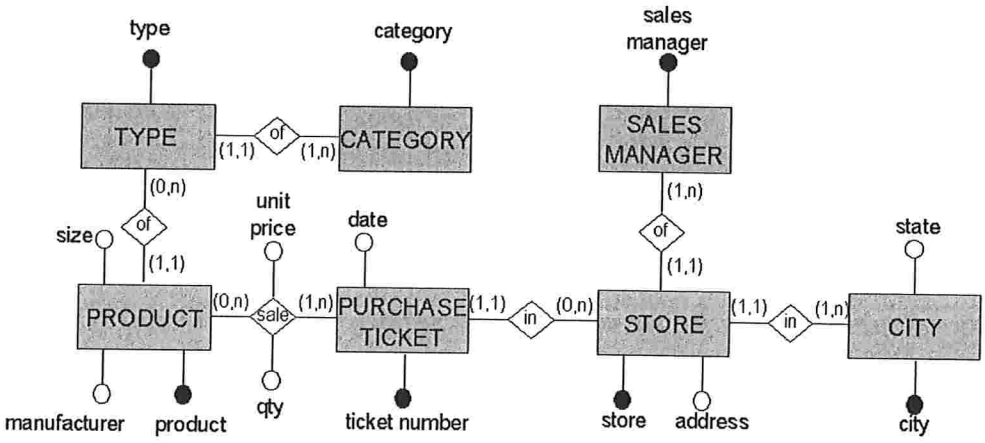


Figura 3.9: Esquema E/R simplificado base para a geração do modelo DF Sale

Definição dos Fatos

Os fatos são os conceitos de maior interesse no processo de tomada de decisões. Tipicamente, eles correspondem aos eventos que ocorrem dinamicamente no ambiente empresarial. Um fato pode ser representado no diagrama E/R ou por uma entidade **F** ou por um relacionamento n-ário entre as entidades $\mathbf{E}_1$,..., $\mathbf{E}_n$. Se o fato for representado por

um relacionamento, devemos transformar este relacionamento em uma entidade **F** substituindo cada ramo $\mathbf{E}_i$ por um relacionamento binário entre **F** e $\mathbf{E}_i$. Cada relacionamento binário tem multiplicidade (1,1) no lado de **F** e (m_i, M_i) no lado de $\mathbf{E}_i$. Os atributos do relacionamento se tornam atributos de **F** e a chave de **F** é a combinação das chaves de $\mathbf{E}_i$, $i = 1,...,n$.

As entidades ou relacionamentos que representam arquivos freqüentemente atualizados (como Sale) são bons candidatos para se tornarem fatos, sendo que os arquivos mais estáticos, como cadastros, não o são.

Cada fato identificado no diagrama E/R se torna a raiz de um esquema de fatos. No caso de Sale, o maior interesse da análise de negócios é a venda de um produto, que no diagrama E/R da figura 3.9 é representado pelo relacionamento Sale. Na figura 3.10 a transformação deste relacionamento em uma entidade é apresentada.

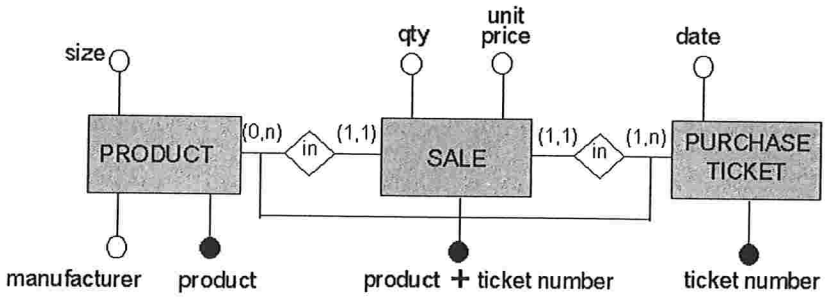


Figura 3.10: Transformação do relacionamento Sale em uma entidade

A criação da árvore de atributos

Uma vez definido o segmento do diagrama ER e a entidade **F** que será trabalhado, pode-se definir a árvore de atributos como sendo a árvore onde:

- Cada vértice corresponde a um atributo do esquema;
- A raiz corresponde à chave de **F**;

- Para cada vértice v, o atributo correspondente determina funcionalmente todos os atributos correspondentes dos descendentes de v.

Para facilitar a construção desta árvore, pode-se aplicar o seguinte algoritmo recursivo sobre a entidade F, que representa o fato a ser analisado:

```
translate(F,identifier(F))
```

onde:

```
translate(E,v):
//E is the current entity, v is the current vertex
{
for each atributte a ∈ E | a ≠ identifier(F)
   do addchild(v,a); //adds child a to vertex v
for each entity G connected to E by a x-to-one relationship R
   do
   {
   for each attribute b ∈ R
      do addChild(v,b);
         addChild(v,identifier(G));
         translate(G,identifier(G));
   }
}
```

Algumas observações:

- Se **F** é identificado pela combinação de dois ou mais atributos, identifier(F) denota esta concatenação;

- Um relacionamento um-para-um pode ser encarado como um tipo particular de um relacionamento muitos-para-um e, por isso, pode ser inserido na árvore de atributos. Porém, em uma pesquisa de data warehouse, a operação de drill down em um relacionamento um-para-um significa acrescentar uma linha sem nenhum nível de

detalhe adicional e este tipo de atributo pode ser representado como um atributo não-dimensional;

- Relacionamentos x-para-muitos não podem ser inseridos na árvore de atributos. De fato, a representação destes relacionamentos no nível lógico seria impossível de ser feita sem que a primeira forma normal fosse desrespeitada;

- Sabe-se que um relacionamento n-ário é equivalente a n relacionamentos binários, sendo que a maioria dos relacionamentos n-ários tem multiplicidade máxima maior que 1 em todos os seus ramos. Este tipo de relacionamento determina, então, n relacionamentos binários um-para-muitos que não podem ser inseridos na árvore de atributos. Porém, um ramo com multiplicidade máxima igual a um determina um relacionamento binário um-para-um que pode ser inserido na árvore.

A árvore de atributos correpondente ao esquema E/R da figura 3.10 é mostrada na figura 3.11.

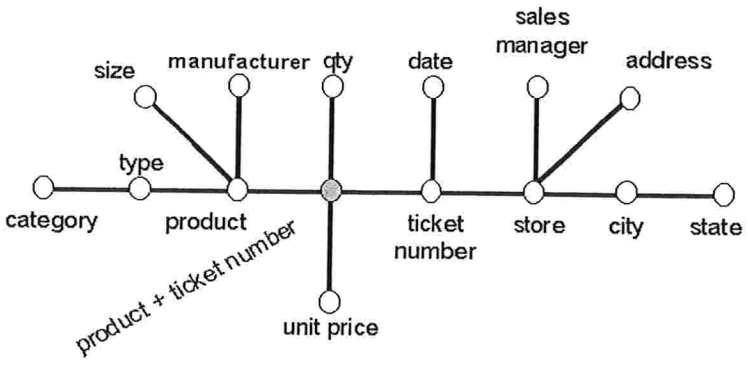


Figura 3.11: Árvore de atributos para o exemplo Sale (a raiz está em cinza)

A adequação da árvore de atributos ("podar"e "enxertar"a árvore)

Como nem todos os atributos representados na árvore são de interesse para o data warehouse, é necessária a adequação da árvore com o objetivo de eliminar os níveis de detalhe desnecessários e, para isso, a árvore deve ser "podada"e "enxertada".

A operação *pruning* ("podar") resulta na eliminação de alguma subárvore. Os atributos eliminados não serão incluídos no esquema de fatos e, por esta razão, não poderão ser utilizados para agregações de dados. Como exemplo, no caso de Sale, a subárvore que inclui city e state poderia ser eliminada.

A operação *grafting* ("enxertar") é usada quando um determinado vértice da árvore expressa uma informação desnecessária, porém, seus descendentes devem ser preservados. Como exemplo, alguém pode querer classificar os produtos diretamente por categoria (category) sem considerar a informação do seu tipo (type).

Seja v o vértice a ser eliminado e v', seu pai, a operação *grafting* pode ser implementada através do algoritmo abaixo:

```
graft(v):
{
for each v"| v"is child of v
   do addChild(v',v");
drop v;
}
```

Como se pode verificar, a operação *grafting* move toda a subárvore enraizada em v para v'. Com isto, o atributo v não será incluído no esquema de fatos e seu correspondente nível de agregação será perdido, mantendo-se, porém, todos os níveis de seus descendentes. No caso de Sale, o detalhe dos tíquetes de compras não interessam, portanto, deve-se aplicar a operação *grafting* no vértice ticket number, tendo como resultado a árvore representada na figura 3.12.

De forma geral pode-se dizer que, se a operação *grafting* for aplicada em um filho da raiz, é obtido um nível mais alto de granularidade das instâncias do fato e que, se o nó que sofre a operação *grafting* tiver 2 ou mais filhos, isto levará a um aumento no número de dimensões no esquema de fatos. Além disso, se um vértice opcional sofre a mesma operação, todos os seus filhos herdam sua opcionalidade.

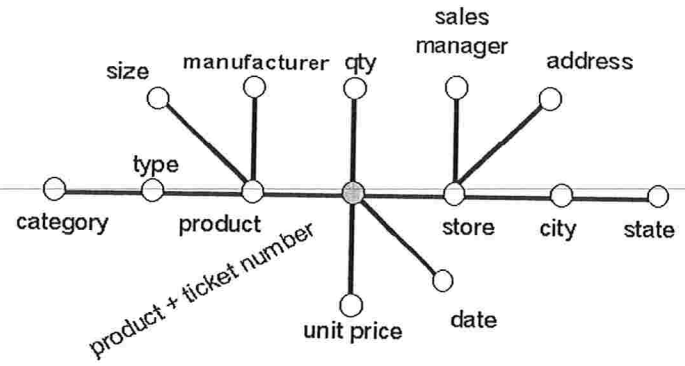


Figura 3.12: Árvore de atributos para o exemplo Sale após a operação *grafting* ("enxertar") sobre o vértice ticket number

A definição das dimensões

As dimensões vão determinar como as instâncias dos fatos podem ser agregadas de forma significativa para o processo de tomada de decisão. As dimensões serão escolhidas entre os vértices que são filhos da raiz (incluindo os vértices que se tornaram filhos da raiz após as operações de *grafting*). A escolha das dimensões é de grande importância já que isto determina a granularidade das instâncias dos fatos.

A dimensão tempo é um elemento chave em todo data warehouse, porém os diagramas E/R podem ser classificados como *snapshot* ou temporais, dependendo da forma com que tratam o tempo. Os esquemas *snapshot* descrevem o estado atual do domínio da aplicação enquanto que os esquemas temporais descrevem a evolução do domínio da aplicação durante um intervalo de tempo, ou seja, as versões antigas dos dados são explicitamente representadas e armazenadas. Nos esquemas temporais, o tempo está representado e, portanto, é um candidato óbvio para uma dimensão. Se o atributo relacionado ao tempo aparecer na árvore de atributos como filho de algum vértice diferente da raiz, a árvore deve sofrer operações *grafting* para que o tempo se torne uma dimensão, ou seja, se torne um filho da raiz. No caso dos esquemas do tipo *snapshot*, o tempo não está explicitamente representado, já que é assumido que o esquema representa os dados no momento atual. Ainda assim, para este tipo de esquema, o tempo deve ser adicionado como uma dimensão

do esquema de fatos.

No exemplo Sale, que está sendo analisando, os atributos escolhidos como dimensões são: product, store e intervalos de date que correpondem às semanas.

A definição dos atributos do fato

Normalmente, os atributos do fato são contadores do número de instâncias de F ou a soma, média, máximo ou mínimo de expressões que evolvem atributos numéricos da árvore de atributos (são excluídos os atributos que já foram escolhidos como dimensões). Pode acontecer também que um fato não possua atributos e, neste caso, a informação é armazenada apenas para indicar a ocorrência do fato. Caso estes atributos existam, é o momento de se criar um glossário com as expressões que os descrevem. Por exemplo:

quantity sold = sum(sale.qty)

total returns = sum(sale.qty * sale.unitPrice)

number of customers = count(sale)

Os operadores de agregação podem ser aplicados a todas instâncias de Sale que sejam da mesma semana, loja e produto.

A definição das hierarquias

Este é o último passo na criação do esquema de fatos. Ao longo de cada hierarquia, os atributos devem ser organizados em uma árvore, onde o relacionamento entre cada nó e seus descendentes é do tipo x-para-um. Neste estágio, ainda é possível realizar operações de *pruning* e *grafting* para eliminar detalhes irrelevantes, como por exemplo, um vértice conectado a seu pai através de um relacionamento do tipo um-para-um. Também é possível adicionar novos níveis de agregação dentro da hierarquia. Isto ocorre tipicamente na dimensão tempo. No exemplo Sale, foram introduzidos os níveis month e week, permitindo a agregação dos valores numéricos em meses e semanas. Durante esta etapa, os atributos que não serão utilizados para agregação, mas apenas para propósitos informativos devem ser identificados como atributos não dimensionais.

Ao término desta etapa, chega-se a uma definição gráfica do modelo conceitual do data warehouse, conforme apresentado na figura 3.5.

3.4 Os esquemas básicos e suas variações

Após a definição do modelo conceitual, podemos partir para o modelo lógico e sua conseqüente implementação física. A implementação física do modelo dimensional, em uma base de dados relacional, pode ser feita de várias formas diferentes. No entanto, para todos os esquemas, os fatos e as dimensões são implementados em tabelas físicas, onde cada linha é única e identificada por uma chave genérica, já discutida anteriormente. Nas próximas seções, são analisados os esquemas básicos, conhecidos como *Star schema* e *Snowflake schema*, e suas variações.

3.4.1 O esquema Star clássico

Provavelmente, o *Star schema*, apresentado por Ralph Kimball [Kim96], seja o primeiro esquema utilizado para desenhar o data warehouse implementado em um banco de dados relacional. Neste esquema, a tabela fato contempla as seguintes características:

- Uma chave primária composta, sendo um elemento da chave para cada dimensão;
- Cada elemento chave para a dimensão deve ser representado e descrito na tabela dimensão correspondente, para efetuar o *join*;
- Opcionalmente, uma ou mais métricas ou fatos a serem avaliados;
- Pode armazenar as métricas sumariadas em diferentes níveis de agregação.

Uma tabela dimensão é definida para cada dimensão do negócio a ser analisada, contendo:

- Uma chave genérica, gerada pelo sistema;

- Uma coluna de descrição genérica para a dimensão ou colunas de descrição para cada atributo da hierarquia;
- Atributos que permitam efetuar os filtros;
- Um indicador nível, para estabelecer o nível da hierarquia a que se refere a linha da tabela;
- Valores nulos ocasionais, dependendo do nível hierárquico da linha.

O objetivo de se armazenar toda a hierarquia é possibilitar, durante as consultas, a utilização de filtros que permitam a agregação ou sumário dos fatos nos vários níveis hierárquicos.

As figuras 3.13(a) e 3.13(b) mostram os detalhes de uma tabela dimensão, chamada Geografia que descreve e define uma hierarquia entre a loja, a cidade onde se encontra, o estado (UF) e a região. Na figura 3.13(a) encontra-se representada uma das possibilidades, com apenas uma descrição genérica para toda a dimensão e a figura 3.13(b) apresenta colunas de descrição específicas para cada elemento da hierarquia. É importante notar que não existe uma preocupação com a normalização das tabelas no data warehouse, conforme já exposto na seção 3.2.2. Este é um dos aspectos que diferencia bastante a modelagem do data warehouse da modelagem convencional. Outro aspecto a destacar é que o indicador nível não precisa ser numérico e pode conter o nome do nível da hierarquia, como por exemplo: "cidade", "loja", "mês"etc.

A dimensão Tempo é uma dimensão especial, sendo diferente de todas as outras dimensões. Esta dimensão requer alguns atributos comuns a todas as dimensões e três atributos especiais que serão de extrema importância no momento das consultas.

As colunas que devem compor a dimensão Tempo são:

- **Chave única genérica**, gerada pelo sistema, preferencialmente numérica e inteira, para ligação lógica com a tabela fato;

a.

GEOGRAFIA_KEY	Geografia Descrição	Região	Estado	Cidade	Loja	Nível
1010	Região Sul	1				4
1020	Região Sudeste	2				4
2010	SP	2	10			3
2020	RJ	2	20			3
3010	São Paulo	2	10	101		2
3020	Campinas	2	10	102		2
4010	Loja ABC	2	10	101	1001	1
4020	Loja DEF	2	10	102	1002	1
4030	Loja GHI	2	10	102	1003	1

b.

GEOGRAFIA_KEY	Região	Região_desc	Estado	Estado_desc	Cidade	Cidade_desc	Loja	...	Nível
1010	1	Região Sul						...	4
1020	2	Região Sudeste						...	4
2010	2	Região Sudeste	10	São Paulo				...	3
2020	2	Região Sudeste	20	Rio de Janeiro				...	3
3010	2	Região Sudeste	10	São Paulo	101	São Paulo		...	2
3020	2	Região Sudeste	10	São Paulo	102	Campinas		...	2
4010	2	Região Sudeste	10	São Paulo	101	São Paulo	1001	...	1
4020	2	Região Sudeste	10	São Paulo	102	Campinas	1002	...	1
4030	2	Região Sudeste	10	São Paulo	102	Campinas	1003	...	1

Figura 3.13: (a)Tabela dimensão em detalhes com apenas uma descrição (genérica) (b)Tabela dimensão com uma descrição para cada nível da hierarquia - Star expandido

- **Descrição** do tempo contido naquela linha. Conforme os exemplos citados na figura 3.13, esta descrição pode ser única para a tabela ou específica para cada nível da hierarquia;

- **Atributos da hierarquia**, como nas outras dimensões;

- **Nível** ou **Resolução**, que indica o nível da hierarquia representado naquela linha, que pode ser numérico ou um pequeno texto;

- **Seqüência na resolução**, que indica a seqüência de um período de tempo dentro do nível ou resolução ao qual ele pertence. Este valor é um número inteiro consecutivo que designa a ordem cronológica para todas os dias, meses, semestres, anos etc, ou seja, para todos os períodos de tempo existentes na tabela dimensão. A seqüência é definida dentro de cada nível ou período de resolução, sem repetições, e [IA99] recomenda que sejam utilizados intervalos de valores para estabelecer a seqüência. Por exemplo, para a seqüência de anos a numeração seria de 1 a 10, para semestres, entre 100 e 200, todos os meses estariam entre 500 e 999 e todos os dias teriam

seqüências numeradas a partir de 1.000. A figura 3.14 mostra que a seqüência na resolução para Janeiro de 1999 é 500, para Fevereiro de 1999 é 501, Março de 1999 tem seqüência 502 e assim por diante. É importante ressaltar novamente que o valor para esta coluna deve ser único dentro de um período de resolução;

- **Indicador Corrente**, se utilizado em conjunto com a coluna de seqüência na resolução, descrita no item anterior, indica qual o período de tempo dentro da resolução especificada, que deveria ser definido como corrente. Com isto, é possível definir qual será o tempo corrente para as consultas: a informação carregada mais recentemente, a data de hoje ou algum período de tempo futuro. A utilização desta coluna permite que determinadas condições de negócio estabeleçam o momento em que os dados estarão disponíveis para os usuários finais. Esta coluna é, normalmente, definida como um caracter e preenchida com 'S' ou 'N', onde 'S' indica que o dado já foi carregado no data warehouse. O período de tempo corrente, dentro da resolução (pode ser o dia corrente, o mês corrente etc.), será aquele que contiver na coluna Indicador Corrente o valor de 'S' e o maior valor na coluna Seqüência na Resolução, descrita no item anterior.

Nem sempre é possível capturar todas as métricas e dimensões importantes para um modelo de negócios, em um único esquema *Star*. Ao contrário, a existência de múltiplas tabelas fato é a norma, definindo um modelo de múltiplas estrelas. Cada estrela é composta por uma tabela fato e suas dimensões associadas [AV98].

O principal fator que pode levar à separação das métricas em diferentes tabelas fato é o modelo de negócios analisado. Um processo de negócios pode envolver diferentes conjuntos de dimensões e as métricas relacionadas a cada processo podem estar sendo coletadas em diferentes intervalos de tempo.

O esquema *Star* é desenhado para simplicidade e velocidade nas consultas. Podemos citar como vantagens, na sua utilização, os seguintes aspectos:

- Facilidade de entendimento do modelo, principalmente por parte do usuário final;

Tempo key	Descrição	Resolução ou Nível	Sequência na Resolução	Indicador Corrente	Ano	Semestre	Mês	Dia
...	...	...	...	...	...	...	...	...
101	Janeiro 1999	Mês	501	S	1999	011999	011999	
102	Fevereiro 1999	Mês	502	S	1999	011999	021999	
103	Março 1999	Mês	503	S	1999	011999	031999	
104	Abril 1999	Mês	504	S	1999	011999	041999	
105	Maio 1999	Mês	505	S	1999	011999	051999	
106	Junho 1999	Mês	506	S	1999	011999	061999	
113	Janeiro 2000	Mês	513	S	2000	012000	012000	
114	Fevereiro 2000	Mês	514	S	2000	012000	022000	
115	Março 2000	Mês	515	S	2000	012000	032000	
116	Abril 2000	Mês	516	S	2000	012000	042000	
21	1 Sem. 1999	Semestre	101	S	1999	011999		
22	2 Sem. 1999	Semestre	102	S	1999	021999		
23	1 Sem. 2000	Semestre	103	S	2000	012000		
24	2 Sem. 2000	Semestre	104	S	2000	022000		
3	1999	Ano	3	S	1999			
4	2000	Ano	4	S	2000			
5	2001	Ano	5	N	2001			
6	2002	Ano	6	N	2002			
...	...	...	...	...	...	...	...	...

Figura 3.14: Tabela representando a dimensão Tempo, com as colunas específicas para este tipo de dimensão

- Excelente desempenho, devido ao uso de chaves genéricas, geralmente numéricas e inteiras;
- Facilidade na definição das hierarquias;
- Número de operações de *join* reduzidas, já que o modelo implementa todos os níveis da hierarquia das dimensões em apenas uma tabela por dimensão além de estar bastante desnormalizado;
- O metadados é simples.

Na figura 3.15 está representado um esquema *Star* clássico, composto por uma tabela fato e suas dimensões Geografia, Produto e Tempo associadas.

Apesar de amplamente utilizado, o esquema *Star* apresenta alguns problemas. O primeiro deles está relacionado ao indicador nível, presente em todas as tabelas dimensão. Este indicador deve ser utilizado em todas as consultas, já que a mesma tabela dimensão armazena informações sobre todos os níveis da hierarquia. Durante a construção da

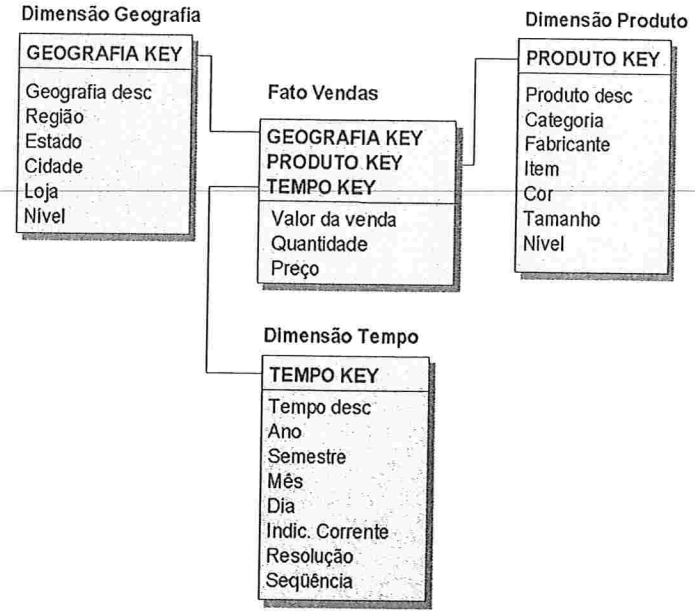


Figura 3.15: Representação de um data warehouse em esquema *Star* clássico

consulta, o usuário deve conhecer a estrutura da base de dados, para poder especificar o nível da informação a ser pesquisada. Este indicador pode ser causa potencial para erros já que, se for esquecido ou mal utilizado, pode levar a resultados incorretos e baixa flexibilidade. Além disso, se a estrutura da dimensão for alterada, incluindo ou eliminando algum nível da hierarquia, o data warehouse necessitará de alterações físicas, elevando os custos de manutenção.

O segundo problema é que, por manter todos os níveis hierárquicos da dimensão fisicamente em uma única tabela, esta se torna muito grande, em vários casos. Outro fator que contribui para o aumento do tamanho da tabela é a desnormalização, que gera uma grande quantidade de valores de atributos repetidos. A desnormalização fica bastante clara, ao analisarmos, por exemplo, a dimensão Geografia, que aparece na figura 3.13(b). No diagrama, representado na figura 3.16, podemos ver as dependências funcionais que ocorrem em uma tabela deste tipo.

O esquema *Star* clássico pode apresentar algumas variações, dependendo do problema de negócio apresentado. As seções seguintes descrevem estas variantes do *Star*.

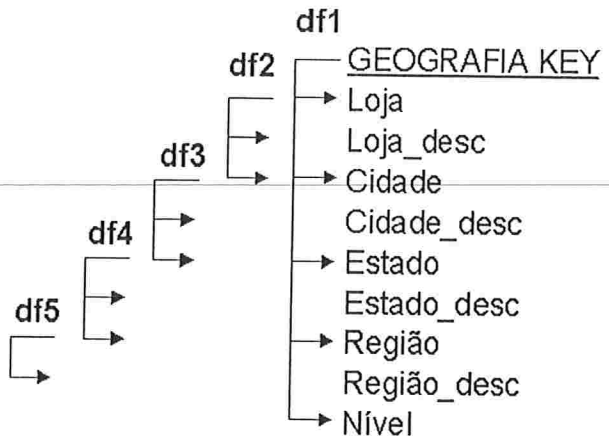


Figura 3.16: Dependências funcionais da tabela dimensão Geografia

3.4.2 As variações do esquema Star

A primeira variação a ser discutida é o esquema *Partial Star* [Tan97]. Nesta variação existem múltiplas tabelas para cada dimensão e para a fato. Estas múltiplas tabelas estão lógica e fisicamente separadas em função dos seus níveis de agregação. Este modelo cria múltiplas estrelas, cada uma delas representando uma combinação de níveis de agregação em cada dimensão. Não existem ligações lógicas entre as várias tabelas fato ou dimensão, apenas entre a tabela dimensão e a tabela fato de cada grupo. Cada dimensão é representada por múltiplas tabelas que são fisicamente separadas pelo nível de agregação, sendo que as tabelas possuem como chave primária, uma chave genérica gerada pelo sistema, da mesma forma que o esquema *Star* clássico. Pode acontecer que uma métrica ocorra apenas para um determinado nível de uma dimensão. Por exemplo, a métrica Preço existe unicamente no nível Item da dimensão Produto. A figura 3.17 retrata esta variação do *Star*, representando apenas a dimensão Geografia, separada nos níveis de Região e de Cidade, e as tabelas fatos associadas a estes níveis.

Esta partição das informações, pelos níveis hierárquicos de agregação, leva à criação de duas estrelas: a primeira representando o nível de Região e a segunda representando o nível de Cidade. Ao se adicionar a este esquema a dimensão Tempo e a dimensão

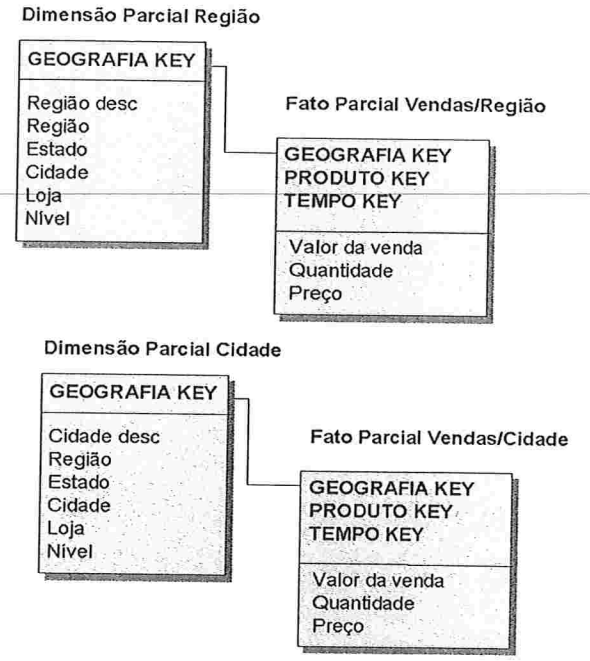


Figura 3.17: Esquema *Partial Star* mostrando apenas a dimensão Geografia e as tabelas fato associadas

Produto agregadas nos níveis de Fabricante e Marca, conforme o modelo representado na figura 3.15 (*Star* Clássico), expande-se para Região/Fabricante/Ano, Região/Marca/Ano, Cidade/Fabricante/Ano e Cidade/Marca/Ano, sendo que cada combinação será representada por uma estrela parcial separada, conforme a figura 3.18.

São várias as vantagens encontradas no esquema *Partial Star*, apesar de sua visível complexidade. Uma delas é a possibilidade de maior controle do tempo de carga, *backup* e manutenção das tabelas, pois seu tamanho fica reduzido em função da partição. Estas partições possibilitam também a existência de fatos em certos níveis específicos e é possível eliminar as colunas que não tenham significado em determinados níveis, reduzindo assim a esparsidade. Como já foi mencionado, a complexidade do modelo aumenta em relação ao *Star* clássico e cada estrela requer uma definição específica no metadados, dificultando o processo de manutenção. Em relação às consultas, serão necessários múltiplos comandos SQL para analisar dados em mais de um nível de agregação em um mesmo relatório, sendo

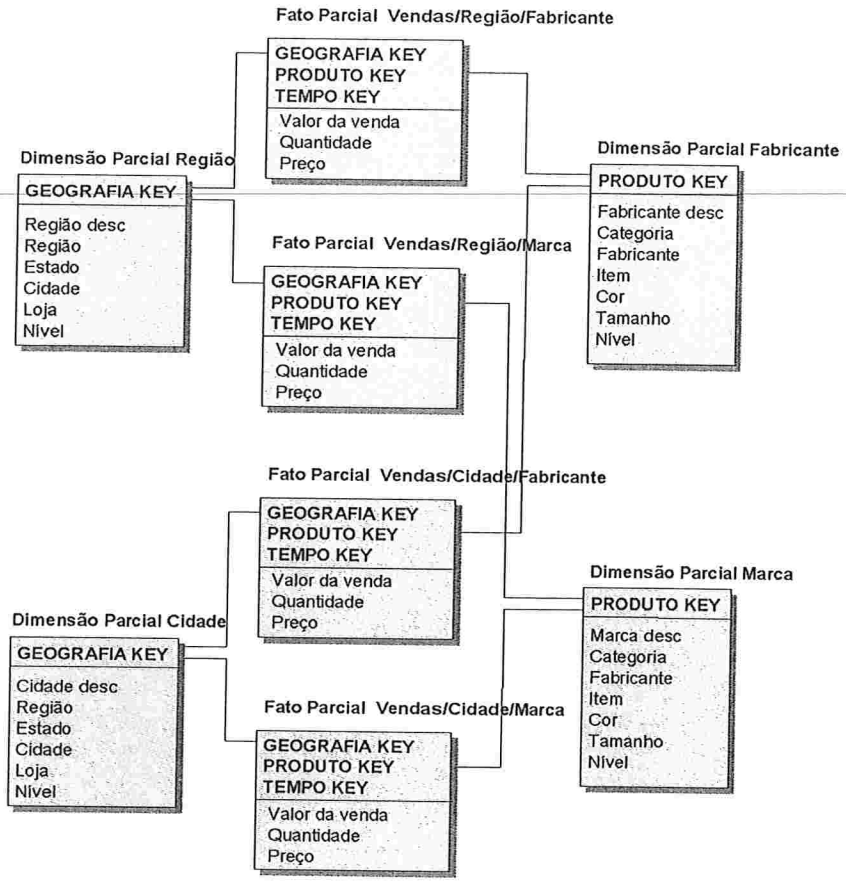


Figura 3.18: Esquema *Partial Star* mostrando as dimensões Geografia e Produto separadas em diferentes níveis de agregação e as tabelas fato associadas

que esta condição pode degradar o desempenho da consulta. Além disso, a estrutura física de cada tabela ou grupo de tabelas requer alterações sempre que novos níveis de agregação ou novas combinações forem adicionados ou removidos.

Para simplificar um pouco o esquema *Partial Star*, utiliza-se uma combinação de seus princípios com o *Star* clássico. Pode-se particionar apenas a tabela fato e manter cada dimensão como uma única tabela ou o inverso, ou seja, manter a tabela fato única e particionar as dimensões. O particionamento é feito sempre baseado na agregação de determinados níveis da hierarquia. Estas variações são chamadas de *Fact Partitioning* e *Dimension Partition*, respectivamente. A variação *Fact Partitioning*, também chamada

Fact Constellation é o esquema que particiona apenas a tabela fato. Este esquema utiliza uma única tabela para cada dimensão que fica ligada às múltiplas tabelas fato, particionadas pelos diferentes níveis de agregação. Este esquema possibilita a existência de fatos em certos níveis, já que as tabelas são particionadas, assim como o *Partial Star*. Se a informação for particionada em níveis de agregação desde o nível mais detalhado, o desempenho, no caso de consultas executadas nos níveis de detalhe mais altos, será melhor. Deve-se levar em conta que, na implementação deste esquema, se a hierarquia é extensa, numerosas tabelas fato serão criadas, aumentando a complexidade do desenho. Além disso, consultas que requeiram a análise das informações em mais de um nível de agregação emitirão múltiplos comandos SQL, o que pode acarretar em um desempenho inferior. A figura 3.19 representa um exemplo desta variação, onde a tabela fato está particionada em função dos níveis hierárquicos Região e Cidade, da dimensão Geografia (as outras dimensões não estão representadas).

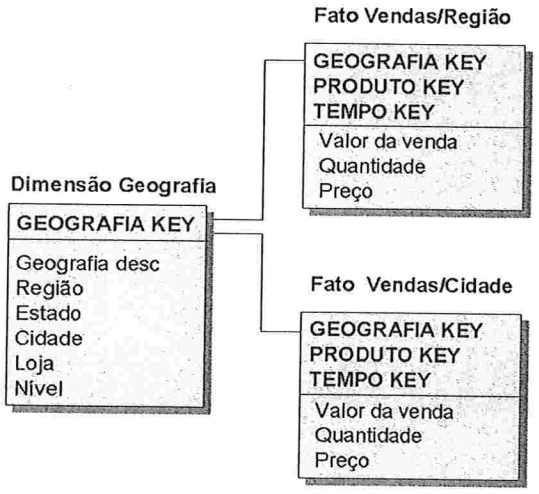


Figura 3.19: Exemplo do esquema *Fact Partitioning* ou *Fact Constellation* para a dimensão Geografia e as tabelas fato correspondentes às partições por Região e Cidade

A variação *Dimension Partitioning* mantém uma única tabela fato, que contém as métricas dos vários níveis de agregação, incluindo o nível mais detalhado. Esta tabela

fato está logicamente ligada às múltiplas tabelas de dimensão, particionadas por diferentes níveis de agregação. A figura 3.20 representa um esquema deste tipo, onde a dimensão Geografia encontra-se particionada por Região e por Cidade (as outras dimensões não estão representadas). As vantagens da *Dimension Partitioning* incluem a possibilidade de assinalar atributos que são específicos cada nível de agregação e um bom desempenho, já que apenas um comando SQL é emitido para a tabela fato, sem a necessidade de efetuar *joins*. Ao mesmo tempo, a vantagem de manter todos os fatos, detalhados e sumariados na mesma tabela, dificulta a recuperação das informações dos níveis mais altos.

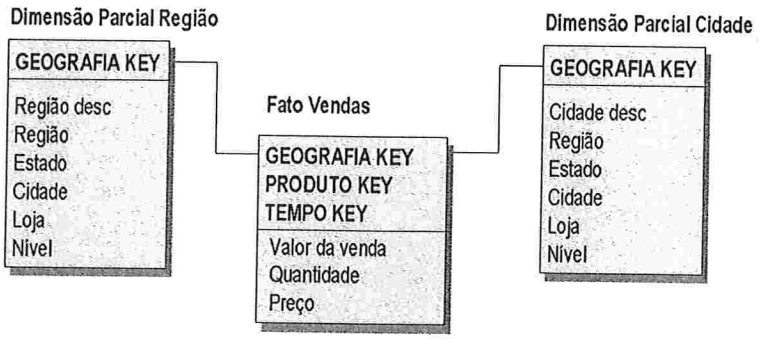


Figura 3.20: Implementação do esquema *Dimension Partitioning*, representando o particionamento da tabela dimensão Geografia

A escolha do esquema a ser implementado, deve levar em conta vários fatores do negócio, bem como os tipos de consultas e o tamanho de cada uma das tabelas definidas no data warehouse. Seja qual for a variação escolhida, o *Star* se carateriza pela simplicidade, rapidez no acesso e desnormalização. O esquema *Star* pode ser refinado e transformado em um esquema chamado *Snowflake* que, para implementar as hierarquias dos atributos, se utiliza de tabelas de subdimensões. Estas subdimensões facilitam a normalização do modelo. Manter as tabelas desnormalizadas, implementando-se um esquema Star é um aspecto bastante discutível, pois a divisão das tabelas, como em um esquema *Snowflake*, em nome da normalização, pode levar as consultas a um desempenho mais baixo. O *Snowflake* pode ser considerado um *Star* normalizado e nas próximas seções tanto este

esquema como suas variações são analisados.

3.4.3 O esquema Snowflake

O esquema *Snowflake* emprega uma combinação de normalização da base de dados, para manter a integridade e reduzir os dados armazenados de forma redundante, com uma desnormalização para obter melhor desempenho [Tan97]. Neste esquema as dimensões são normalizadas em subdimensões, sendo que cada nível da hierarquia fica em uma subdimensão. Por esta razão, não há necessidade de se utilizar o indicador de nível que existe nos esquemas do tipo *Star*. A tabela principal da dimensão tem uma chave para cada nível hierárquico representado na subdimensão e não mais uma única chave, como no *Star*.

O *Snowflake* apresenta duas variações básicas que diferem na disposição das tabelas que representam as subdimensões, os *Snowflake Lookup* e o *Snowflake Chain*, descritos na próxima seção. Sua representação gráfica fica similar a um floco de neve, devido ao particionamento das tabelas dimensão.

3.4.4 As variações do esquema Snowflake

O esquema *Snowflake Lookup* emprega tabelas adicionais para nomes e descrições dos atributos, todas ligadas a uma tabela principal da dimensão. Desta forma é possível reduzir o tamanho da tabela dimensão, eliminando a redundância do armazenamento das mesmas descrições em várias linhas diferentes, sendo que as tabelas adicionais atuam como tabelas de *lookup* para a chave ou valores codificados da tabela principal da dimensão que, por sua vez, está logicamente ligada a uma única tabela de fatos. A figura 3.21 mostra o mesmo exemplo citado na figura 3.15, porém modelado em *Snowflake Lookup*. Na figura 3.21 está representada apenas a dimensão Gegrafia e suas subdimensões Região, Estado, Cidade e Loja.

Na figura 3.21, pode-se notar que a ligação entre a tabela fato e a tabela da dimensão principal é feita da mesma forma que no esquema *Star*, através de uma chave gerada

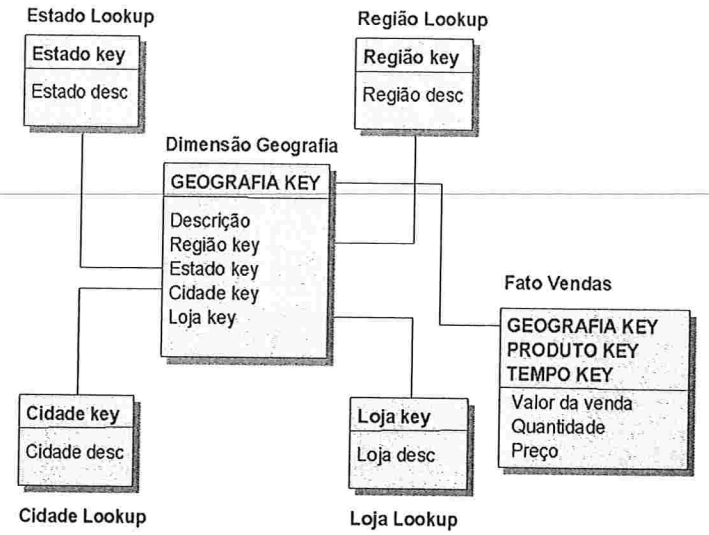


Figura 3.21: Desenho lógico da dimensão Geografia em um data warehouse implementando o esquema *Snowflake Lookup*

genérica. A tabela principal da dimensão se conecta logicamente às subdimensões, que são as tabelas de *lookup*, através de suas chaves primárias.

As descrições não precisam ser repetidas como no esquema *Star*, simplificando o armazenamento, reduzindo o tamanho relativo das tabelas dimensão e melhorando o controle de integridade dos dados. O uso das chaves geradas genéricas, geralmente números inteiros, levam a um melhor desempenho, menor manutenção do metadados e mais flexibilidade ao data warehouse. Vale ressaltar que o desempenho pode ficar afetado se múltiplas consultas ou múltiplos *joins* têm que ser emitidos para decodificar as chaves, ao buscar as descrições nas tabelas adicionais. Além disso, a manutenção da base de dados requer manutenção adicional dado o número maior de tabelas físicas distintas.

O esquema *Snowflake* apresenta um nível muito maior de normalização, se comparado aos esquemas do tipo *Star*. A figura 3.22 mostra os diagramas que representam as dependências funcionais do esquema que aparece na figura 3.21 em *Snowflake Lookup*.

A segunda variação do esquema *Snowflake*, conhecida por *Snowflake Chain*, utiliza também subdimensões, particionadas pelos níveis hierárquicos da dimensão, sendo que a

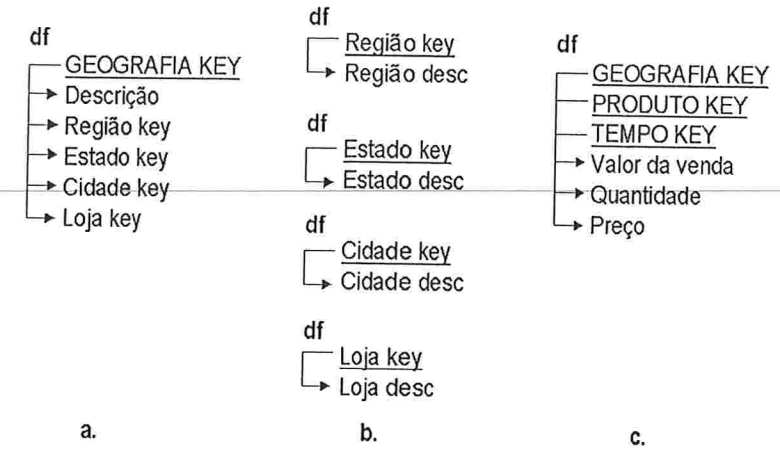


Figura 3.22: Diagramas representando as dependências funcionais em: (a)Tabela principal da dimensão Geografia; (b)Tabelas de Lookup (subdimensões) e (c)Tabela Fato Vendas

tabela principal da dimensão representa o nível mais baixo (mais detalhado) da hierarquia. As subdimensões estão encadeadas, partindo-se da tabela fato que está logicamente conectada à tabela da subdimensão principal ou raiz. Na figura 3.23 encontramos um exemplo desta implementação, representando a dimensão Geografia com suas subdimensões e a tabela fato de Vendas.

Cada tabela da subdimensão contém sua chave primária e suas descrições associadas. Além disso, contém também a chave para o próximo nível da hierarquia da dimensão e assim por diante, até chegarmos à última subdimensão, que contém o nível mais alto (menos detalhado) da hierarquia. É tipicamente utilizada quando os fatos contêm informações apenas sobre o nível de detalhe mais baixo da hierarquia. Como podemos verificar na figura 3.23, a tabela fato já não utiliza uma chave para a dimensão Geografia toda, a chave utilizada é diretamente para a subdimensão principal. Fica claro que esta implementação não é recomendada quando os relatórios necessitam freqüentemente vários níveis de agregação da informação, já que são necessários vários passos na cadeia para se chegar ao resultado. Este esquema oferece um alto grau de integridade de dados, pois os nomes e as descrições são mantidos em um único local, reduzindo o tamanho das tabelas de dimensão. A maior desvantagem está no baixo desempenho, uma vez que todos os

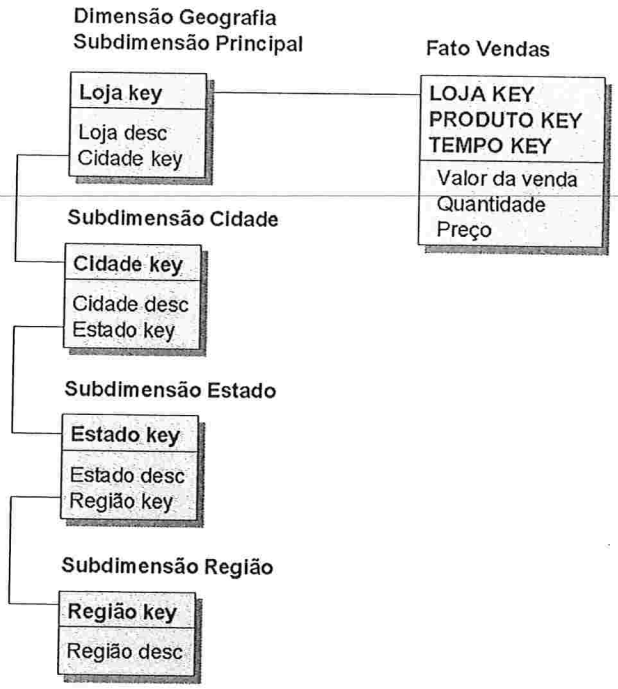


Figura 3.23: Exemplo de implementação do esquema *Snowflake Chain*, representando a ligação da tabela Fato com a dimensão Geografia (dividida em subdimensões encadeadas)

níveis da cadeia devem ser acessados, mesmo quando se requer apenas os níveis mais altos de agregação. Em situações práticas, existe uma alternativa para minimizar este efeito, adicionando-se, a cada subdimensão, chaves para todos os níveis superiores da hierarquia. Com isso, pode-se "saltar" determinados níveis, utilizando-se a chave que leva diretamente para o nível requerido. A figura 3.24 apresenta esta alternativa, que visa basicamente um melhor desempenho do *Snowflake Chain*.

Podem ocorrer, em determinados modelos, vários atributos diferentes que não estão associados a nenhuma dimensão específica. Em vez de criarmos várias dimensões com apenas um atributo, é possível agrupá-los em apenas uma dimensão. Este esquema é chamado de *Snowflake Attribute* [Tan97], sendo que a dimensão artificial pode também implementar a combinação de dimensões pouco utilizadas com o objetivo de reduzir o número de dimensões do data warehouse, simplificar o modelo e melhorar o desempenho. A tabela principal desta dimensão artificial contém as chaves estrangeiras dos diferentes

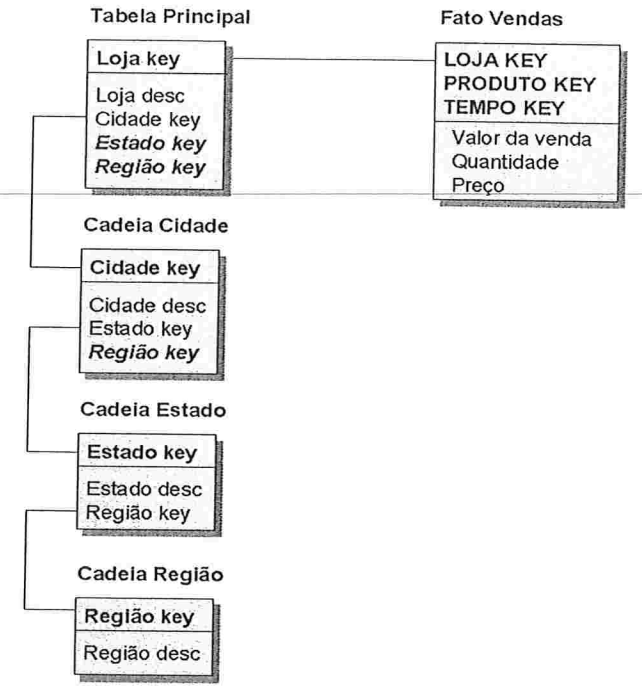


Figura 3.24: Exemplo de *Snowflake Chain* com chaves adicionais para melhorar o desempenho das consultas

atributos ou das tabelas dimensão a serem agrupadas. A chave primária da dimensão artificial é uma chave genérica gerada pelo sistema e uma única linha é adicionada para cada combinação válida de todos os atributos ou dimensões envolvidas. Cada chave estrangeira da tabela principal leva a uma chave primária na tabela de descrição daquele atributo ou dimensão. Na figura 3.25, encontra-se um exemplo, onde Tamanho, Cor, Formato e Perfume são atributos não dimensionais dos produtos vendidos, que não podem ser associados a nenhuma das dimensões existentes no modelo. Utilizando-se este tipo de esquema, os atributos podem ser agrupados em uma única dimensão artificial que contém todas as combinações válidas destes atributos, sendo que a tabela fato contém uma chave para conectar-se logicamente a esta dimensão, da mesma forma que se conecta às outras dimensões do modelo.

As vantagens encontradas neste tipo de implementação são várias: redução do número de dimensões, que leva à redução do tamanho global dos índices e da complexidade do

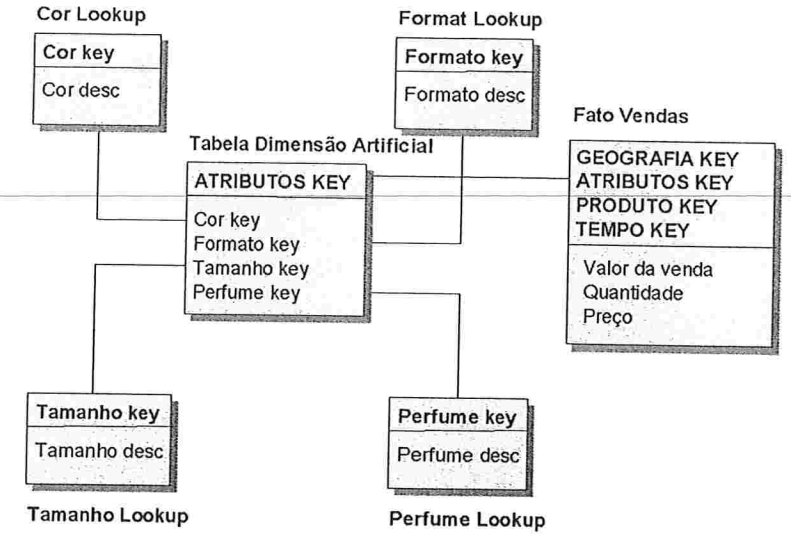


Figura 3.25: Exemplo do esquema *Snowflake Attribute* agrupando os atributos Cor, Formato, Tamanho e Perfume, características dos produtos vendidos

SQL gerado, maior integridade dos dados e os vários atributos juntos ou dimensões pouco usadas compartilham de um único ponto de entrada na tabela de fatos. Sem este tipo de esquema, cada um destes atributos ou dimensões seria mantido como uma chave estrangeira na tabela fato, resultando em um ponto de entrada que não faria parte da chave primária e, provavelmente, sem índice físico para facilitar o acesso. Esta situação, conforme retratada na figura 3.26, deve ser evitada.

3.5 As agregações das informações

Apesar dos dados no data warehouse serem armazenados segundo a granularidade definida, muitas das consultas realizadas necessitam, além das informações detalhadas, de informações sumariadas ao longo das dimensões. A informação armazenada no nível de detalhe é importante, porém o acesso à informação em níveis sumariados permite aos analistas de negócio terem uma visão global do modelo de negócios analisado. Estas consultas, partindo de uma base onde existem apenas os dados de nível básico, ou seja, do nível mais detalhado, se tiverem que sumariar os dados no momento da execução, sobre-

carregarão todo o processo de análise. De acordo com [Kim96], é necessário um determinado conjunto de vários agregados precomputados para acelerar cada uma das consultas, sendo que o efeito sobre o desempenho é considerável, chegando a obter reduções de dez a mil vezes no tempo de processamento. Por esta razão, a utilização de dados previamente sumariados é um recurso bastante eficiente para controlar o desempenho do data warehouse. Nesta seção, estaremos analisando aspectos da definição, implementação e utilização destes valores agregados.

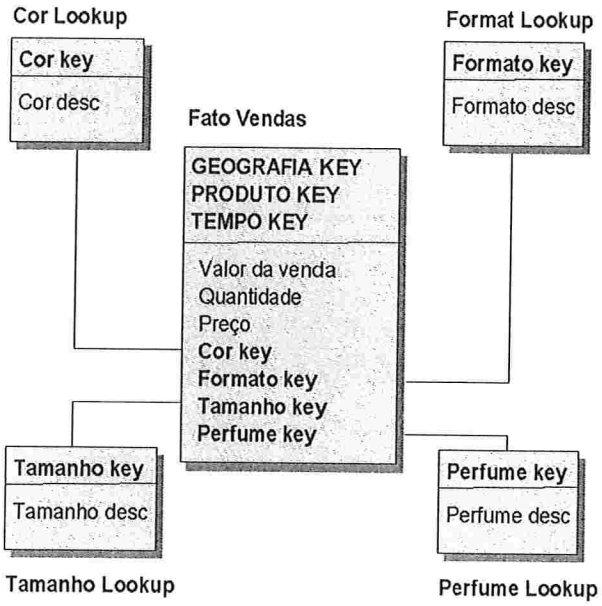


Figura 3.26: Implementação alternativa sem a utilização do esquema *Snowflake Attribute*

3.5.1 A definição dos agregados

A escolha dos agregados a serem previamente armazenados no data warehouse, já que armazenar todos os agregados é impraticável, não é uma tarefa fácil. É necessária uma análise das consultas que serão executadas pelos usuários, identificando os agrupamentos mais utilizados, além de constante acompanhamento após sua implantação.

Uma forma de se descrever as agregações a serem aplicadas na base é como um "agregado n-direcional", onde n indica o número de dimensões sumariadas como resultado da

agregação. Um exemplo de um agregado *uni*-direcional seria sumariar uma dimensão (em algum nível da hierarquia) enquanto que as outras dimensões ficam em seu nível atômico, mais detalhado. Um agregado *bi*-direcional agrega um nível da hierarquia para duas dimensões deixando as outras dimensões em seu nível atômico. Ao utilizar o exemplo citado na figura 3.15, que representa um esquema *Star* clássico, com três dimensões Geografia, Produto e Tempo e implementar no data warehouse informações sumariadas para totais de Categoria na dimensão Produto, totais de Estado na dimensão Geografia e totais mensais na dimensão Tempo, obtém-se sete tipos de agregados diferentes:

1. Agregado unidirecional: totais de categoria por loja por dia;
2. Agregado unidirecional: totais de cidade por item de produto por dia;
3. Agregado unidirecional: totais mensais por item produto por loja;
4. Agregado bidirecional: totais de categoria por totais de cidade por dia;
5. Agregado bidirecional: totais de categoria por totais mensais por loja;
6. Agregado bidirecional: totais de cidade por totais mensais por item de produto;
7. Agregado tridirecional: totais de categoria por totais de cidade por totais mensais.

Neste exemplo fica claro que não estão sendo representados todos os possíveis agregados para as dimensões do modelo. Como já mencionado, isto seria impraticável, dada a grande quantidade de dados a ser administrada. Ainda assim, para os agregados escolhidos é necessária uma avaliação do tamanho do resultado a ser gerado, levando-se em conta a explosão do número de linhas e a dispersão das informações armazenadas no data warehouse. Uma tabela de fatos de nível básico, para este exemplo, é normalmente bastante dispersa no preenchimento das chaves [Tan97]. Para entender este aspecto, deve-se que lembrar que apenas uma parte dos produtos são vendidos por dia em cada uma das lojas. Porém, ao se calcular o tamanho do conjunto de agregados, deve-se levar em conta que sua criação aumenta drasticamente a taxa de ocupação, aumentando bastante o tamanho do

data warehouse. Para comprovar este fato, pode-se partir para a análise de um exemplo prático em um data warehouse fictício, supondo que no modelo exemplo que está sendo analisando existam 10.000 itens produtos diferentes, 1.000 lojas e o período de tempo armazenado é de 2 anos, equivalente a 730 dias. Além disso, neste data warehouse exemplo, apenas aproximadamente 10 por cento dos produtos são vendidos em uma determinada loja em um determinado dia. Isto significa que a tabela de fatos ocupa apenas 10 por cento do que ocuparia caso todos os produtos fossem vendidos em todas as lojas, todos os dias. Entretanto, com a criação dos agregados esta taxa sobe dramaticamente. Supondo também que, no caso de produtos, 10.000 itens levem a 2.000 categorias de produto, as 1.000 lojas serão agrupadas em 100 cidades e o período de tempo total será sumariado em 24 períodos mensais. Além disso deve-se considerar a dispersão e o número de linhas da tabela de fatos de acordo com a tabela abaixo:

Tabela	Produto	Geografia	Tempo	Dispersão	Núm.Linhas
Base: item de produto por loja por dia	10.000	1.000	730	10%	730 milhões
Unidir: categoria por loja por dia	2.000	1.000	730	50%	730 milhões
Unidir: item por cidade por dia	10.000	100	730	50%	365 milhões
Unidir: item por loja por mês	10.000	1.000	24	50%	120 milhões
Bidir: categoria por cidade por dia	2.000	100	730	80%	116 milhões
Bidir: categoria por loja por mês	2.000	1.000	24	80%	38 milhões
Bidir: item por cidade por mês	10.000	100	24	80%	19 milhões
Tridir: categoria por cidade por mês	2.000	100	24	100%	4,8 milhões
Total aproximado:					2.122 milhões

Tabela 3.1: Quantidade de linhas após a criação dos agregados

Levando em conta o tamanho inicial da tabela de fatos base, que é de aproximadamente 730 milhões de linhas, e o número de linhas após a criação destes sete níveis de agregados, verifica-se que o aumento foi de 200%. O fator de dispersão pode ser difícil de prever e a escolha dos agregados deve ser feita com bastante cuidado. De acordo com [Tan97], a

forma mais eficiente para controlar a explosão de agregados, mas ainda assim beneficiar-se de seu valor, é garantir que, em média, cada agregado resuma no mínimo 20 e, de preferência, 20 ou mais itens de nível básico.

É importante lembrar que um determinado agregado pode servir para acelerar uma consulta que requeira um agregado de nível superior, ou seja, se existe um agregado de nível intermediário na hierarquia, este pode ser utilizado, não sendo necessário executar a consulta agregando o nível mais detalhado.

3.5.2 A implementação dos agregados

A implementação dos valores agregados no data warehouse conta com várias opções de desenho. É possível manter os fatos sumariados na mesma tabela fato original ou separados por tipo de agregado, o mesmo ocorre com as dimensões. A forma de se implementar os agregados vai depender do esquema implementado para o data warehouse e do número de linhas geradas pelos agregados.

Em um esquema do tipo *Star* onde, para cada categoria de informações, é gerada uma estrela composta por uma tabela fato e uma tabela para cada dimensão, os agregados podem residir na tabela de fatos original e seus identificadores já estão presentes nas tabelas de dimensão. Cada linha da tabela fato é identificada por uma chave composta, como já visto anteriormente, que identifica, nas dimensões, os elementos associados aos valores armazenados na fato. Antes da criação dos agregados, a tabela fato contém apenas valores associados ao nível mais detalhado de cada dimensão. Assim, com a inclusão dos valores agregados na tabela fato, estes deverão relacionar-se com os outros níveis da hierarquia das dimensões. Isto é possível, já que cada tabela dimensão, no esquema *Star*, possui uma linha descrevendo cada elemento da hierarquia, sendo que o campo Nível identifica esta estrutura hierárquica. É claro que, se os agregados são representados nas tabelas de dimensão e fatos originais, por meio desta estrutura de campos Nível, todas as consultas feitas a esse esquema devem restringir o campo Nível a um único valor, caso contrário, ocorrerá contagem dupla [Kim96], envolvendo valores relacionados

a níveis de agregação diferentes. A figura 3.27 mostra este relacionamento, representando uma dimensão Geografia e uma tabela fato que possui valores para os vários níveis da dimensão.

Como se pode verificar, na figura 3.27, a tabela fato continua sendo única, contendo as linhas de fatos para os níveis detalhados e agregados da dimensão. No exemplo, foram representados parcialmente dois agregados unidirecionais, sendo:

- Agregado: Item de produto por dia por Estado e
- Agregado: Item de produto por dia por Cidade.

Tabela Fato:

	Tempo key	Produto key	Geografia key	Valor	Quantidade
(1)	980715	101	4030	348,00	140
(2)	980716	101	4030	287,00	114
(3)	980716	101	4010	443,00	170
(4)	980716	102	4010	630,00	158
(5)	980717	101	2010	5.800,00	2320
(6)	980718	102	2010	7.540,00	1720
(7)	980718	101	2010	6.860,00	2740
(8)	980716	101	3010	2.840,00	1184
(9)	980716	103	3010	5.834,00	4059
	...	...	...	...	...

Tabela Dimensão Geografia:

Geografia key	Geografia Descrição	Região	Estado	Cidade	Loja	Nível
1010	Região Sul	1				Região
1020	Região Sudeste	2				Região
2010	SP	2	10			Estado
2020	RJ	2	20			Estado
3010	São Paulo	2	10	101		Cidade
3020	Campinas	2	10	102		Cidade
4010	Loja ABC	2	10	101	1001	Loja
4020	Loja DEF	2	10	102	1002	Loja
4030	Loja GHI	2	10	102	1003	Loja
...	...	...	...	...	...	...

Figura 3.27: Representação de uma única tabela fato contendo informações de nível detalhado e de agregados associados à dimensão Geografia

Portanto, na tabela fato da figura 3.27, as linhas (1), (2), (3) e (4) representam valores

associados aos níveis detalhados Item de produto, Dia e Loja, enquanto que as linhas (5), (6), (7), (8) e (9) representam os agregados.

A grande desvantagem desta implementação é a utilização do campo Nível que, como citado anteriormente, pode acarretar contagem dupla, se não for utilizado corretamente.

Uma alternativa seria manter os fatos agregados em tabelas distintas, ou seja, uma tabela de fatos para cada agregado. [Kim96] cita algumas das vantagens desta alternativa de implementação:

- Não ocorrerá contagem dupla em qualquer aplicação que utilize as tabelas;
- Os vários tipos de agregados podem ser criados, eliminados, carregados e indexados separadamente quando estão em tabelas diferentes. Como eles provavelmente serão introduzidos no banco de dados em momentos diferentes, o uso de tabelas separadas permite um gerenciamento incremental.

Em um esquema do tipo *Snowflake*, a implementação mais adequada é, obviamente, a utilização de tabelas fato separadas por agregado, já que as dimensões são também separadas. Não há motivos para se implementar o atributo Nível em um esquema do tipo *Snowflake*, unicamente com o objetivo de manter os agregados em apenas uma tabela fato.

3.5.3 A utilização dos agregados com um novo componente: o navegador de agregados

A melhor alternativa para a utilização dos agregados é, sem dúvida, o pré-armazenamento daqueles mais utilizados, sendo que algumas vezes a agregação é efetuada pela consulta durante sua execução, por não existir um agregado compatível.

O controle deste procedimento não deve ficar a cargo do usuário final, pois a criação ou eliminação de algum agregado levaria a uma série de procedimentos administrativos impraticáveis. Por esta razão, em um data warehouse projetado corretamente, os usuários finais e desenvolvedores de aplicações jamais verão quaisquer dessas tabelas de agrega-

dos [Kim96]. Torna-se necessária, portanto, a definição de um componente de software que escolha a tabela de agregados mais apropriada durante uma consulta, baseado, normalmente, em informações existentes no metadados do data warehouse. Assim, este componente, chamado por [Kim96] de navegador e por [IA99], de *Aggregate Aware*, para citar apenas dois exemplos, é capaz de criar as combinações de agregação que não estejam presentes no data warehouse, de forma dinâmica durante a execução da consulta. É importante que este componente seja capaz de criar estas agregações dinâmicas partindo do agregado predefinido mais adequado para a consulta e otimizando ao máximo o desempenho destas agregações.

A utilização deste navegador não elimina as preocupações relacionadas aos agregados. Como estas agregações dinâmicas são mais lentas que o acesso a um agregado armazenado, recomenda-se uma análise profunda para a definição destes agregados, seguida de acompanhamento ininterrupto de sua utilização.

3.6 O processo de carga na implementação estática

Uma das leis imutáveis do data warehousing, conforme [AV98], é sobre a complexidade do processo de carga, que nos diz que a parte mais complexa na construção de um data warehouse é a carga dos dados. Vários aspectos contribuem para aumentar a complexidade desta tarefa: baixa qualidade e falta de dados adequados, péssima documentação dos sistemas, múltiplas, redundantes e conflitantes fontes de dados. Muitas características complexas dos dados ficam "escondidas", vindo à tona somente no momento do processo de carga do data warehouse. A utilização de ferramentas que automatizam este processo pode facilitar o trabalho, porém a complexidade semântica que envolve o processo de transformação dos dados não pode ser minimizada. O processo de carga é, na grande maioria das vezes, redundante e desnecessariamente complicado, tornando-se um ponto crítico para a introdução de erros.

A complexidade desta etapa surge devido à diversidade e abrangência das tarefas envolvidas, recuperando, convertendo e migrando os dados a partir das diversas fontes,

executando sua transformação de modo a garantir que a base de dados resultante compreenda sempre informações precisas, integradas, válidas e disponíveis em tempo hábil.

O processo de extração conta com várias dificuldades operacionais como, por exemplo, encontrar uma janela de tempo adequada para a execução do conjunto de programas necessários para o processo, normalmente em *batch*, principalmente se a atualização do data warehouse for diária. Outro problema a ser enfrentado refere-se ao ponto de sincronismo entre os vários sistemas transacionais, que são atualizados em diferentes intervalos de tempo e que servirão de entrada para este processo de extração.

Uma vez que a periodicidade do processo de carga esteja definida, é necessário escolher um método para **identificar e capturar as informações que foram atualizadas** nas bases de origem desde a execução da última carga. Existem algumas opções que podem ser empregadas para esta tarefa e, de acordo com [Kel94], as mais importantes são:

- **Utilização de Logs**: a maioria dos gerenciadores de bancos de dados mantém *logs* ou *journals* que podem ser utilizados para identificar as alterações ocorridas nas bases de dados. Estas alterações, uma vez identificadas, devem ser escritas em um arquivo diferente que será utilizado para migrar as alterações para o ambiente de data warehouse. Esta alternativa produz bons resultados se as atualizações a serem capturadas são obtidas de forma simples e direta. Porém, se os dados necessários para a atualização do data warehouse estiverem em um número considerável de arquivos ou bases de dados separados, dificultando o sincronismo dos *logs*, este processo começa a aumentar em complexidade. Adicionalmente, pode-se citar os problemas relativos às alterações necessárias no programas de carga devido a novas versões dos gerenciadores de bancos de dados que podem levar a modificações na estrutura dos *logs*.

- **Comparação das informações**: apesar de conceitualmente simples, é uma opção bastante trabalhosa para executar. Esta opção consiste em desenvolver um programa que leia os dados nas bases fonte e no data warehouse, identificando as alterações ocorridas desde a última execução da carga. Este método é caro, no que

se refere aos recursos de máquina necessários para comparar os dados e pode ser demorado dependendo do tamanho das bases pesquisadas e dos recursos disponíveis para seu processamento.

- **Reengenharia dos sistemas aplicativos transacionais**: esta opção não se mostra muito atrativa, já que requer que as aplicações em produção sejam alteradas com o objetivo de armazenar, em arquivos destinados para este fim, as modificações ocorridas nos dados fonte. Ocorre, porém, que a maioria das aplicações não pode sofrer facilmente um processo de reengenharia sendo que esta opção, em muitas situações, só poderia ser utilizada nas novas aplicações que estivessem sendo desenvolvidas. Neste caso, as necessidades do data warehouse passariam a fazer parte da especificação da aplicação.

Após sua captura, os dados devem passar por uma etapa de **transformação**, já que, simplesmente obter os dados do ambiente fonte e passá-los para o data warehouse não é suficiente. É necessário, com o objetivo de otimizar o potencial do data warehouse, transformar as entidades obtidas em novas composições de entidades que são requeridas para processo de geração de informação relevante para o usuário.

Além da transformação, os dados passam por um processo de **enriquecimento**, que é, normalmente, um produto da integração de dados e ocorre quando um atributo adicional é acrescentado a uma entidade. Se um dado externo está sendo acrescentado ao data warehouse, uma entidade *Cliente*, por exemplo, pode ser "enriquecida"com este novo atributo que foi selecionado de fontes econômicas, demográficas, financeiras, etc. Estas fontes podem ser valiosas para o "enriquecimento"das informações, porém mais valiosas ainda, são as próprias aplicações do data warehouse cujos atributos podem ser derivados dos padrões de comportamento analisados.

Uma outra etapa necessária para a migração dos dados envolve um **mecanismo de transporte** partindo do ambiente operacional para o data warehouse. Estes dois ambientes podem estar localizados em distintas plataformas, além da possibilidade de estarem fisicamente remotas levando a uma maior dificuldade na transferência dos dados. Por ser

inconcebível um meio de transporte de dados que não seja eletrônico, a implementação do mecanismo de transporte pode envolver uma avaliação de todas as opções disponíveis, já que diferentes fabricantes podem oferecer diferentes opções proprietárias com custos, capacidades e limitações bastante distintas. Ao considerar a velocidade da transferência, deve-se levar em conta cada aspecto envolvido no processo, seja na plataforma de origem ou na plataforma destino. Esta análise é particularmente importante para determinar, no contexto considerado, o tempo disponível para a execução desta etapa nas plataformas envolvidas.

Na seqüência do processo de carga um fator muito importante está relacionado a garantir a **integridade dos dados** que estão sendo adicionados ao data warehouse. Este controle de integridade deve ser implementado de forma a garantir dois aspectos. O primeiro deles é a garantia de que os dados que foram extraídos dos sistemas de origem são exatamente os mesmos que estão sendo carregados no data warehouse. O segundo aspecto deve garantir que os dados que estão sendo carregados estão consistentes com os dados que foram pedidos para as bases de origem em um determinado instante no tempo.

A **reformatação dos dados** pode também ser uma etapa necessária neste processo, uma vez que os ambientes de origem e o data warehouse podem estar em ambientes computacionais heterogêneos, que muitas vezes causam problemas relacionados ao formato dos dados. O problema mais comum refere-se às diferenças entre os formatos de arquivos em EBCDIC, encontrados nos ambientes IBM e os formatos ASCII utilizados pela maioria das outras plataformas.

Como último passo do processo, encontramos a etapa de **carga dos dados** que depende, obviamente, do volume de dados envolvido no processo. Pode ser que o tempo necessário para a carga dos dados adicionado ao tempo requerido para transferi-los, cause um impacto negativo na disponibilidade do data warehouse para os usuários. Portanto, devem ser considerados todos os aspectos que possam dificultar este processo. Um destes aspectos é a quantidade de índices criados nas tabelas do data warehouse. Uma grande quantidade de índices adequados vai agilizar a execução das consultas, objetivo primário de um data warehouse, porém este alto nível de indexação pode diminuir sensivelmente

a velocidade do processo de carga.

É fácil perceber, após uma análise das etapas necessárias ao processo de carga, a alta complexidade desta tarefa que, se não for definida com bastante cuidado, pode levar à introdução de erros, comprometendo todo o processo de tomada de decisão executado pelos usuários. Além disso, qualquer alteração na arquitetura dos sistemas transacionais e nas bases de origem deve ser detalhadamente verificada, já que pode causar alterações em várias etapas do processo de carga.

3.7 Conclusão

Com a implementação desta abordagem estática, o sincronismo entre o data warehouse e os sistemas transacionais fica dependente da periodicidade do processo de carga. Normalmente, a periodicidade da sincronização implementada é, no mínimo, diária. Para um grande conjunto de aplicações em data warehouse, esta periodicidade é satisfatória. Porém, para determinadas aplicações, que requerem a propagação dos dados da base transacional para o data warehouse de forma imediata, é necessária uma forma dinâmica para implementar este sincronismo. Outro aspecto a considerar é que, durante o processo de carga descrito na seção anterior, o data warehouse fica indisponível para consultas. Se considerarmos as características de uma economia globalizada, onde empresas localizadas em diferentes países possam estar utilizando o mesmo data warehouse para os processos de tomada de decisão, este período de indisponibilidade é totalmente inaceitável.

Face aos aspectos expostos, é necessário buscar uma implementação mais dinâmica do data warehouse, que minimize os problemas apresentados acima. Esta implementação é descrita no próximo capítulo deste trabalho.

Capítulo 4

A abordagem dinâmica utilizando visões materializadas

Partindo das definições já citadas anteriormente, pode-se dizer que o data warehouse é um conjunto de dados integrados, a partir de diversas fontes heterogêneas, em uma grande base de dados [GM96]. Este data warehouse agrega os dados detalhados e armazenados ao longo de diversas dimensões do negócio e mantém estes dados sumariados para o processamento das consultas efetuadas através de ferramentas de suporte à decisão e de ferramentas OLAP.

Uma outra forma de implementar um data warehouse é defini-lo como um conjunto de *visões materializadas* que integram os dados a partir de múltiplas fontes heterogêneas, e eventualmente distribuídas, de informação [DEB95].

Uma visão é uma relação derivada, definida em termos de relações base, que é computada todas as vezes que uma referência a ela é feita. Uma visão é dita materializada quando ela é realmente armazenada na base de dados em vez de ser computada a partir das relações base em resposta a consultas [QGMW97]. Uma visão materializada pode ser vista como um *cache*, uma cópia dos dados que pode ser acessada rapidamente.

Além de definir o próprio data warehouse como um conjunto de visões materializadas baseadas nos dados dos ambientes operacionais [QGMW97, Gup97], estas visões também

podem ser utilizadas com o objetivo de otimizar consultas complexas [Qua97], sendo que, para isto, deve-se definir um conjunto compartilhado de visões, criteriosamente escolhidas a partir de uma análise das consultas mais freqüentes executadas no data warehouse. Estas visões, ao serem materializadas, agilizam o acesso aos dados necessários para a realização das consultas no data warehouse [YKL96], aspecto crítico e diferencial nas aplicações onde a quantidade de consultas é alta e as visões são complexas ou são definidas sobre dados em bases remotas.

Estes dois temas têm apresentado grande relevância, já que a utilização dos data warehouses vem aumentando significativamente e inclui aspectos como otimização das consultas, replicação dos dados e construção e otimização das ferramentas de suporte à decisão e OLAP. Vários trabalhos vêm sendo realizados envolvendo a utilização das visões materializadas em sistemas de data warehouse, com o objetivo de conferir a estas enormes bases de dados, melhor integração das informações, facilidades no processo de carga e, obviamente, otimização das consultas efetuadas.

Como a tecnologia de visões materializadas pode apoiar todo o processo de implantação e utilização dos data warehouses? Quais as vantagens em se utilizar este tipo de visões? Quais os limites encontrados nesta abordagem? Estas são algumas das questões respondidas ao longo das próximas seções.

A utilização das visões materializadas pode conferir ao data warehouse um dinamismo não encontrado na abordagem estática discutida anteriormente. Com o uso das visões, o data warehouse pode manter-se atualizado e sincronizado com as bases de dados transacionais originais, em intervalos de tempo bem menores, já que não dependerá do processo de carga periódico para isto. Tipicamente, uma visão é mantida imediatamente, como parte da transação que atualiza as tabelas bases, quando da inserção ou eliminação de alguma linha destas tabelas. Esta forma de atualizar as visões causa, porém, uma sobrecarga ao processo de atualização das tabelas base. Uma outra opção é protelar esta manutenção por um certo período de tempo e ativá-la em intervalos predefinidos ou sob demanda. É importante, no processo de atualização das visões, levar em conta dois fatores básicos: minimizar o tempo em que a visão fica "bloqueada"para acesso (durante sua

manutenção) e minimizar a sobrecarga imposta às transações de atualização das tabelas base considerando que este processo passa por, basicamente, duas etapas. A primeira etapa consiste da definição das alterações que devem ser aplicadas nas visões e a segunda, na aplicação destas atualizações.

Fica claro que desta abordagem mais dinâmica do data warehouse, que utiliza visões materializadas, surgem vários problemas. Um deles é a decisão sobre a selecão das visões mais adequadas para a implantação do data warehouse. É necessário analisar os custos de manutenção em função dos benefícios na utilização de cada visão. Outro aspecto que deve ser analisado se refere à capacidade de automanutenção das visões, minimizando ao máximo o acesso às bases remotas para manter o data warehouse sincronizado com as fontes de informação originais. Além disso, as técnicas de manutenção das visões dependem da definição de quando serão aplicadas as alterações. Todos estes problemas são também discutidos nas próximas seções, incluindo as soluções desenvolvidas para viabilizar a utilização desta abordagem nos data warehouses.

4.1 O uso das visões materializadas no data warehouse

Como já citado, uma visão materializada é uma consulta cujo resultado já está computado e armazenado na base de dados [Qua97]. As consultas que puderem utilizar as visões já armazenadas podem ser executadas de forma muito mais rápida, sendo que, para consultas complexas envolvendo grandes volumes de dados, esta alternativa favorece dramaticamente os resultados: de horas ou dias para segundos ou minutos. De fato, as visões materializadas são vistas como uma das principais opções para o controle do desempenho de um data warehouse [Kim96].

A desvantagem das visões materializadas é que as alterações feitas aos dados base, a partir dos quais uma visão materializada é definida, torna a visão desatualizada. Para que a visão possa estar novamente sincronizada aos dados base, será necessário recriar a

visão a partir dos dados de origem ou então, atualizá-la incrementalmente [Qua97].

Outro aspecto importante é que estas atualizações podem ser executadas imediatamente, tão logo a alteração seja recebida, ou podem ser proteladas, de tal forma que todas as alterações ocorridas nos dados base serão reunidas e aplicadas em processos batch.

O uso de visões materializadas nos data warehouses, conforme descrito nos parágrafos acima, requer, portanto, a análise de alguns tópicos relacionados, como:

1. A seleção das visões mais adequadas levando-se em conta uma análise dos custos de manutenção e os benefícios de cada visão e os algoritmos disponíveis;

2. Aspectos da materialização e utilização das visões para responder às consultas;

3. Os mecanismos que permitem a propagação correta quando da atualização das fontes de dados base, para vários tipos de visões, preferencialmente de forma online;

4. A manutenção das visões de forma dinâmica e incremental;

5. A criação de algoritmos que permitam a manutenção das visões de maneira autônoma, utilizando o conceito de *self maintainable views*.

A seguir são descritas as visões materializadas e alguns aspectos referentes à sua definição, seleção, utilização e manutenção.

4.2 A seleção das visões a serem materializadas

[GM95] define uma visão como uma relação derivada, em termos das relações base armazenadas. Uma visão é, portanto, uma função que parte de um conjunto de tabelas base para uma tabela derivada, sendo que esta função é recalculada todas as vezes que é referenciada.

Uma visão pode ser materializada, através do armazenamento das tuplas da visão na base de dados. Com esta materialização, é possível criar índices que tornarão o acesso a

estas visões materializadas muito mais rápido que o recálculo, que é executado todas as vezes que as visões são referenciadas.

Conforme descrito em [YKL96], um data warehouse, devido aos diferentes tipos de análises, pode conter múltiplas visões. Quando estas visões estão relacionadas umas com as outras, isto é, quando estão definidas sobre partes sobrepostas dos dados base, então pode ser mais eficiente materializar apenas certas visões compartilhadas, ou porções dos dados base, em vez de materializar todas as visões. Assim, surge o primeiro aspecto a ser analisado, que se refere à escolha das visões que serão materializadas. Esta escolha se baseia na determinação de um conjunto de visões, de uso compartilhado no data warehouse, de modo a combinar bom desempenho com baixo custo de manutenção. O objetivo é selecionar um conjunto apropriado de visões que minimize o tempo total de reposta das consultas e o custo de manutenção das visões selecionadas, dado um certo limite de recursos, como por exemplo, tempo de materialização, espaço para armazenamento etc.

Vários trabalhos tratam deste tema, como por exemplo, [HRU96] que apresenta e analisa algoritmos para a seleção das visões em um caso especial de "cubos de dados". [GHRU96] estende os resultados para a seleção de visões e índices em cubos de dados, porém ambos ignoram os custos de manutenção das visões. Uma estrutura teórica é definida em [Gup97] para o problema geral de seleção de visões, apresentando uma heurística para alguns casos especiais importantes de problemas que ocorrem na prática. [YKL96] define aspectos de uma metodologia para desenho das visões materializadas, por exemplo, como selecionar um conjunto de resultados intermediários das consultas a serem materializados de forma que o custo total seja mínimo. Apresenta também um algoritmo para escolha deste conjunto, levando em conta as freqüências das consultas e as freqüências das atualizações nos dados base. Todo o processo de escolha das visões está baseado em um plano de processamento das visões envolvidas (MVPP - Multiple View Processing Plan), gerado por um algoritmo, que parte dos planos individuais de cada consulta envolvida. O trabalho apresentado por [YKL96] analisa o desenho da visão materializada em termos de desempenho, levando em conta também os recursos utilizados na manutenção da visão. A discussão é apresentada em termos do modelo relacional com

operações *select, project* e *join*, sendo que a abordagem utilizada pode ser estendida para operações mais complexas, como consultas com agregações e consultas recursivas.

4.2.1 Uma descrição resumida do trabalho de [YKL96]

Nesta seção são apresentados os conceitos do trabalho realizado por [YKL96]. A apresentação destes conceitos estará baseada em uma base de dados exemplo, que contém as seguintes relações:

Product[7] (Pid, name, Did)

Division (Did, name, city)

Order (Pid, Cid, quantity, date)

Customer (Cid, name, city)

Part (Tid, name, Pid, supplier)

Para simplificar os diagramas, Pd, Div, Ord, Cust e Pt representam, respectivamente, as relações acima. Além disso, assume-se que estas relações encontram-se todas no mesmo local e, portanto, são desconsiderados os custos de comunicação de dados para os cálculos que se seguirão.

Supondo as seguintes consultas, freqüentemente utilizadas no acesso ao data warehouse:

Consulta 1:

```
Select Pd.name
From Pd, Div
Where Div.city = "LA" and Pd.Did = Div.Did;
```

Consulta 2:

```
Select Pt.name
From Pd, Pt, Div
Where Div.city = "LA" and Pd.Did = Div.Did
                      and Pt.Pid = Pd.Pid;
```

[7] Os nomes dos atributos e das relações envolvidas no exemplo serão mantidos no idioma original.

Para cada uma das consultas é gerado um grafo de processamento, que representa seu plano de acesso individual, conforme a figura 4.1.

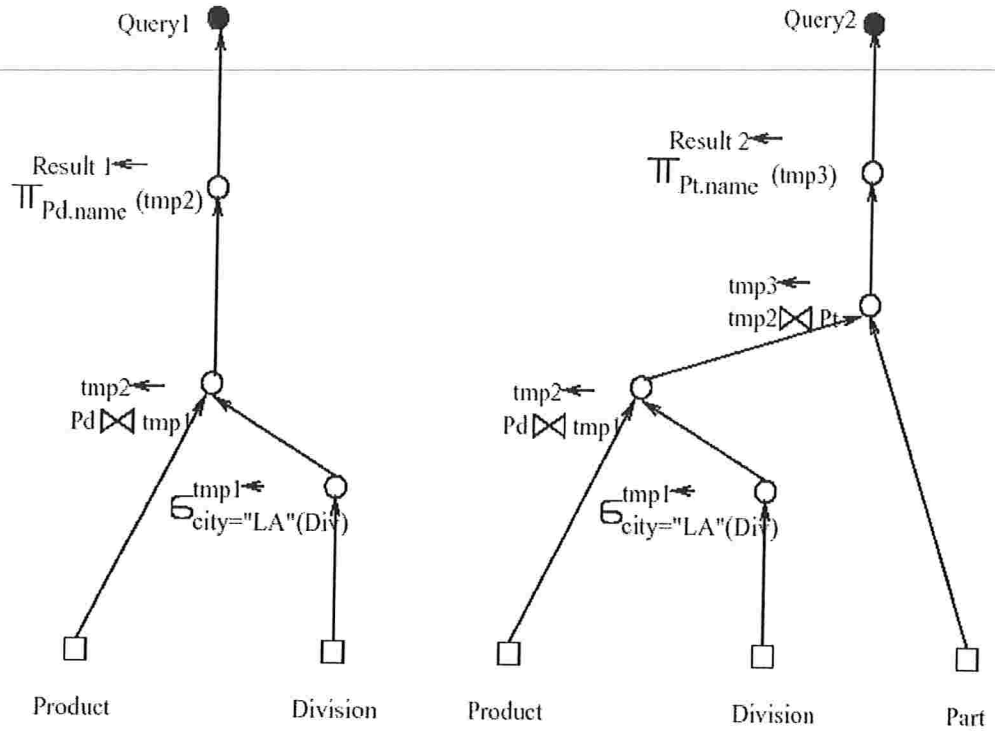


Figura 4.1: Plano de acesso individual para as consultas 1 e 2 (query1 e query2)

Uma das alternativas para obter um rápido tempo de respostas para as consultas acima, seria materializar alguns dos nós intermediários de cada plano de acesso individual, sendo que, para o cálculo do custo total, os custos de manutenção das visões também deveriam ser levados em conta. O nó intermediário tmp2 da query1 é equivalente ao da query2, na figura 4.1, e são chamados de sub-expressões comuns. Os dois planos individuais podem ser combinados de modo a formar um único plano conforme mostra a figura 4.2.

Fica claro que, se o nó identificado como tmp1 for materializado, pode ser utilizado pelas duas consultas, em vez do acesso às relações base Product, Division e Part, diminuindo assim seus custos de execução. Além disso, o custo de manutenção de apenas um nó tmp1 será menor que o de dois tmp1s. Desta forma, haverá ganhos em termos de custo total do acesso global e da manutenção da visão.

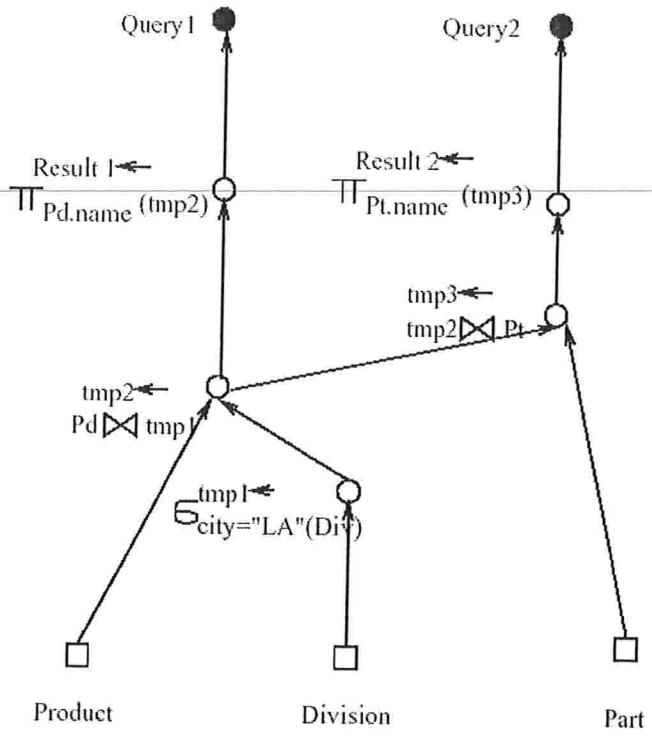


Figura 4.2: Plano de acesso combinado para query1 e query2

Agora, vamos supor duas outras consultas freqüentes ao data warehouse:

Consulta 3:

Select Cust.name, Pd.name, quantity

From Pd, Div, Ord, Cust

Where Div.city = "LA" and Pd.Did = Div.Did

and Pd.Pid = Ord.Pid and Ord.Cid = Cust.Cid

and date > 7/1/96;

Consulta 4:

Select Cust.city, date

From Ord, Cust

Where quantity > 100 and Ord.Cid = Cust.Cid;

A figura 4.3 representa um plano de acesso global para as quatro consultas, de forma que os planos de acesso locais ou individuais foram combinados com base nas operações

compartilhadas dos dados comuns. Este plano global é chamado de **Multiple View Processing Plan (MVPP)**. Após a criação deste plano global, pode-se decidir quais são os nós a serem materializados, de forma que o custo da consulta de o da manutenção da visão sejam mínimos.

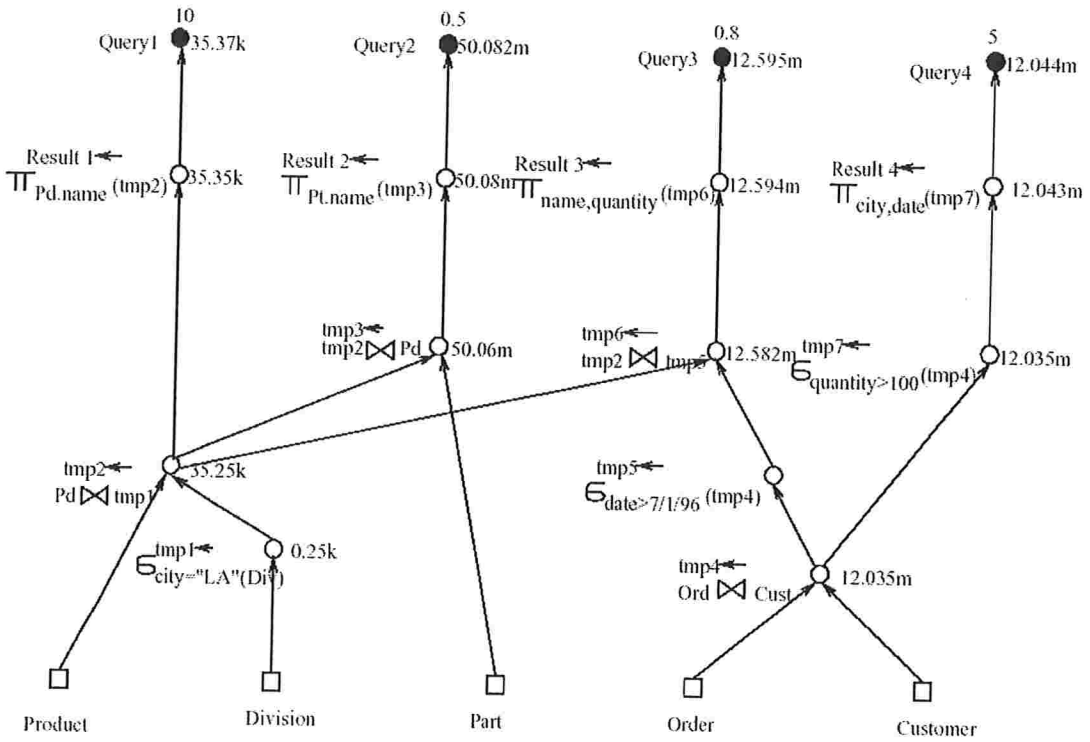


Figura 4.3: Um MVPP para o exemplo

Obviamente, no grafo, existem várias opções para a escolha do conjunto de visões materializadas, a saber: (1) materializar todas as consultas; (2) materializar alguns dos nós intermediários, por exemplo, tmp1, tmp2, tmp4 etc.; (3) manter virtuais todos os nós que não sejam folhas. Para a decisão do melhor conjunto, o custo de cada alternativa deve ser calculado em termos do processamento da consulta e da manutenção da visão.

O trabalho apresentado em [YKL96] define dois algoritmos para implementar uma solução para o problema:

1. **Algoritmo para a definição de múltiplos MVPP**:

Normalmente para uma consulta existem vários planos de processamento, sendo que, entre eles existe um plano considerado ótimo. Da mesma forma, pode-se ter múltiplos MVPPs baseados em diferentes combinações destes planos individuais. Para reduzir o espaço da pesquisa, o trabalho parte de planos individuais ótimos e os ordena, baseado nas freqüências das consultas e seus custos. Uma vez que a ordem está definida, o algoritmo trabalha com os vários planos individuais, combinando suas sub-expressões comuns chegando a um conjunto de k planos globais ou k MVPPs. Cada MVPP gerado tem um custo total que será analisado pelo segundo algoritmo descrito em seguida.

2. **Algoritmo para selecionar os nós intermediários a serem materializados:** Dado um MVPP, o objetivo é definir um conjunto de visões materializadas tal que o custo total do processamento da consulta e da manutenção da visão seja mínimo, tentando comparar os custos de cada combinação de nós possível. O trabalho apresentado propõe um algoritmo que compara o custo total de cada MVPP gerado, selecionando o de menor custo.

4.3 Alguns aspectos da manutenção das visões

Por se comportar como um *cache*, uma visão materializada oferece acesso rápido aos dados, sendo que esta velocidade pode ser crítica em aplicações onde a quantidade de consultas é alta e a complexidade das visões leva à impossibilidade de recálculo da visão em todo acesso.

Assim, como um *cache* que se torna obsoleto quando seus dados base sofrem atualização, as visões materializadas também ficam desatualizadas quando as bases de dados de origem são modificadas. Neste caso, é necessária a execução de um processo de manutenção sendo que, na maioria dos casos, é desperdício de recursos proceder com seu recálculo a partir do zero [GM95]. Isto porque apenas uma parte da visão muda em resposta às alterações das bases originais e, portanto, é possível alterar o conteúdo da visão materializada, de forma incremental. Obviamente, isto é apenas heurística. Se ocorrer,

por exemplo, a eliminação total da relação base, será mais barato recalcular a visão, que depende desta relação eliminada, do que apenas aplicar as alterações. Vários trabalhos descrevem algoritmos que permitem a manutenção incremental das visões materializadas, com diferentes domínios de aplicabilidade, como [SI84, CW91, HD92, GMS93].

[GM95] propõe uma classificação do problema de manutenção das visões, através de quatro dimensões, ao longo das quais este problema pode ser analisado, a saber:

- **Dimensão da Informação**: Refere-se à quantidade de informação disponível para a manutenção da visão. Na análise sob esta dimensão, pergunta-se se o acesso é possível a toda ou apenas parte das relações base e se existe acesso à visão materializada, se são conhecidas regras de integridade e chaves. A quantidade de informação utilizada é ortogonal à "incrementabilidade"da manutenção da visão. "Incrementabilidade"se refere à capacidade de se computar ou calcular apenas aquela parte da visão que foi alterada. Nesta dimensão, portanto, os dados utilizados para computar as alterações na visão devem ser analisados.

- **Dimensão da Modificação**: Nesta dimensão são analisadas as modificações que o algoritmo de manutenção da visão pode tratar. Questiona-se, então, se inserções ou eliminações de tuplas nas tabelas base são tratadas pelo algoritmo, se as atualizações nas tuplas são tratadas diretamente ou se são modeladas como eliminações seguidas por inserções e aspectos relacionados às alterações na definição da visão.

- **Dimensão da Linguagem**: Esta dimensão está relacionada a questões sobre a forma com que a visão está expressa, ou seja, foi definida como uma consulta do tipo *select-project-join* (também conhecida como visão SPJ ou como consulta conjuntiva) ou baseada em algum outro subconjunto da álgebra relacional, se utiliza a SQL ou um subconjunto da SQL, se a visão possui duplicidades e se utiliza agregação e recursão.

- **Dimensão das Instâncias**: Esta dimensão divide-se em dois tipos: informações relativas às instâncias da base de dados e às instâncias da modificação. É importante

saber se o algoritmo de manutenção da visão funciona para todas as instâncias da base de dados ou apenas para algumas e se funciona para todas as instâncias da modificação ou apenas para algumas.

Para esclarecer a utilização desta classificação, [GM95] propõe alguns exemplos, que são transcritos abaixo:

Exemplo 1: Dimensões da Informação e Modificação

Considerando a relação:

$$\mathsf{part(part\_no,\ part\_cost,\ contract)}$$

que lista o custo de uma peça, negociado em um contrato. É importante notar que uma peça pode ter preços diferentes por contrato. Considerando também a visão expensive_parts definida como:

$$\mathsf{expensive\_parts(part\_no)} = \Pi_{part\_no}\ \sigma_{part\_cost<1000}\ \mathsf{(part)}$$

Esta visão contém números de peça **distintos** para as peças que custam mais que $1000 presentes em, ao menos, um contrato (a operação de projeção descarta as duplicidades). Considerando a manutenção da visão quando uma tupla for inserida na relação part, se a tupla inserida tiver part_cost $\leq$ 1000, então a visão não sofrerá alterações. Entretanto, supondo que part($p1$, 5000, $c1$) foi inserida, tendo custo>1000, algoritmos distintos podem ser definidos, dependendo da informação disponível para determinar se $p1$ deve ser ou não inserido na visão:

- Apenas a visão materializada está disponível: utiliza-se a antiga visão materializada para determinar se a peça de número part_no já está presente na visão. Se já estiver, não há alteração na materialização, caso contrário, deve-se inserir a peça $p1$ na materialização;

- Apenas a relação base part está disponível: utiliza-se a relação part para verificar se alguma tupla existente na relação tem o mesmo número part_no, porém com custo igual ou maior. Se tal tupla já existir, então a nova tupla inserida não contribui para a visão;

- É conhecido que part_no é a chave: infere-se que a peça de part_no não está na visão e, portanto, deve ser inserida.

Um outro problema de manutenção das visões está relacionado a eliminações utilizando apenas a visão materializada. Considere-se a eliminação da tupla part($p1$, 2000, $c12$). Está claro que a peça $p1$ tem que estar na materialização, porém não se pode eliminar $p1$ da visão, já que alguma outra tupla, como part($p1$, 3000, $c13$), pode ter contribuído em levar a peça $p1$ para a visão. A existência (ou inexistência) desta tupla não pode ser provada utilizando-se apenas a visão. Desta forma, não existe um algoritmo que possa resolver este problema de manutenção para eliminações, utilizando apenas a visão materializada. É importante notar que, se a relação part estivesse disponível, ou se fosse conhecida a restrição da chave ou ainda, se a quantidade de tuplas derivadas estivessem disponíveis, então a visão poderia ser mantida.

Com respeito à dimensão da informação, deve-se notar que, tanto a definição da visão quanto a alteração real, estão sempre disponíveis para o processo de manutenção e, em relação à dimensão da modificação, as atualizações são tratadas, tipicamente, como uma eliminação seguida por uma inserção.

Exemplo 2: Dimensões da Linguagem e das Instâncias

O exemplo anterior considerou uma visão, cuja definição consiste de operações de seleção e projeção. Para este segundo exemplo, a linguagem de definição da visão será estendida para implementar também a operação *join* , sendo que a visão supp_parts é definida como um *equijoin* entre as relações supp(supp_no, part_no, price) e part, sendo que $\times_{\text{part\_no}}$ representa um *equijoin* no atributo part_no:

$$\text{supp\_parts(part\_no)} = \Pi\ \text{part\_no(supp} \times_{part\_no} \text{part)}$$

A visão contém os números distintos das peças que são fornecidas por, no mínimo, um fornecedor (a operação de projeção descarta as duplicidades). Para o processo de manutenção deve-se considerar apenas a utilização do conteúdo antigo da visão supp_parts. A inserção da peça part($p1$, 5000, $c15$), será analisada neste exemplo. Se supp_parts já

contiver a peça com part_no $p1$, então esta inserção não afetará a visão existente. Entretanto, se supp_parts não contiver $p1$, então o efeito da inserção não pode ser determinado utilizando-se apenas a visão.

É importante lembrar que a visão expensive_parts, do exemplo anterior, sofreu o processo de manutenção em decorrência das inserções em part, utilizando apenas a visão. Diferentemente, a operação *join* torna impossível a manutenção de supp_parts, como consequência das inserções em part, utilizando-se apenas a visão. Com isso, a visão supp_parts pode ser mantida se já contiver a peça com part_no $p1$, caso contrário, não. Portanto, a capacidade de sofrer manutenção de uma visão depende também de instâncias particulares da base de dados e da modificação.

4.4 A manutenção incremental das visões

Como uma visão materializada é uma relação derivada e definida a partir de relações base, operações de inserção, eliminação e atualização nestas bases de origem fazem com que a visão se torne desatualizada. Recriar a visão completa, principalmente em data warehouses onde as relações existentes contêm um número muito grande de elementos, pode ser um processo caro, demorado e ineficiente, dado que, na maior parte dos casos, a janela de tempo disponível para a manutenção do data warehouse é limitada. Uma alternativa bastante utilizada é aplicar na visão, apenas as alterações. Este tipo de manutenção é chamado de manutenção incremental das visões.

A maioria das técnicas de manutenção define as visões como uma fórmula matemática e obtém uma expressão que representa o processo de manutenção. O exemplo abaixo ilustra estas técnicas, conforme [GM95]:

Exemplo 3:

Seja a relação base link(S, D), de tal forma que link(a, b) é verdadeiro, se existe uma ligação que parte do nó a e chega no nó b. A visão hop é definida de tal forma que hop(c, d) é verdadeiro, se c está conectado a d utilizando duas ligações, através de um nó intermediário:

$$\mathcal{D}\text{: } \mathsf{hop}(X,Y) = \Pi_{X,Y}(\mathsf{link}(X,\, V) \times_{V=W} \mathsf{link}(W,Y))$$

Seja um conjunto de tuplas $\Delta(\mathsf{link})$, inserido na relação link. As inserções $\Delta(\mathsf{hop})$ que devem ser efetuadas na visão hop, de forma correspondente, podem ser computadas matematicamente, pela definição da diferenciação $\mathcal{D}$, para se obter a seguinte expressão:

$$\Delta(\mathsf{hop}) = \Pi_{X,Y}((\Delta(\mathsf{link})(X,\, V) \times_{V=W} \mathsf{link}(W,\, Y)) \cup (\mathsf{link}(X,\, V) \times_{V=W} \Delta(\mathsf{link})(W,\, Y)) \cup (\Delta(\mathsf{link})(X,\, V) \times_{V=W} \Delta(\mathsf{link})(W,\, Y)))$$

No exemplo 3 citado acima, se as tuplas forem eliminadas da relação link, a mesma expressão pode computar as eliminações da visão hop. Se as tuplas forem inseridas e eliminadas da relação link, então $\Delta(\mathsf{hop})$ pode ser obtido através do processamento do conjunto de eliminações $\Delta^{-}(\mathsf{hop})$ e do conjunto de inserções $\Delta^{+}(\mathsf{hop})$, separadamente, conforme [QW91, HD92]. Em [GMS93], para casos especiais, inserções e eliminações podem ser tratados em um único passo.

O trabalho desenvolvido por [GMS93] apresenta dois algoritmos para manutenção incremental que aplica nas visões materializadas as alterações (inserções, eliminações e atualizações) ocorridas nas relações base, sendo que as visões podem utilizar uniões, negações, agregações (como SUM, MIN) e recursões. Os dois algoritmos utilizam a definição da visão para produzir regras que implementarão as alterações nas visões utilizando, para isto, as alterações ocorridas nas relações base e as visões materializadas na situação anterior às modificações.

O primeiro algoritmo descrito em [GMS93] é um algoritmo de *counting*. Este algoritmo pode ser utilizado em visões com ou sem linhas duplicadas. Sua idéia básica é manter um contador do número de derivações para cada tupla da visão. Para descrevê-lo de forma mais detalhada, dada sua importância, será utilizado como base o exemplo 3 acima, com a relação base link e a visão hop.

Exemplo 4: A definição da visão hop, em SQL, é:

CREATE VIEW hop(S,D) as

(select distinct l1.S, l2.D from link l1, link l2 where l1.D = l2.S)

Seja a relação link = $\{(a,b), (b,c), (b,e), (a,d), (d,c)\}$ e, como resultado, a visão hop $\{(a,c), (a,e)\}$. A tupla hop (a,e) tem ocorrência única, ou seja, é derivada uma única vez de link, sendo que, a tupla hop (a,c) possui duas derivações. Se a visão tivesse duplicidade, ou seja, não tivesse sido definida com o operador *distinct*, então hop (a,e) teria *count* igual a 1 e hop (a,c) teria *count* igual a 2. O algoritmo de *counting* simula a duplicidade na visão e armazena estes contadores.

Supondo que a tupla link (a,b) seja eliminada, hop seria recomputada como $\{(a,c)\}$. O algoritmo de *counting* infere que uma derivação de cada uma das tuplas hop (a,c) e hop (a,e) deve ser eliminada. Baseado nos contadores armazenados, infere também que hop (a,c) possui ainda uma derivação e portanto, apenas hop (a,e), que não possui mais nenhuma derivação, deve ser eliminada.

O segundo algoritmo descrito em [GMS93] não pode ser utilizado em visões com duplicidade de tuplas, ou seja, em visões definidas sem o operador *distinct*. Este algoritmo, de nome **DRed** (Deletion and Rederivation) calcula as alterações nas visões em três passos. No primeiro passo, o algoritmo faz uma super estimativa das tuplas derivadas de devem ser eliminadas. Uma tupla t está nesta super estimativa se as alterações feitas nas relações base invalidam qualquer ocorrência de t. Em um segundo passo, esta superestimativa sofre a remoção daquelas tuplas que tenham uma ocorrência ou derivação alternativa na nova base de dados. Finalmente, as novas tuplas que devem ser inseridas são computadas utilizando-se a visão materializada parcialmente atualizada e as inserções feitas nas relações base. Para ilustrar este algoritmo, vamos considerar o próximo exemplo.

Exemplo 5:

Seja a visão hop, definida no exemplo 4 e a eliminação da tupla link (a,b). O algoritmo **DRed** primeiramente elimina as tuplas hop (a,c) e hop (a,e), já que ambas dependem da tupla eliminada. O algoritmo procura, então, pelas derivações alternativas de cada uma das tuplas eliminadas. Assim, no segundo passo, a tupla hop (a,c) é novamente derivada e inserida na visão materializada. O terceiro passo do algoritmo é vazio, para este exemplo, já que não existem tuplas a serem inseridas na tabela link.

O processo de manutenção incremental das visões depende da quantidade de in-

formações disponíveis. Conforme descrito em [GM95], vários trabalhos foram desenvolvidos apresentando algoritmos para a manutenção das visões em função da quantidade de informações disponíveis e do tipo de visão.

Foram analisados os casos onde toda a informação está disponível para o processo de manutenção, ou seja, todas as relações base e as visões materializadas estão disponíveis durante o processo de manutenção. Estes trabalhos consideram também, características da formação das visões. Segue abaixo uma breve descrição dos vários algoritmos definidos nestes trabalhos, agrupados pela quantidade de informação disponível e pelo tipo de visão que pode ser tratada.

1. **Utilizando informação completa**: A maioria dos trabalhos em manutenção de visões assume que todas as relações base e as visões materializadas estão disponíveis para o processo de manutenção, considerando características como agregações, duplicidades, recursão e *outer join*. As técnicas diferem quanto à linguagem de definição utilizada, quanto ao uso das chaves e regras de integridade e quanto à capacidade de tratar inserções e eliminações de forma separada ou em apenas um passo. As atualizações são modeladas e tratadas como uma eliminação seguida de uma inserção. Todas as técnicas funcionam para todas as instâncias da base de dados tanto para inserções como para eliminações de tuplas.

   (a) **Visões não recursivas**: as técnicas apropriadas neste caso incluem o algoritmo de *counting* (descrito acima) apresentado por [GMS93], além de outros algoritmos que também utilizam contadores, como [SI84] que cria estruturas com ponteiros que partem da tupla de origem para suas tuplas derivadas. [BLT86] utiliza os contadores exatamente como o algoritmo *counting*, porém apenas para visões do tipo SPJ, computando inserções e eliminações separadamente. O algoritmo de manutenção por diferenciação algébrica, introduzido por [Pai84] e utilizado posteriormente em [QW91] para manutenção de visões, diferencia expressões algébricas para derivar a expressão relacional que determina a alteração em uma visão SPJ. Este tipo de algoritmo gera duas expressões para

cada visão: uma para definir as inserções na visão e outro, para definir as eliminações. O algoritmo Ceri-Widom [CW91] define regras de produção para manter as visões SQL que não possuam duplicidades, agregações e negação e aquelas onde os atributos das visões determinam funcionalmente a chave da relação base que está sendo atualizada.

(b) **Visões com outer-join**: a manutenção de visões com *outer-join* é discutida em [GJM94], cujo algoritmo redefine a visão, obtendo duas instruções (uma com *left-outer-join* e outra com *right-outer-join*) para calcular as alterações a serem implementadas.

(c) **Visões recursivas**: Os algoritmos citados nesta seção aplicam-se também a visões não recursivas. O primeiro deles é o algoritmo DRed [GMS93], descrito acima. Vários outros algoritmos se aplicam a este tipo de visão, estando entre eles: o algoritmo PF (Propagation/Filtration) definido em [HD92], que é muito similar ao DRed, sendo que, para visões não recursivas, o DRed sempre funciona melhor; o algoritmo Kuchenhoff [Kuc91] que cria regras para calcular a diferença entre estados consecutivos da base de dados e o algoritmo Urpi-Olive [UO92] que cria regras de transição mostrando como cada modificação na relação base se traduz em uma modificação em cada relação derivada.

2. **Utilizando informação parcial**: As visões podem ser mantidas utilizando-se somente um subconjunto das relações envolvidas na definição da visão. De forma diferente da utilização da informação completa, quando a manutenção utiliza apenas informação parcial, nem sempre é possível realizar o processo de manutenção da visão. A manutenção vai depender de aspectos como o tipo de modificação, ou seja, se é uma inserção, uma eliminação ou atualização. Portanto, os algoritmos neste caso devem se preocupar se a visão pode ser mantida e então, como efetuar a manutenção, sendo que em alguns casos, mesmo que não exista o algoritmo para eliminação+inserção, é possível existir o algoritmo para atualização.

(a) **Nenhuma informação disponível**: Muitos trabalhos têm sido feitos para

determinar quando uma modificação na base não acarreta alteração na visão. Esta situação é conhecida como "o problema da atualização irrelevante". [BLT86, BCL89, Elk90, LS93] determinam atualizações irrelevantes em várias condições diferentes.

(b) **Utilizando a visão materializada: Self-Maintenance**. As visões podem ser mantidas utilizando-se apenas a visão materializada e as regras implementadas através das chaves. Estas visões são tratadas por [GJM94], que apresenta vários resultados em *self-maintenance* em visões do tipo SPJ e com *outer-join*, respondendo a inserções, eliminações e atualizações. Este trabalho define uma visão *self-maintainable* com respeito ao tipo de modificação, não considerando as visões com *self-joins* ou *outer-joins*, as visões que não usam informações das chaves e as que não consideram a capacidade de auto manutenção com respeito a todas as instâncias de modificações.

3. **Utilizando a visão materializada e algumas relações base: Referência parcial**. O problema da manutenção com referência parcial é manter a visão, dados apenas um subconjunto das relações base e a visão materializada. Dois subproblemas interessantes surgem desta abordagem: o primeiro é quando estão disponíveis a visão e todas as relações base, com exceção da relação que sofreu a modificação e o segundo, quando estão disponíveis a visão e a relação modificada. Vários trabalhos, como [JMS95, GB95, Gup94], tratam de variações desta situação.

4.5 Conclusão

A manutenção dinâmica das visões materializadas, de maneira similar à definição do processo de carga estático, não é uma tarefa simples. Muitas das etapas envolvidas na carga estática estão presentes na manutenção dinâmica, diluídas nos processos de atualização de cada visão materializada. Existem algumas opções relacionadas ao momento em que as visões são atualizadas. Basicamente, a atualização pode ser imediata, ou seja, no

momento em que os dados base são atualizados, ou protelada, sendo que, neste caso, devem ser considerados os aspectos necessários para minimizar o tempo que as visões ficam indisponíveis para o acesso por parte dos usuários.

Normalmente, as bases de dados fonte, para o data warehouse, podem ser sistemas legados ou sistemas que, de uma forma geral, não conhecem as visões materializadas implementadas no ambiente analítico. As aplicações no ambiente fonte devem informar ao data warehouse quando ocorre uma atualização, por exemplo, a contratação de um novo funcionário ou um paciente que pagou sua conta hospitalar. Entretanto, estas aplicações não têm condições de determinar quais são os dados necessários para se implementar as atualizações nas visões materializadas do data warehouse. Quando a informação sobre uma atualização chega ao data warehouse, temos que buscar os dados adicionais, nas bases fonte, para atualizar as visões corretamente e de forma incremental. Por esta razão, o warehouse deverá emitir algumas consultas para as bases fonte após a ocorrência da atualização. Existem trabalhos que implementam vários algoritmos distintos, para garantir que as visões materializadas no data warehouse sejam atualizadas de forma correta, garantindo a consistência das visões, no menor tempo possível. A escolha do algoritmo mais adequado depende de uma série de fatores. Aspectos como: o tipo da visão implementada, sua dependência das bases de dados fonte durante o processo de atualização, incluindo sua capacidade de se auto manter sem a necessidade de acesso às bases originais, a distribuição dos ambientes fonte e do data warehouse e a concorrência no momento da atualização devem ser levados em conta na escolha do algoritmo de propagação das atualizações mais adequado.

Fica claro que, assim como no modelo de data warehouse estático, com processos de carga periódicos, conforme descrito na seção 3.6 deste trabalho, a atualização do data warehouse em uma implementação mais dinâmica também é a etapa mais complexa e crítica de se implementar.

Ocorre que, para alguns tipos de aplicações transacionais em conjunto com as aplicações analíticas a abordagem estática é preferível, em detrimento da dinâmica e, para outros tipos, ocorre o inverso.

Capítulo 5

A classificação dos domínios transacionais e analíticos

5.1 Introdução

A periodicidade do processo de carga, discutido nas seção 3.6, é quem vai definir os pontos de sincronismo entre os sistemas transacionais e o data warehouse, com suas aplicações analíticas. Quando se quer implementar um data warehouse dinâmico, sincronizado em tempo real com os sistemas transacionais, o intervalo entre os pontos de sincronismo deve ser reduzido de tal forma que as atualizações efetuadas nas bases transacionais sejam enviadas imediatamente para a base analítica.

Falar sobre a manutenção do data warehouse de forma dinâmica, em tempo real, não significa que o processo de carga e suas implicações deixem de existir. Nestas circunstâncias, o processo de carga fica diluído entre as várias transações de manutenção do data warehouse. Cada um dos subprocessos da carga inclui todas as etapas de captura, transformação, enriquecimento, transporte, reformatação e carga das informações envolvidas, o que contribui para manter seu nível de complexidade bastante alto. Outro ponto importante é que estes subprocessos, se implementados nas aplicações transacionais, podem sobrecarregar o processamento do ambiente operacional, fator altamente indesejável.

É importante lembrar que o sincronismo em tempo real é executado em um intervalo de tempo tão pequeno quanto possível, levando-se em conta as condições operacionais necessárias para tratamento e transporte das informações.

Conforme citado anteriormente, o data warehouse pode ser implementado com um modelo de sincronismo híbrido, já que partes ou porções de informações têm comportamentos e requisitos diferentes, tanto transacionais quanto analíticos. Esta visão favorece a otimização dos recursos disponíveis para manutenção do data warehouse, diminuindo ao máximo sua complexidade. Neste trabalho são avaliadas as características de cada uma destas porções de informação, que são provenientes de determinados sistemas transacionais e analíticos, sendo que cada um deles possui um conjunto diferenciado de aspectos funcionais e requerimentos. A definição e as formas de tratamento das porções de informação são tratadas neste capítulo, a partir da seção 5.4.

A definição do modelo mais adequado para o data warehouse requer a análise dos domínios das aplicações transacionais e deve levar em conta fatores como a **aplicabilidade** da carga dinâmica, a **adaptabilidade** do sistema, o **limite** ou **intervalo de tempo** desejável para o sincronismo e o **nível de complexidade** do ambiente transacional. Para cada domínio de aplicação transacional avaliado, as variações destes fatores poderão auxiliar na definição da necessidade e viabilidade de um nível de sincronismo em tempo real, que implicará em um processo dinâmico de carga do data warehouse.

Porém, apenas a análise das aplicações transacionais não é suficiente para caracterizar a implementação deste nível de sincronismo e justificar seus custos. As características das aplicações analíticas também devem ser contempladas. No escopo do ambiente analítico, alguns parâmetros mostram-se relevantes para a avaliação da manutenção dinâmica do data warehouse: o nível de utilização dos **agregados** nas consultas analíticas e a forma de **implementação dos agregados** na base de dados. As próximas seções descrevem, detalhadamente, as características relevantes dos ambientes transacionais e analítico, bem como os parâmetros utilizados para a avaliação.

Vale lembrar que estes parâmetros são aqueles considerados relevantes para o escopo deste trabalho e, certamente, não constituem uma lista completa.

5.2 A análise das características do ambiente transacional

A definição das porções dinâmicas e estáticas do data warehouse deve ser iniciada com a análise das características do ambiente transacional, visando a carga do data warehouse. O ambiente transacional a ser analisado é o conjunto de funções que gera as informações que serão armazenadas nas porções citadas acima. Para que esta avaliação seja feita, é necessário dividir as aplicações transacionais em conjuntos de funções que manipulam uma parte dos dados. Esta parte ou subconjunto dos dados, deve ser considerada como uma unidade funcional dentro do ambiente transacional. Cada uma destas unidades funcionais pode ter comportamentos distintos no que se refere à necessidade de sincronismo com o data warehouse.

Neste contexto, existem alguns fatores que devem ser avaliados para a definição da implementação deste nível de sincronismo. Estes parâmetros são descritos nas próximas seções.

5.2.1 A necessidade da manutenção dinâmica

O primeiro fator relaciona-se às características funcionais dos sistemas transacionais que dão origem às informações do data warehouse. Analisando estas características, pode-se avaliar a aplicabilidade da manutenção dinâmica das informações. Alguns sistemas, devido às suas características funcionais, são considerados estáticos. Casos típicos são encontrados nos sistemas que trabalham com fechamentos periódicos para processamento, sejam eles diários, quinzenais, mensais etc.

Um exemplo bastante característico deste tipo de sistema refere-se ao tratamento de folha de pagamento, que produz resultados a cada período, normalmente quinzenal ou mensal. Neste caso, os fatos a serem avaliados, como, por exemplo, valores de salários pagos, de benefícios concedidos e de descontos efetuados, são obtidos a cada período de processamento, o que leva a uma carga tipicamente estática, com a mesma periodicidade

do processamento das informações.

Outros sistemas podem requerer um intervalo de sincronismo com periodicidade bem menor, de preferência imediata. Talvez o exemplo mais típico de aplicação para esta abordagem seja o *data webhouse*[8]. Um webhouse mantém o *clickstream*[9] de cada usuário, para análise de seu comportamento em determinados ambientes da Internet, podendo ser considerado uma instanciação do data warehouse. Sistemas orientados a entender o comportamento de um cliente, como os sistemas de Gerenciamento de Relacionamento do Cliente (CRM - Customer Relationship Management), além dos sistemas de comércio eletrônico podem requerer a manutenção destes webhouses dinamicamente, já que os resultados dos negócios devem estar disponíveis em tempo real. Nestes casos, a aplicabilidade da carga dinâmica existe e pode ser considerada essencial.

Os valores possíveis para este parâmetro são:

- **Tipicamente estático** - o sistema transacional tem características que requerem uma carga tipicamente estática, sem o menor requisito para a implementação de atualizações dinâmicas;
- **Dinâmico** - o sistema transacional requer, por características funcionais, a atualização dinâmica do data warehouse, no que se refere ao conjunto de transações que está sendo analisado.

5.2.2 A capacidade de adaptação do ambiente transacional

Um importante fator refere-se à possibilidade de se implementar a abordagem dinâmica no sistema transacional analisado, que chamamos de **adaptabilidade** ou capacidade de adaptação. Para que seja possível implementar a manutenção dinâmica do data warehouse, é necessário que exista um controle sobre o sistema transacional e que seja possível

[8] O termo **data webhouse** é utilizado em seu idioma original, o inglês, em toda a literatura escrita em português, da mesma forma que o termo data warehouse.

[9] O termo **clickstream**, utilizado em seu idioma original, o inglês, em toda a literatura escrita em português, significa a seqüência de cliques armazenados que representam a interação do visitante em sua navegação pelas páginas de um *website*.

alterar seus processos para a inserção das funções necessárias ao processo dinâmico de carga. Qualquer sistema que não possibilite estas alterações, não possui a capacidade de adaptação para o processo de sincronismo dinâmico e em tempo real. Nestes casos, o data warehouse funciona como um integrador sistêmico de acesso aos dados, que são atualizados de forma estática.

Em muitos casos, o gerenciador de banco de dados utilizado para armazenar as informações transacionais possui algumas opções, em sua arquitetura, que possibilitam a implementação do sincronismo dinâmico. Uma das opções disponíveis são os *triggers* [10], que aparecem na maioria dos gerenciadores de bancos de dados relacionais. Um *trigger* é uma parte de código que espera pela ocorrência de um evento [UW97], sendo que os eventos possíveis são a inserção ou eliminação de um certo tipo de item de dados. Quando o evento ocorre, uma seqüência de ações é executada. Esta seqüência de ações pode ser codificada de forma a atualizar alguma base de dados intermediária ou, dependendo do caso, diretamente o data warehouse, permitindo a implementação do sincronismo dinâmico e até em tempo real. O nível de sincronismo dinâmico a ser implementado, de acordo com os requisitos funcionais do ambiente transacional, é que vai definir qual a melhor forma de utilização dos *triggers*, dentre as várias opções disponíveis.

Uma alternativa, caso o banco de dados não ofereça a opção descrita acima, é a implementação de um conjunto de rotinas que, ao verificar periodicamente o estado da base de dados, atualiza o data warehouse de forma assíncrona. Estas rotinas podem ser codificadas diretamente dentro do sistema transacional, e verificam as mudanças ocorridas nas bases de dados, enviando para o ambiente analítico as novas informações. Outra alternativa é a análise periódica dos *logs* do banco de dados, onde ficam armazenadas todas as alterações ocorridas nos dados. Um conjunto de rotinas pode verificar periodicamente as inserções e eliminações relacionadas aos dados que estão sendo analisados.

Através da implementação de uma destas opções, o sincronismo dinâmico pode ser implementado. Obviamente, deve-se considerar também os casos onde nenhuma das opções

[10] *Trigger* ou gatilho, em português. A palavra *trigger* é normalmente utilizada, mesmo na literatura em português.

acima é possível, seja pelas limitações do gerenciador de banco de dados utilizado, seja pela impossibilidade de alterar a aplicação transacional, o que limitará o sincronismo unicamente à opção estática.

Os valores que este parâmetro pode assumir são:

- **Permite a implementação da carga dinâmica** - é possível inserir nas transações envolvidas, os requisitos necessários para obter e propagar as informações ao data warehouse;
- **Não permite** - não existe um nível mínimo de controle para implementar as funções.

5.2.3 O intervalo de tempo desejável para o sincronismo

Outro fator a ser avaliado está diretamente relacionado com o **intervalo de tempo** desejável para o sincronismo. Deve-se avaliar a real necessidade do sincronismo dinâmico, em tempo real ou em um curto intervalo de tempo, já que o custo gerado por este tipo de implementação pode ser alto demais para os benefícios proporcionados. Nesta avaliação, duas perguntas devem ser formuladas. Qual a periodicidade que o sistema transacional implementa em seu processamento? Quais são as aplicações que requerem o menor intervalo de tempo de sincronismo possível? Ao responder estas questões pode-se identificar quais conjuntos de informações precisam estar sincronizados dinamicamente em tempo real ou com um intervalo de tempo determinado. Com base neste requerimento pode-se definir um panorama que justifique o custo da manutenção dinâmica do data warehouse.

Para o escopo deste trabalho, os valores possíveis são:

- **Mínimo possível (Tempo Real)** - o sincronismo do data warehouse com os dados transacionais deve ser feito em tempo real, observando-se apenas o tempo mínimo necessário para o transporte e processamento das informações;
- **Intervalo determinado** - a aplicação requer um sincronismo em determinado período de tempo, maior que o requerido em tempo real, porém menor que a periodicidade da carga estática;

- **Sincronismo postergado** - o sincronismo pode ser mantido em intervalos regulares, determinado pelo processo estático de atualização do data warehouse, quando da sua carga periódica.

5.2.4 A complexidade no ambiente transacional

Deve-se analisar também um fator que se refere à **complexidade** do processo no ambiente transacional. Em outras palavras, deve-se definir quanto custa levar o conjunto de informações do ambiente transacional para o data warehouse, no momento em que as informações são atualizadas na base operacional. Para isto, é necessária uma análise das características dos domínios das aplicações transacionais que gerenciam as informações a serem transportadas para o data warehouse. É muito importante avaliar o nível de agregação das informações que vão para o data warehouse. Se o data warehouse utiliza informações do nível mais baixo de granularidade do ambiente transacional, o processo de carga fica simplificado. Manter as informações neste nível de granularidade torna o data warehouse bastante similar ao ambiente operacional, minimizando a complexidade envolvida nos subprocessos de carga e, por conseqüência, o limite de tempo necessário para sua execução. Este parâmetro pode ser analisado sob dois aspectos distintos, cujas descrições seguem nas próximas seções.

A complexidade na propagação

A **complexidade na propagação** está relacionada ao número de diferentes objetos envolvidos na transação avaliada, quando da realização de uma atualização na base transacional. A cada atualização considerada relevante para o data warehouse, um ou mais objetos do ambiente transacional estão envolvidos e, para que o sincronismo possa ser efetivado, é preciso considerar qual o nível de complexidade da transação, já que estas atualizações serão refletidas no ambiente analítico. Obviamente, quanto maior o número de objetos envolvidos no processo transacional, mais complexa será a propagação das informações para o data warehouse.

Os intervalos de valores considerados neste trabalho são:

- **Baixa** - o número de objetos distintos envolvidos na atualização é pequeno, da ordem de 1 ou 2;
- **Média** - de 3 a 5 objetos objetos distintos envolvidos;
- **Alta** - o número de objetos distintos envolvidos é alto, envolvendo mais de 5 objetos.

A complexidade na consistência

A **complexidade na consistência** está diretamente relacionada à complexidade das verificações e validações que devem ser executadas pela transação imediatamente após a atualização na base transacional, com o objetivo de disponibilizar o conjunto de informações para o processo de carga dinâmica. Pode ocorrer que, mesmo envolvendo poucos objetos diferentes na atualização, a transação requeira alto grau de complexidade na verificação de consistência das informações a serem propagadas. Considerando, por exemplo, a operação realizada no modelo relacional, em SQL (Structured Query Language [UW97]), os possíveis valores são:

- **Baixa** - não existe, ou é extremamente pequena a necessidade de validar ou verificar elementos adicionais à informação que está sendo inserida durante a transação. A única função necessária é o repasse da informação, tal como se apresenta, para o data warehouse. São os casos de seleção sem envolvimento de operações mais complexas como, por exemplo, um *join*;
- **Média** - a complexidade envolvida é de nível médio, sendo as operações de seleção que envolvem *join*, na preparação das informações para a carga;
- **Alta** - a transação, devido à sua complexidade, requer alta complexidade nas verificações e validações para a preparação das informações a serem levadas ao data warehouse. É o caso de operações de seleção que envolvem *join* das informações e uso de operadores *group by* e *having* .

5.3 A análise das características do ambiente analítico

Em relação às aplicações analíticas, deve-se analisar quais os tipos de consultas que serão executadas sobre o data warehouse. Diferentes tipos de usuários requerem diferentes tipos de consultas, envolvendo as mais variadas visões do data warehouse em suas inúmeras combinações de agregações e dados detalhados.

A partir deste ponto, portanto, inicia-se uma avaliação do comportamento analítico do sistema. Um data warehouse contém informações armazenadas em um nível de detalhe básico, já definido em seções anteriores como sua granularidade. Obviamente, as várias tabelas fato contidas no data warehouse podem conter níveis de granularidade diferentes e, portanto, a avaliação do comportamento analítico deve ser realizada sobre os vários subconjuntos de informações contidas nesta base.

A grande maioria das consultas ao data warehouse envolvem funções de agregação como *sum* , *count* e *group by* , entre outras. Não é difícil perceber que a complexidade das aplicações analíticas depende dos níveis de agregação envolvidos em cada consulta e, para agilizar seu tempo de resposta, várias estratégias podem ser implementadas. Estas estratégias efetuam a pré-computação de partes das informações, agregadas em diferentes níveis, conforme sua utilização nas consultas.

Algumas das técnicas comuns para utilização de agregados em data warehouse são:

- Agregação dinâmica [Alb99], quando a agregação das informações ocorre durante a execução das consultas, isto é, cada consulta, quando executada tem seus resultados de agregação calculados no momento de sua execução. Este processo, se realizado por uma ferramenta OLAP, pode ser executado e armazenado para uso posterior;
- A criação de visões que contemplem as funções de agregações necessárias, chamadas "visões de agregações"[IA99], utilizadas e computadas durante a execução das consultas;
- Visões de agregações são armazenadas em *caches* e utilizadas para responder consultas executadas posteriormente [DRSN98]. As consultas podem ser decompostas

em múltiplas consultas, sendo que cada uma delas pode ser resolvida por uma visão diferente;

- Tabelas de sumariação [MQM97], que representam visões de agregação que, como as visões citadas no item anterior, contêm diferentes conjuntos de atributos utilizados em funções *group by* além de computar várias outras funções de agregação. A diferença é que estas tabelas de sumariação são visões materializadas e, portanto, as alterações das tabelas base devem ser refletidas nestas visões;
- Resultados de consultas, materializadas no data warehouse, que podem ser reutilizados em consultas semelhantes executadas posteriormente;
- Agregados multidimensionais, armazenados em cubos [AAD+96]. Estes cubos são gerados através da utilização do operador *cube*, apresentado em [GBLP96], que é uma generalização *n*-dimensional do operador *group by*. O operador *cube* executa as funções *group by* correspondentes a todas possíveis combinações de uma lista de atributos envolvidos na consulta que gera o cubo. Estes cubos são estruturas físicas separadas do data warehouse central e sua geração é executada através de um procedimento executado periodicamente.

Estas técnicas otimizam o tempo de execução das aplicações analíticas, que requerem diferentes níveis e quantidades de agregados. Desta forma, pode-se perceber que a complexidade destas aplicações depende do nível de agregação utilizado em suas consultas.

Uma alternativa para avaliação do comportamento do ambiente analítico está baseada em dois parâmetros: uma classificação das consultas dentro do nível de agregação que elas utilizam e a forma como as agregações necessárias foram implementadas no data warehouse. A análise destes dois parâmetros, em conjunto, pode contribuir para caracterizar a viabilidade da implementação da abordagem dinâmica no data warehouse, conforme apresentado nas próximas seções.

5.3.1 A utilização dos agregados nas consultas analíticas

Este parâmetro refere-se à **utilização dos agregados** nas consultas analíticas. Algumas consultas ficam limitadas aos dados no nível mais detalhado, enquanto que outras utilizam apenas valores agregados ou uma combinação entre detalhes e agregações:

As consultas analíticas podem, então, ser classificadas segundo sua utilização de agregados, da seguinte forma:

- **Apenas detalhe** - as consultas envolvidas não utilizam agregados e, portanto, acessam somente a informação em seu nível de granularidade mais baixo;
- **Agregados** - as consultas executam funções como *sum*, *count*, *group by* e outras, fornecendo informações agregadas ao usuário.

5.3.2 A forma de implementação dos agregados

Este parâmetro descreve a forma de **implementação dos agregados** na base analítica. Para classificar as opções de implementação de agregações, as várias alternativas apresentadas anteriormente nesta seção, foram simplificadas. Assim, os valores que este parâmetro pode assumir são:

- **Nenhuma** - indica que a base analítica não contém pré-agregação das informações. Qualquer requerimento para acesso a valores agregados deve ser resolvido pela própria aplicação, computando o agregado durante a execução da consulta;
- **Visões virtuais com agregados** - a base analítica possui visões de agregações, não materializadas, que podem ser utilizadas pelas consultas, porém são computadas durante a execução das mesmas;
- **Visões materializadas com agregados** - indica que a base analítica possui tabelas ou visões de agregações materializadas, que contêm as informações agregadas requeridas pela consulta. Estas tabelas ou visões devem refletir quaisquer mudanças ocorridas nas tabelas de origem.

- **Cubos** - os cubos representam a opção de manter, em uma base de dados à parte, todos os níveis de agregações requeridos pelas consultas, previamente preparados e armazenados junto com a base analítica.

5.4 Os parâmetros para a avaliação do modelo de sincronismo

As características dos ambientes transacional e analítico, descritas nas seções 5.2 e 5.3 devem, agora, ser consideradas como parâmetros e avaliadas em conjunto. Esta avaliação indica qual a interferência de um parâmetro nos outros, com o objetivo de se obter diretrizes para definição do modelo de sincronismo do data warehouse.

Na tabela 5.1 pode-se visualizar, resumidamente, os parâmetros e seus possíveis valores.

Para maior clareza da análise, os parâmetros serão apresentados sob a forma de árvores de decisão. Uma das árvores representa a avaliação dos parâmetros do ambiente analítico, enquanto que a outra representa os parâmetros do ambiente transacional.

Desta forma, fica mais simples determinar quais os modelos possíveis, sincronismo dinâmico ou estático, para o data warehouse, baseado no conjunto de valores informados para os parâmetros.

As folhas de cada árvore representam o modelo de sincronismo proposto para o data warehouse com base no caminho da decisão, em função dos valores apresentados. O modelo escolhido é o que requer menor intervalo de tempo possível para o sincronismo, buscando-se sempre um data warehouse sincronizado com o ambiente operacional na maior parte do tempo. As opções apresentadas para o modelo de sincronismo são:

- **Dr** - modelo dinâmico com sincronismo em tempo real, considerando-se apenas o tempo necessário para a seleção e transporte dos dados;
- **Ddt** - modelo dinâmico com sincronismo em determinado intervalo de tempo (Δt);
- **E** - modelo estático com sincronismo baseado em cargas periódicas.

Parâmetros do ambiente transacional:
P1 - Aplicabilidade Tipicamente estático Dinâmico
P2 - Adaptabilidade Permite a implementação da carga dinâmica Não permite
P3 - Intervalo de tempo Mínimo possível (tempo real) Intervalo determinado Sincronismo postergado (carga estática)
P4 - Complexidade na propagação Baixa Média Alta
P5 - Complexidade na consistência Baixa Média Alta
Parâmetros do ambiente analítico:
P6 - Utilização dos agregados Apenas detalhe Agregados
P7 - Implementação dos agregados Nenhuma Visões virtuais com agregados Visões materializadas com agregados Cubos

Tabela 5.1: Resumo dos parâmetros avaliados para a definição do modelo do data warehouse

A decisão de se representar duas opções para o modelo dinâmico (Dr e Ddt) deve-se ao fato de que a atualização dinâmica do data warehouse pode permitir variações do intervalo de tempo para o sincronismo. O valor deste intervalo de tempo depende dos requisitos impostos pelo ambiente analítico e transacional e também da complexidade das operações que devem ser realizadas para se implementar o sincronismo. Nos limites, estão

a alternativa **Dr**, que representa o sincronismo dinâmico em tempo real e a alternativa **E**, que representa o sincronismo baseado na carga estática periódica. A alternativa **Ddt** representa o sincronismo dinâmico, com um intervalo de tempo (Δt) que varia dependendo da complexidade e da necessidade das aplicações. Este Δt é maior que o utilizado pela opção de sincronismo em tempo real e menor que o utilizado pela carga periódica. Pode ocorrer que um determinado caminho na árvore seja inválido, devido a um certo conjunto de valores. Isto acontece quando a combinação de valores de dois ou mais parâmetros é considerada incompatível. Nas árvores de decisão, estes valores estão representados por '–'.

5.4.1 A árvore de decisão do ambiente transacional

A árvore de decisão do ambiente transacional, gerada pelos cinco parâmetros descritos no capítulo 2, resulta em um total de 108 diferentes caminhos. Os parâmetros P1 - Aplicabilidade, P2 - Adaptabilidade e P3 - Intervalo de tempo, definem grupos de caminhos onde, a aplicação dos parâmetros P4 - Complexidade na propagação e P5 - Complexidade da consistência não interferem. Por esta razão, e por motivos de melhor visualização, a árvore mostrada abaixo, representa apenas os parâmetros P1, P2 e P3, indicando os grupos de caminhos, que levarão aos mesmos resultados, após a aplicação dos parâmetros P4 e P5.

Com isso, a avaliação detalhada deve ser aplicada a um conjunto de caminhos menor, requerendo a análise conjunta dos cinco parâmetros (P1, P2, P3, P4 e P5). Estes caminhos, na árvore abaixo, estão identificados como (7) e (8).

Portanto, a árvore de decisão do ambiente transacional, **resumida aos parâmetros P1, P2 e P3** pode ser representada da seguinte forma, conforme a figura 5.1.

Cabe ressaltar algumas observações:

1. Os caminhos (1) e (2) foram considerados inaplicáveis, já que um sistema transacional que é considerada tipicamente estático, de acordo com a definição do parâmetro P1, não apresenta requisitos para a implementação de atualizações dinâmicas. Desta

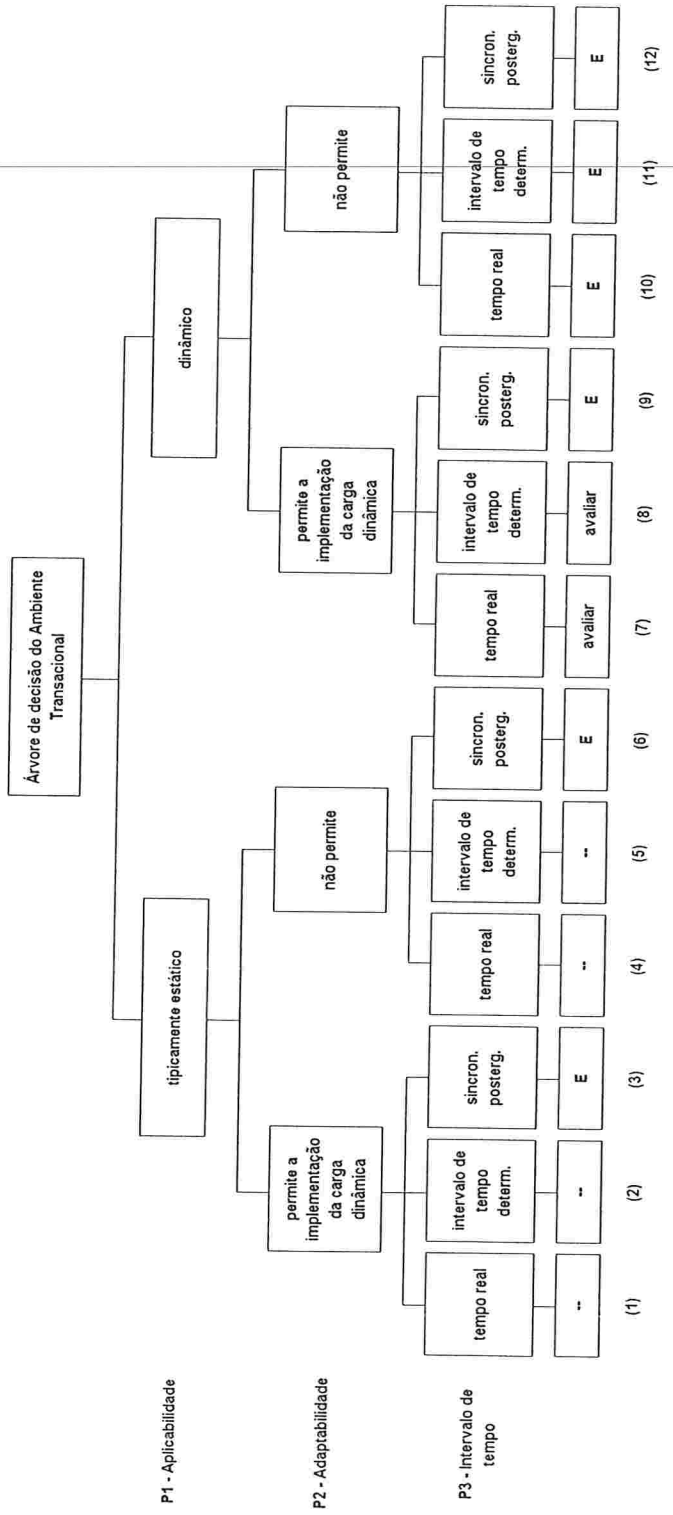


Figura 5.1: Árvore de decisão resumida, para análise do ambiente transacional

forma, não faz sentido requerer intervalo de tempo (P3) como tempo real ou intervalo determinado que são requisitos específicos de sincronismo dinâmico;

2. O caminho (3), leva a uma implementação estática, com sincronismo postergado, independentemente dos parâmetros que estariam nos níveis abaixo (P4 e P5 não representados na árvore resumida);
3. Para os caminhos (4) e (5) vale a mesma observação 1, descrita acima;
4. Para o caminho (6) vale a mesma observação 2, descrita acima;
5. Os caminhos (7) e (8) são os únicos caminhos que requerem uma avaliação em conjunto com os outros parâmetros P4 e P5, sendo que a árvore resumida acima é insuficiente para apresentar qualquer resultado;
6. Na representação do caminho (9), apesar da aplicação apresentar características funcionais para implementação do sincronismo dinâmico (em P1) e permitir a implementação da carga dinâmica (em P2), o intervalo de tempo requerido é de sincronismo postergado, levando a uma opção mais adequada que é a atualização por carga periódica, estática, que não necessita da avaliação dos parâmetros P4 e P5, não representados na árvore resumida;
7. Os caminhos (10), (11) e (12) também levam a uma solução tipicamente estática, já que o ambiente transacional não possui capacidade de adaptação que permita implementar a manutenção dinâmica, limitando assim as opções disponíveis. Também, neste caso, são irrelevantes os valores apontados pelos parâmetros P4 e P5.

Conforme as observações acima, resta apenas um conjunto de caminhos que deve ser avaliado incluindo todos os parâmetros definidos para a análise do ambiente transacional. A árvore mostrada na figura 5.2 representa os caminhos (7) e (8), da árvore resumida (5.1), incluindo todos os parâmetros (P1, P2, P3, P4 e P5).

As observações importantes para a árvore parcial representada na figura 5.2, são:

1. Todos os caminhos levam a uma solução dinâmica, já que P1 e P2 indicam que este tipo de sincronismo é aplicável e existem meios de implementá-lo;

2. Conforme a complexidade dos parâmetros P4 e P5 aumenta, também aumenta o valor do intervalo de tempo necessário para o transporte e processamento das informações. Isto indica que o sincronismo em tempo real só é possível quando as complexidades são mais baixas.

A figura 5.2 representa a árvore de decisão do ambiente transacional, exibindo os detalhes dos caminhos (7) e (8).

5.4.2 A árvore de decisão do ambiente analítico

A árvore de decisão do ambiente analítico pode ser representada conforme a figura 5.3.

Algumas observações sobre esta árvore são relevantes:

1. O Δt do caminho (4) pode ser considerado maior que o Δt do caminho (3), já que a carga do cubo de informações normalmente requer mais tempo que a atualização da visão materializada, ao se avaliar o mesmo conjunto de informações. A mesma relação ocorre entre os caminhos (7) e (8);
2. O caminho (5) é considerado inválido, já que a aplicação utiliza agregados (conforme o parâmetro P6), porém não existe implementação dos agregados necessários (conforme o parâmetro P7).

5.5 A função de análise dos parâmetros

A análise dos parâmetros, descritos nos capítulos anteriores, deve ser feita sobre subconjuntos de informação contidos no data warehouse. Cada um destes subconjuntos contém informações provenientes de uma ou mais funções de negócio das aplicações do ambiente transacional. Desta forma, ao se definir uma porção de informações do data warehouse, deve-se assumir que a análise será efetuada sobre esta porção, levando em conta também as características da base que a originou, no ambiente transacional. Para que esta função de análise possa ser formalizada, são necessárias algumas definições, que são apresentadas abaixo.

Figura 5.2: Árvore de decisão parcial para análise do ambiente transacional

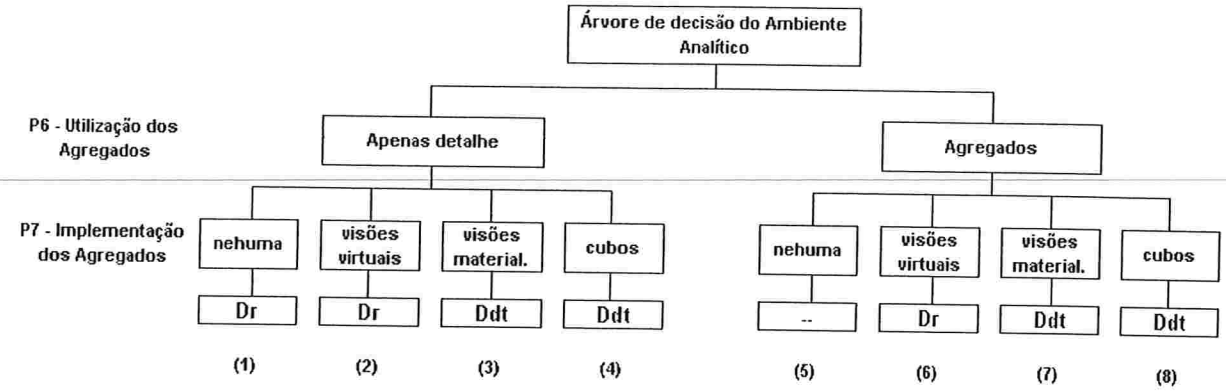


Figura 5.3: Árvore de decisão para avaliação do ambiente analítico

i Seja **D** um conjunto que representa todos os elementos (valor atômico de informação) pertencentes a um data warehouse, definido como:

$$D = \{x \mid x \text{ é elemento do data warehouse}\}$$

ii Seja o conjunto **P(D)** constituído por todos os sub-conjuntos que representam porções de informação do data warehouse, identificados por **X** e representados da seguinte forma:

$$P(D) = \{X \mid X \subset D\} \quad \text{e}$$

$$(\forall x)\ (x \in X \rightarrow x \in D) \quad \text{ou seja, X} \subset \text{D.}$$

Estas porções de elementos do data warehouse configuram uma unidade para avaliação e aplicação da função de análise.

iii Os domínios válidos para os parâmetros P1, P2, P3, P4, P5, P6 e P7, definidos na seção 5.4, são:

$$P1 = \{\text{'tipicamente estático', 'dinâmico'}\};$$

$$P2 = \{\text{'permite a implementação da carga dinâmica, 'não permite'}\};$$

$$P3 = \{\text{'mínimo possível', 'intervalo determinado',}$$

$$\text{'sincronismo postergado'}\};$$

$P4 = \{$'baixa', 'média', 'alta'$\}$;

$P5 = \{$'baixa', 'média', 'alta'$\}$;

$P6 = \{$'apenas detalhe', 'agregados'$\}$ e

$P7 = \{$'nenhuma', 'visões virtuais com agregados',

'visões materializadas com agregados', 'cubos'$\}$;

iv Seja **A** um conjunto que contém uma combinação de valores válidos para os parâmetros P1, P2, P3, P4, P5, P6 e P7 respectivamente, tal que:

$$A = \{(p_1, p_2, p_3, p_4, p_5, p_1, p_7) \mid p_1 \in P1, p_2 \in P2, p_3 \in P3, p_4 \in P4,$$
$$p_5 \in P5, p_6 \in P6, p_7 \in P7\}$$

v Seja **B** o conjunto que representa os possíveis níveis de sincronismo para a definição do modelo, conforme a seção 5.4, tal que:

$$B = \{\text{'Dr'}, \text{'Ddt'}, \text{'E'}\}$$

Analogamente à definição (ii), temos o conjunto $\mathbf{P(B)}$ constituído por todos os sub-conjuntos que representam combinações dos elementos do conjunto **B**, tal que:

$P(B) = \{Y \mid Y \subset B\}$ e

$(\forall y)\ (y \in Y \rightarrow y \in B)$ ou seja, Y $\subset$ B.

A **função de análise e avaliação dos parâmetros** será designada por γ e aplicada sobre uma porção de informações do data warehouse, tendo por definição:

$$\gamma(\mathbf{P(D)},\mathbf{A}) \rightarrow \mathbf{P(B)}$$

onde:

- $\mathbf{P(D)}$ representa o conjunto que contém as porções de informação escolhidas para o processo de análise, definido em (ii);
- **A** é um conjunto de combinações de valores válidos para os parâmetros P1, P2, P3, P4, P5, P6 e P7, definido em (iv) e

- **P(B)** é o conjunto de valores que representam os possíveis níveis de sincronismo, definido em (v).

Sendo X $\subset$ P(D), Y $\subset$ P(B) e (p_1, p_2, p_3, p_4, p_5, p_6, p_7) $\in$ A, temos:

$$\gamma(X,\ p_1,\ p_2,\ p_3,\ p_4,\ p_5,\ p_6,\ p_7) \rightarrow\ Y$$

Esta função γ deve ser aplicada para todos os subconjuntos contidos no conjunto **P(D)**, de forma a poder avaliar qual a opção de sincronismo é mais adequada para cada uma das porções de informação, também em função dos valores escolhidos p_1, p_2, p_3, p_4, p_5, p_6 e p_7, conforme as características do subconjunto **X**.

Para fins de aplicação prática, a função γ deve ser aplicada em etapas, conforme descrito abaixo:

- **Etapa 1** - a função é aplicada ao conjunto de parâmetros do **ambiente transacional**, da seguinte forma:

$$\gamma\ (X,\ p_1,\ p_2,\ p_3,\ p_4,\ p_5) \rightarrow\ Y_1$$

- **Etapa 2** - a função é aplicada ao conjunto de parâmetros do **ambiente analítico**, da seguinte forma:

$$\gamma\ (X,\ p_6,\ p_7) \rightarrow\ Y_2$$

- **Etapa 3** - sobre os conjuntos resultantes $\mathbf{Y}_1$ e $\mathbf{Y}_2$ é aplicada a função:

$$\delta\mathbf{:}\ \mathbf{P(B)} \times \mathbf{P(B)} \rightarrow \mathbf{P(B)}$$

tal que δ determina em **P(B)** o **sub-conjunto unitário** que contém a opção de implementação mínima. A função δ estabelece a **precedência de complexidade** entre as opções de sincronismo. A definição da complexidade de cada opção está relacionada ao número de transações que efetuam trocas de mensagens entre os ambientes transacional e analítico, quando da implementação do modelo de sincronismo escolhido. A precedência de complexidade entre as opções definidas no item (v) desta seção pode ser representada por:

$$E \prec Ddt \prec Dr$$

que indica que a opção de sincronismo E é a menos complexa, efetuando menor quantidade de trocas de mensagens que as demais. Isto por que a implementação da carga estática de uma determinada porção de informações é normalmente realizada em um conjunto de processos *batch*, onde as informações coletadas da base transacional são tratadas e posteriormente remetidas ao ambiente analítico como um lote. A opção de sincronismo mais complexa é a *Dr*, que efetua o processamento e o envio das informações coletadas dos sistemas transacionais ao ambiente analítico de forma síncrona com a alteração das bases de origem. Pode-se imaginar que, em determinados sistemas, a quantidade de trocas de mensagens entre os ambientes transacional e analítico, quando este modelo de sincronismo está implementado, é bastante alta. Seguindo os raciocínios expostos acima, pode-se concluir que a opção *Ddt* fica em um nível intermediário na precedência de complexidade implementada pela função δ.

Assim, pode-se definir os resultados da aplicação da função δ como sendo:

$\delta('E','Ddt','Dr') = \{'E'\};$

$\delta('E','Ddt') = \{'E'\};$

$\delta('Dr','Ddt') = \{'Ddt'\};$

$\delta('E','Dr') = \{'E'\};$

$\delta('Dr','Dr') = \{'Dr'\};$

$\delta('Ddt','Ddt') = \{'Ddt'\}$ *e*

$\delta('E','E') = \{'E'\};$

O sub-conjunto unitário resultante da aplicação da função δ deve ser considerado como o resultado da aplicação da função de análise γ.

Ao aplicarmos a função γ para dois subconjuntos $\mathbf{X}$ e $\mathbf{X'}$, sendo $\mathbf{X} \subset \mathbf{P(D)}$ e $\mathbf{X'} \subset \mathbf{P(D)}$, utilizando os valores p_1, p_2, p_3, p_4, p_5, p_6, p_7 e p_1', p_2', p_3', p_4', p_5', p_6', p_7', respectivamente, temos:

$\gamma\ (\mathbf{X}, p_1, p_2, p_3, p_4, p_5, p_6, p_7) \rightarrow \mathbf{Y}$ e

$$\gamma\ (\mathbf{X'},\ p_1',\ p_2',\ p_3',\ p_4',\ p_5',\ p_6',\ p_7') \rightarrow \mathbf{Y'}$$

Pode, então, ocorrer que:

1. $\{x \in D \mid x \in X \vee x \in X'\} = X \cap X' = \emptyset$

   ou seja, X e X' são conjuntos disjuntos ou;

2. $\{x \in D \mid x \in X \wedge x \in X'\} = X \cap X' \neq \emptyset$

Na situação (1), onde os conjuntos são disjuntos, o modelo de sincronismo a ser implementado em **X** e **X'** é o que foi definido pela função γ, **Y** e **Y'**, respectivamente.

Na situação (2), ou seja, no caso das porções serem conjuntos não disjuntos, após a aplicação da função de análise γ, devemos proceder com a escolha da opção de implementação mais adequada, dadas as características dos subconjuntos de informações envolvidos. Neste caso, algumas opções devem ser avaliadas, a saber:

1. Criação de três porções distintas de informação:

   - **Porção X - (X ∩ X')**, cuja opção de sincronismo será aquela definida inicialmente por γ, ou seja **Y**;
   - **Porção X' - (X ∩ X')**, cuja opção de sincronismo será aquela definida inicialmente por γ, ou seja **Y'**;
   - **Porção (X ∩ X')**, cuja opção de sincronismo deverá ser escolhida com base no conteúdo e forma de utilização do conjunto de informações. Neste caso, deve-se levar em conta que uma determinada opção de sincronismo, diferente de **Y** ou de **Y'** pode acarretar inconsistências temporárias na base analítica, até que as três porções estejam sincronizadas novamente. Ainda assim, esta é uma alternativa viável, já que esta inconsistência temporária pode não ser relevante para as aplicações analíticas que envolvam esta porção de informações.

2. Criação de duas porções de informacão:

   - **Porção X**, cuja opção de sincronismo será aquela definida inicialmente por γ, ou seja **Y**;

- **Porção X'**, cuja opção de sincronismo será aquela definida inicialmente por γ, ou seja **Y'**.

  Neste caso, mantêm-se as estratégias já definidas, porém, faz-se necessário um controle de concorrência de sincronismo sobre as informações que estão contidas em (X ∩ X'). Isto por que estas informações serão atualizadas tanto pelos processos de sincronismo para **X** quanto pelos processos para **X'**, já que fazem parte de ambos os conjuntos.

3. Criação de apenas uma porção de informação:

   - **Porção (X ∪ X')**, cuja opção de sincronismo deve ser decidida, escolhendo-se a de menor impacto, aqui considerada a de menor complexidade. Isto pode ser obtido com a aplicação da função δ aos subconjuntos unitários resultantes da aplicação da função de análise γ em **X** e **X'**, ou seja, δ deve ser aplicada sobre os subconjuntos **Y** e **Y'**. Assim, a opção de sincronismo final resultante neste caso, é definida por:

     $\delta(\gamma(\mathbf{X}, p_1, p_2, p_3, p_4, p_5, p_6, p_7), \gamma(\mathbf{X'}, p_1', p_2', p_3', p_4', p_5', p_6', p_7')) \rightarrow \mathbf{Y''}$

     sendo $\mathbf{Y''} = \mathbf{Y}$ ou $\mathbf{Y''} = \mathbf{Y'}$.

5.6 Conclusão

Neste capítulo foi apresentada uma análise das características do ambiente transacional e do analítico, que permitiu definir o conjunto de parâmetros para avaliação do modelo de sincronismo do data warehouse. A partir destas características foram apresentadas as árvores de decisão dos ambientes transacional e analítico, com o objetivo de indicar qual a melhor opção de sincronismo para uma determinada combinação de valores dos parâmetros.

A função de análise dos parâmetros foi formalizada, descrevendo também as características das porções de informações avaliadas. Este capítulo apresenta a premissa prin-

cipal do trabalho que é a possibilidade de existir um modelo híbrido para o sincronismo do data warehouse, servindo de base para a conclusão apresentada no capítulo 7.

Capítulo 6

A aplicação da classificação dos domínios transacionais e analíticos

6.1 A implementação de um protótipo para avaliação do ambiente

No decorrer deste trabalho ficou clara a necessidade de se desenvolver uma ferramenta para possibilitar a implementação da função de análise descrita na seção anterior, de tal forma que situações do ambiente real sejam avaliadas. Portanto, foi desenvolvido um protótipo, com o objetivo de validar a função de análise descrita na seção 5.5.

Um outro ponto importante deve ser destacado: a escolha de uma opção de sincronismo é dinâmica. Já que, ao longo do tempo, as características do ambiente transacional e do ambiente analítico se modificam e podem requerer uma opção de sincronismo mais adequada.

No caso do ambiente transacional, as mudanças ocorrem mais raramente. Fatores como alterações dos requisitos de negócios, planos econômicos ou substituição completa do aplicativo podem levar a alterações nos valores dos parâmetros analisados anteriormente.

O mesmo ocorre no ambiente analítico, porém de forma muito mais freqüente. Novos usuários, novos requisitos de análise, períodos de tempo determinados que requerem

análises específicas e, em muitos casos, muito mais complexas, provocam mudanças significativas nos perfis das consultas executadas sobre esta base. O aumento no volume de dados do data warehouse pode requerer diferentes estratégias de agregação, assim como as novas consultas podem precisar de informação mais agregada que o habitual.

Fica claro que os parâmetros utilizados para a escolha da opção de sincronismo devem ser monitorados freqüentemente, com o objetivo de se identificar possíveis alterações no comportamento dos ambientes transacional e analítico. Este monitoramento permite avaliar a necessidade de implementação de uma opção de sincronismo que pode ser completamente diferente daquela em vigor.

Para facilitar a escolha da melhor opção de sincronismo, no ambiente real e a variação de comportamento descrita acima, foi desenvolvida, no protótipo, uma função que simula a coleta dos parâmetros de análise.

Este protótipo tem como objetivo auxiliar, de forma prática, a modelagem do data warehouse, no que se refere à implementação do nível de sincronismo mais adequado para cada uma das porções de dados avaliadas.

Nas seções seguintes são descritas a estrutura de dados utilizada e as principais funções implementadas neste protótipo.

6.1.1 A implementação das árvores de decisão

As árvores de decisão, descritas nas seções 5.4.1 e 5.4.2 foram implementadas sob a forma de tabelas, em um banco de dados relacional, sendo uma tabela para representar a árvore do ambiente transacional e outra para a árvore do ambiente analítico. A escolha desta implementação é decorrente da facilidade de se efetuar consultas sobre estas tabelas, variando-se os valores dos parâmetros, conforme o perfil da porção de dados avaliada.

Cada caminho da árvore de decisão, incluindo a folha, com a opção de sincronismo, foi implementada como uma linha da tabela.

O protótipo possui funções para a manutenção e carga destas tabelas. Na figura 6.1 pode-se visualizar um detalhe da implementação, com a função de manutenção da tabela

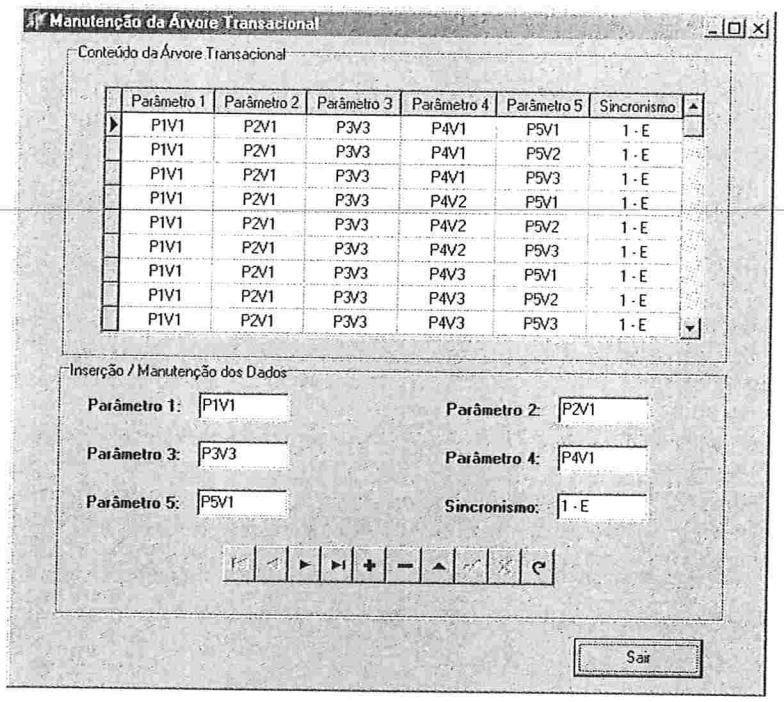


Figura 6.1: **Protótipo** - função de manutenção na tabela que representa a árvore de decisão para avaliação do ambiente transacional

que contém a árvore de decisão do ambiente transacional.

6.1.2 A função de avaliação dos parâmetros

No protótipo foi implementada uma função que, após informados os valores dos parâmetros, busca na árvore de decisão qual(is) o(s) caminho(s) válido(s) para o conjunto de parâmetros informado, apresentando a opção de sincronismo resultante .

O algoritmo utilizado, representado na figura 6.2 foi baseado nas funções γ e δ, cujas descrições encontram-se na seção 5.5.

Além dos valores válidos para estes parâmetros, apresentados na seção 5.4, foi acrescentada a opção *qualquer*, que indica o desconhecimento do valor de um determinado parâmetro ou quando este valor não é único. No caso da escolha do valor *qualquer* para um ou mais parâmetros, são pesquisados, na árvore de decisão, os vários caminhos, incluindo, como argumento de pesquisa, todos os valores possíveis para aquele(s) parâmetro(s).

ALGORITMO ANÁLISE DE CAMINHO:

- ▷ Recebe os valores para os parâmetros P1, P2, P3, P4, P5, P6 e P7;
- ▷ Busca, na árvore transacional, o caminho que contenha os nós cujos valores correspondem aos informados em P1, P2, P3, P4 e P5;
  - O valor do nó folha, para aquele caminho, é a opção de sincronismo da árvore transacional;
- ▷ Busca, na árvore analítica, o caminho que contenha os nós cujos valores correspondem aos informados em P6 e P7;
  - O valor do nó folha, para aquele caminho, é a opção de sincronismo da árvore analítica;
- ▷ Escolhe a menos complexa entre as opções de sincronismo da árvore transacional e analítica;
- ▷ Retorna a opção resultante.

Figura 6.2: Descrição do algoritmo utilizado para implementar a função de análise

A figura 6.3 apresenta a aplicação da função de análise para um determinado conjunto de valores especificados.

6.1.3 A avaliação do comportamento das porções de informação

Conforme descrito no início deste capítulo, a escolha de uma opção de sincronismo não é definitiva, na medida que as características do ambiente transacional e do ambiente analítico se modificam com o passar do tempo e a opção de sincronismo tende a mudar. Por esta razão foi considerado importante o desenvolvimento de uma função para avaliar o comportamento de cada porção de informações do data warehouse, em função dos valores de seus parâmetros, ao longo do tempo. Assim, foi desenvolvida inicialmente uma rotina para simular a coleta periódica dos valores dos parâmetros, para que, porteriormente, fosse possível identificar a necessidade de alteração na opção de sincronismo implementada.

O protótipo coleta os valores dos parâmetros para as porções de dados identificadas, aplicando a função de análise similar àquela apresentada na figura 6.3, para obter a opção de sincronismo resultante do caminho da árvore de parâmetros. Pode ocorrer a existência de mais de um caminho válido, para aquele conjunto de parâmetros. Neste caso, o protótipo armazena as distintas opções de sincronismo encontradas e os dados ficam armazenados para análise posterior, identificados pelo código da porção e data e horário da coleta. Estas informações constituem um histórico das mudanças ocorridas

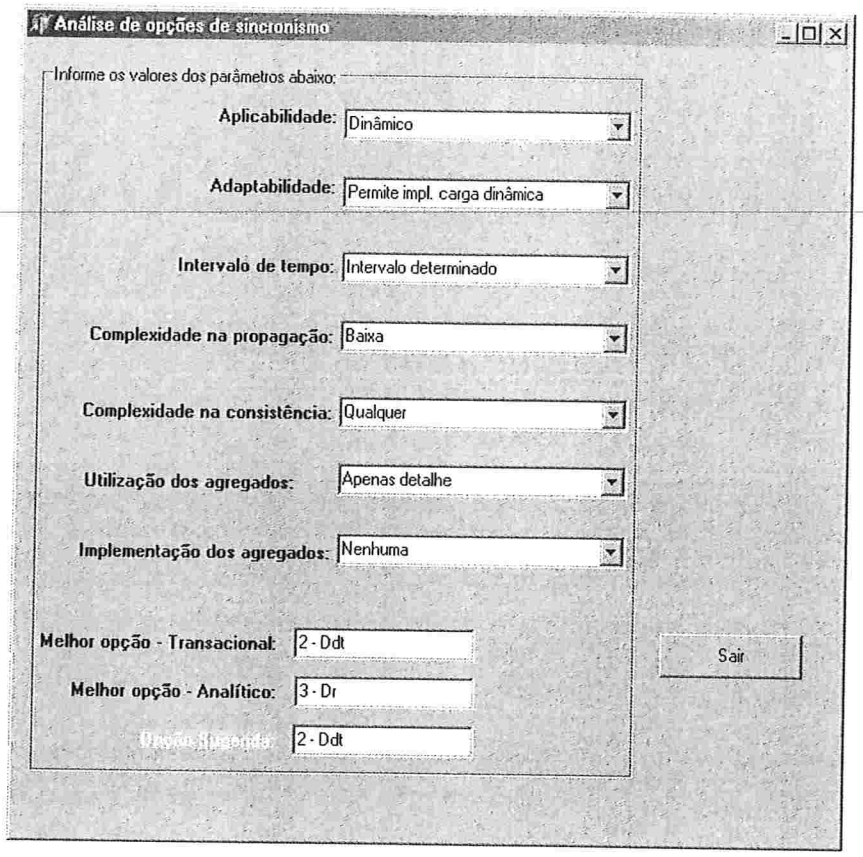


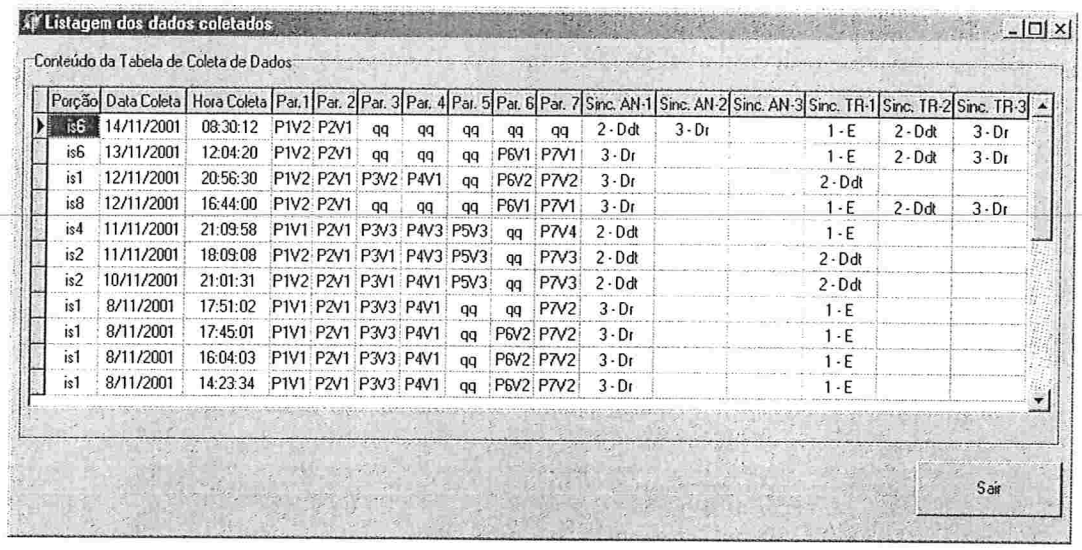
Figura 6.3: **Protótipo** - escolha da opção de sincronismo com base na função de análise

no comportamento de uma determinada porção de dados, possibilitando uma análise detalhada das variações das características dos ambientes transacional e analítico ao longo do tempo.

A forma de armazenamento destas informações pode ser visualizada na figura 6.4.

Para analisar o histórico do comportamento de uma porção, foi, então, desenvolvida uma função que permite visualizar todos os dados coletados para uma determinada porção. Com o auxílio desta função, o responsável pela definição do modelo de sincronismo do data warehouse pode avaliar as variações de comportamento ocorridas em certos intervalos de tempo nos ambientes transacional e analítico e alterar a opção de sincronismo, se for o caso.

Esta função está representada na figura 6.5.



Porção	Data Coleta	Hora Coleta	Par.1	Par. 2	Par. 3	Par. 4	Par. 5	Par. 6	Par. 7	Sinc. AN-1	Sinc. AN-2	Sinc. AN-3	Sinc. TR-1	Sinc. TR-2	Sinc. TR-3
is6	14/11/2001	08:30:12	P1V2	P2V1	qq	qq	qq	qq	qq	2 - Ddt	3 - Dr		1 - E	2 - Ddt	3 - Dr
is6	13/11/2001	12:04:20	P1V2	P2V1	qq	qq	qq	P6V1	P7V1	3 - Dr			1 - E	2 - Ddt	3 - Dr
is1	12/11/2001	20:56:30	P1V2	P2V1	P3V2	P4V1	qq	P6V2	P7V2	3 - Dr			2 - Ddt		
is8	12/11/2001	16:44:00	P1V2	P2V1	qq	qq	qq	P6V1	P7V1	3 - Dr			1 - E	2 - Ddt	3 - Dr
is4	11/11/2001	21:09:58	P1V1	P2V1	P3V3	P4V3	P5V3	qq	P7V4	2 - Ddt			1 - E		
is2	11/11/2001	18:09:08	P1V2	P2V1	P3V1	P4V3	P5V3	qq	P7V3	2 - Ddt			2 - Ddt		
is2	10/11/2001	21:01:31	P1V2	P2V1	P3V1	P4V1	P5V3	qq	P7V3	2 - Ddt			2 - Ddt		
is1	8/11/2001	17:51:02	P1V1	P2V1	P3V3	P4V1	qq	qq	P7V2	3 - Dr			1 - E		
is1	8/11/2001	17:45:01	P1V1	P2V1	P3V3	P4V1	qq	P6V2	P7V2	3 - Dr			1 - E		
is1	8/11/2001	16:04:03	P1V1	P2V1	P3V3	P4V1	qq	P6V2	P7V2	3 - Dr			1 - E		
is1	8/11/2001	14:23:34	P1V1	P2V1	P3V3	P4V1	qq	P6V2	P7V2	3 - Dr			1 - E		

Figura 6.4: **Protótipo** - tabela de coleta de parâmetros para análise de comportamento

6.2 Estudo de caso - a avaliação do data mart para uma área de vendas

Para melhor avaliar a aplicação da proposta feita neste trabalho, optou-se por analisar um caso real. A área de negócios escolhida foi a área de vendas, já que, em uma grande parte das empresas reais, as vendas identificam e orientam o crescimento dos negócios. É claro que o gerenciamento das informações de vendas pode ser feito de várias formas distintas, variando de empresa para empresa. Por esta razão, visando o entendimento do modelo do data mart e avaliação correta dos parâmetros, são descritos inicialmente os requerimentos básicos de negócio, para o exemplo escolhido. O modelo de vendas que será avaliado mais detalhadamente é o modelo de comercialização no varejo, através da Internet. O modelo a ser avaliado é considerado um esquema genérico para *e-commerce*[11].

Para ilustrar e descrever o modelo de negócios de vendas, com o objetivo de se aplicar posteriormente a metodologia que este trabalho sugere, deve-se partir de um produto, uma companhia e um cliente. Um cliente que procura por um produto deve encontrar alguém

[11] Forma de realização de negócios (comércio) *on-line* através da Internet.

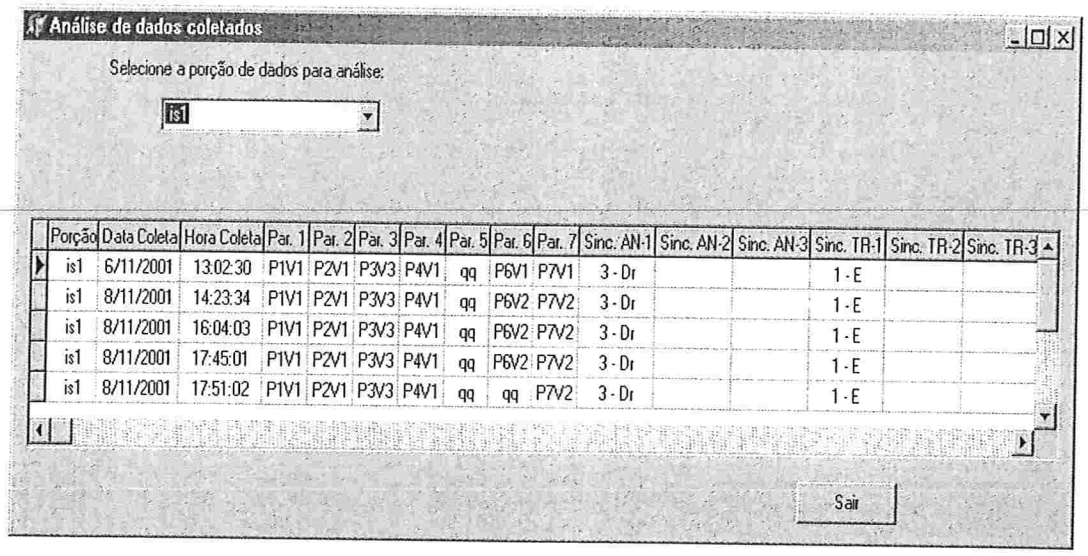


Figura 6.5: **Protótipo** - análise de histórico de comportamento de uma porção de informações

para vendê-lo. Do mesmo modo, um fabricante precisa encontrar canais de venda para o produto. Como consumidores, efetuamos inúmeras compras de produtos a cada ano, porém raramente adquirimos estes produtos diretamente do fabricante. Os comerciantes podem, em muitos casos, adquirir os bens e revendê-los, enquanto que em outros, podem operar simplesmente como agentes de vendas, apenas repassando os produtos ou serviços. É o caso típico de um agente de seguros, que apenas representa diferentes companhias e seus respectivos produtos. O tipo de relacionamento do canal de vendas com o fabricante é tão importante que deve ser considerado para a definição de modelo do data warehouse de vendas.

Em linhas gerais, pode-se considerar alguns modelos de vendas como, por exemplo, as vendas diretas, onde os produtos são vendidos diretamente pelo seu fabricante ao consumidor final e as vendas indiretas, onde os produtos ou serviços são vendidos através de canais externos ao fabricante.

Neste estudo de caso é considerado um modelo para o comerciante, no segmento de varejo, visando o controle da utilização de seus recursos, quando das vendas de produtos

de vários fabricantes através da Internet.

6.2.1 Descrição dos elementos lógicos de vendas para o e-commerce

O ambiente de *e-commerce* requer uma série de novas necessidades já que, diferentemente do ambiente de vendas tradicional, não pode contar com uma força de vendas para acompanhar todo o processo de negócio [SS99]. Como não existe a figura do representante de vendas que possa influenciar o cliente no processo de compra, é preciso encontrar outras formas de exercer esta influência. Isto significa que os negócios a serem realizados através de *e-commerce* requerem mais atenção no que se refere a promoções e propaganda. É importante avaliar freqüentemente o efeito que estes recursos exercem sobre todo o processo de compra, alterando, sempre que necessário, sua utilização.

Este modelo de vendas também requer o acompanhamento da atividade de *clickstream* da interação do visitante em sua navegação pelas páginas do ambiente que envolve a aplicação comercial. A análise deste *clickstream* pode ser comparado à análise das lojas físicas, onde o analista de negócios determina quais são os itens e quais os locais que geram maior volume de vendas se comparados com itens similares localizados em outros pontos de venda. Um exemplo disso é a colocação de determinados produtos em locais de destaque dentro da loja, comparado com a colocação do mesmo produto em uma prateleira comum em um corredor. Existe uma similaridade entra a movimentação de um cliente em uma loja física e a navegação de um cliente em um *website*. Uma aplicação de *e-commerce* deve maximizar a disposição das informações para fornecer um ambiente amigável, de fácil navegação e que oriente os consumidores para os itens mais rentáveis da empresa. Efetuar este acompanhamento é mais difícil que em uma loja física, já que existem muitas possíveis combinações de padrões de navegação através das páginas disponíveis em um *website*.

Neste contexto, todas ocorrências devem estar imediatamente disponíveis, tanto para a aplicação transacional quanto para a aplicação analítica, já que existe uma interação entre ambas em tempo real.

Obviamente, o comerciante está preocupado em saber exatamente quais itens são vendidos e como. Assim, para este modelo de negócios, podemos considerar algumas das perguntas, que devem ser respondidas pelo data mart a ser definido, de acordo com [SS99]:

- Qual é o histórico ou padrão de compras dos usuários?
- Qual tipo de opção de pagamento é mais comum?
- Qual é a demanda dos 5 itens mais vendidos, a cada época do ano e por localização?
- Lista de vendas por grupos de produtos, ordenado por endereço IP.
- Existe correlação entre pedidos e compras de vários produtos?
- Quantos fabricantes diferentes estão, tipicamente, em um carrinho de compras de um consumidor?
- Quanto um determinado fabricante atrai um grupo sócio-econômico?
- Desde o último ajuste de preços, quais produtos tiveram suas vendas aumentadas e quais tiveram suas vendas diminuídas?
- Que tipos de padrões de navegação resultam na maioria das vendas?
- Em que horário do dia ocorre um pico de acessos?
- Quão freqüentemente os compradores procuram por informações detalhadas dos produtos?
- Quais são as 10 páginas mais procuradas? Por tipo, final de semana, mês, etc.
- Quanto tempo é gasto nas páginas com *banners* e sem *banners*?
- Produtos apresentados com imagens e descrições detalhadas vendem melhor que aqueles sem imagens?
- Qual a freqüência de visitantes provenientes de outros *sites* através de anúncios? E qual a freqüência de compras destes visitantes?

- Os itens destacados na página principal vendem mais?
- Quais são as maiores reclamações sobre o *web site*?
- Após uma promoção de 10% de desconto, qual é o aumento de vendas para os produtos?
- Os incentivos de vendas "por tempo limitado"aumentam as vendas?
- Qual a média de tempo entre o pedido e a data de envio? Existem variações em função dos produtos?

As questões apresentadas acima representam uma pequena parte da grande variedade de perguntas que podem ser respondidas pelo data mart para uma aplicação de *e-commerce*.

6.2.2 O data mart para o modelo de e-commerce

A primeira etapa para a definição do data mart consiste em avaliar quais são as necessidades que um comerciante, em um modelo de negócios de *e-commerce* deve ter, no que se refere à uma aplicação analítica. As questões apresentadas na seção anterior evidenciam algumas das características necessárias para a definição de um data mart que atenda aos requisitos de análise do ambiente de *e-commerce*. Com o objetivo de identificar as dimensões e hierarquias do data mart, algumas destas perguntas são analisadas mais detalhadamente:

- Qual é a demanda dos 5 itens mais vendidos, a cada época do ano e por localização?
  Requisitos da consulta:
  - descrição dos itens de produtos;
  - agregação das vendas (quantidade, preço, desconto etc.) por época do ano, que pode ser mês, trimestre, semestre e estação;
  - agregação das vendas por localização do cliente, que pode ser por bairro, cidade, estado, região e país.

- Quanto um determinado fabricante atrai um grupo sócio-econômico?
  Requisitos da consulta:
  - agregação dos itens vendidos por fabricante;
  - agregação das vendas por dados demográficos do cliente, como nível de escolaridade e salarial.

- Desde o último ajuste de preços, quais produtos tiveram suas vendas aumentadas e quais tiveram suas vendas diminuídas?
  Requisitos da consulta:
  - comparação dos valores de vendas entre intervalos de tempo, que podem ser agregadas em determinada quantidade de dias antes e depois do ajuste de preços;
  - descrição dos itens de produtos.

- Em que horário do dia ocorre um pico de acessos?
  Requisitos da consulta:
  - informações das páginas acessadas agregadas por hora;
  - comparativos em determinados intervalo de tempo, como mês, indicador de fim de semana, indicador de feriado.

- Qual a freqüência de visitantes provenientes de outros *sites* através de anúncios? E qual a freqüência de compras destes visitantes?
  Requisitos da consulta:
  - descrição dos anúncios, classificados por data de início e fim, agregados por tipo, canal e custo;
  - identificação das páginas que contêm os anúncios, agregadas por *website*;
  - quantidade de páginas acessadas por URL de origem;
  - identificação das vendas geradas pelos acessos descritos acima.

Para este data mart, assim como para a maioria das áreas de negócios, é obrigatória a presença da dimensão produto e da dimensão que represente a data da compra (tempo). Além destas, são importantes as dimensões que informem o horário da venda, apresentem características demográficas dos clientes, indiquem as promoções e os itens de propagandas, representem os modos de envio de mercadorias e, sem dúvida, as que armazenem as seqüências de cliques nas páginas disponíveis. [SS99] apresenta um modelo completo e detalhado de um data mart para o modelo de vendas em *e-commerce*.

O primeiro ponto a ser analisado é a granularidade da tabela de fatos. A granularidade determina o nível mais atômico da informação que o data warehouse deve capturar, sendo que, para este modelo, é a transação que vende um item a um cliente em uma determinada data.

Após determinar qual o nível de granularidade, deve-se estabelecer os atributos de cada dimensão e seus níveis hierárquicos para agregações. Esta definição depende dos requisitos da aplicação analítica, como nos exemplos apresentados acima, que servirão de base para a definição das dimensões e tabela fato. A estrutura necessária para responder as consultas relativas a um modelo de *e-commerce* segue abaixo, onde estão identificadas fatos, dimensões e atributos mais importantes e ilustrada de acordo com a figura 6.6:

- **Dimensão Produto**: Chave do produto, descrição, marca, subcategoria, categoria, fabricante, tipo de embalagem, peso, cor, tamanho;
- **Dimensão Data**(tempo): Chave da data, dia, dia da semana, dia no mês, semana no ano, mês, trimestre, semestre, ano, ano fiscal, mês fiscal, indicador de feriado, indicador de fim de semana, estação, evento associado;
- **Dimensão Horário**: Chave do horário, hora formato 24h, indicador de manhã, indicador de horário de almoço, indicador de tarde, indicador de horário de jantar, indicador de noite;
- **Dimensão Cliente**: Chave do cliente, nome completo, primeiro nome, sobrenome, sexo, estado civil, data de nascimento, nível de escolaridade, nível salarial, in-

formações sobre pagamento (nome, endereço, etc.), informações sobre envio (nome, endereço, etc.), telefone, e-mail, forma de pagamento, situação de crédito, data do primeiro pedido;

- **Dimensão Promoção**: Chave da promoção, nome da promoção, tipo, nome da mídia da propaganda, condição de preço, condição de anúncio, condição de *displays*, data de início, data de fim;

- **Dimensão Anúncio**: Chave do anúncio, nome, tipo, canal, custo, data de início, data de fim;

- **Dimensão Website**: Chave do *website*, nome da *webpage*, URL, descrição, tipo, *webmaster*, *designer*, tipo de navegação, tipo de *banner*, nome do *banner*, número de produtos na página, indicador de descrição resumida, indicador de imagem, localização da imagem;

- **Dimensão Navegação**: Chave da navegação, URL de origem, URL da página de entrada, URL do primeiro *link*, URL do segundo *link*, URL do terceiro *link*, ..., URL do décimo quinto *link*, URL de saída, indicador de acesso à página de busca, indicador de acesso à página de ajuda;

- **Dimensão Modo de Envio**: Chave do modo de envio, nome, descrição, tipo, tipo de transporte, nome da transportadora, endereço da transportadora, nome do contato, telefone do contato.

- **Fatos**: Chave do produto, chave da data, chave do horário, chave do cliente, chave da promoção, chave do anúncio, chave do *website*, chave da navegação, chave do modo de envio, número do pedido, número do item na linha do pedido, quantidade do item, valor do item, valor do desconto do item, preço médio dos itens, desconto médio dos itens.

Este conjunto de tabelas é considerado uma porção de informações para a aplicação da função de análise, descrita na seção 5.5. Devem ser considerados também, para a

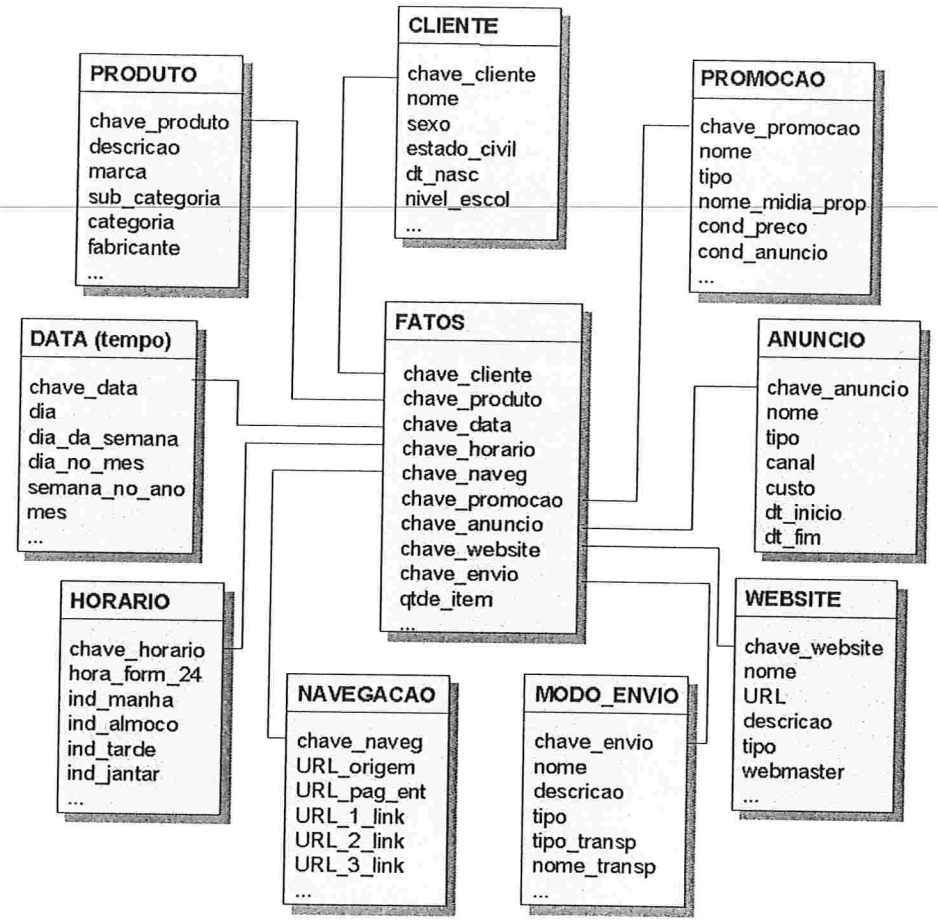


Figura 6.6: Esquema para um modelo de *e-commerce*

definição dos parâmetros e análise, os sistemas transacionais que deram origem a estas informações.

[SS99] analisa as opções de agregação do modelo analítico, baseando-se nas questões que devem ser respondidas pelo data mart e analisando as cardinalidades de cada nível dentro das hierarquias, além da quantidade de elementos vazios. [SS99] propõe um modelo de tabelas de agregação para os fatos e para alguns níveis da hierarquia de algumas dimensões como, por exemplo, mês, marca, promoção, cidade do cliente e modo de envio. Cada tabela de agregação contém um subconjunto da dimensão original ou base. Portanto, para o modelo analítico, foi considerado que o data warehouse está implementado com visões materializadas com agregados, já que existe a necessidade de se emitir consultas

que envolvam vários níveis de agregação, o que pode ser constatado através das questões apresentadas, e o volume de linhas na tabela fato, dentro de alguns meses, pode ser bem grande.

6.2.3 A avaliação os parâmetros e a aplicação da função de análise

Para a porção de informações que representa o modelo de *e-commerce*, apresentado nas seções 6.2.1 e 6.2.2, cada um dos parâmetros, definidos no capítulo 5, é agora analisado. É importante ressaltar que os valores resultantes de cada parâmetro pode mudar ao longo do tempo. Esta seria, portanto, uma avaliação do modelo de negócios inicial, levando em conta uma situação fictícia. Esta avaliação é aplicável para um conjunto considerável de modelos reais de negócio.

- **P1 - Aplicabilidade** - Para este parâmetro deve-se avaliar a necessidade da manutenção dinâmica que o modelo de negócios apresenta. O modelo de vendas de *e-commerce*, para o tipo de negócio apresentado, normalmente é dinâmico, sendo que a informação deve ser trabalhada o mais rápido possível. Pode existir um requerimento de se analisar as vendas efetuadas, por exemplo, nas horas seguintes a uma determinada promoção, ou então, o resultado da colocação de anúncios em sites de busca ou portais de informação, como parte das atividades de uma determinada campanha de marketing. Dentro destas condições, o valor para este parâmetro é *dinâmico*. Devemos considerar, porém, que isto não é válido para todos os modelos de negócios em *e-commerce*. Por não tratarmos aqui de um caso real específico, é necessário avaliar também a possibilidade de que alguns modelos de negócios não requerem que os dados estejam disponíveis imediatamente ou em curtos intervalos de tempo. Para algumas empresas, a atualização do data warehouse pode ser efetuada de forma estática, através de cargas periódicas. Assim, foi considerado que a opção *tipicamente estático* também pode ser válida para determinados modelos de *e-commerce*.

- **P2 - Adaptabilidade** - Este parâmetro pressupõe a capacidade de se implementar a manutenção dinâmica do data warehouse, a partir dos sistemas transacionais. Em certas implementações, os sistemas ou as bases de dados transacionais permitem a inclusão de rotinas para a atualização dinâmica do data warehouse. Quando este for o caso, o valor para este parâmetro é *permite a implementação da carga dinâmica.* Pode ocorrer, porém, que, devido à escolha tecnológica para a implantação do sistema comercial que trata as vendas por Internet e o data warehouse, não exista uma forma de se transferir dinamicamente as informações do ambiente transacional para o analítico. Um exemplo disso é o caso em que a plataforma onde está implementado o sistema transacional do *e-commerce* está completamente desconectada da plataforma onde se encontra o data warehouse. Portanto, a definição do valor deste parâmetro depende da implementação do sistema transacional. Neste caso, deve-se também considerar a possibilidade deste parâmetro receber o valor *não permite.*

- **P3 - Intervalo de tempo** - O modelo de negócios apresentado pode ser implementado com um sincronismo imediato, ou seja, em tempo real. Para alguns modelos de negócios mais dinâmicos é importante que a base analítica receba as informações sobre as vendas assim que forem fechadas. O modelo escolhido para este estudo de caso pressupõe que a quantidade de vendas efetuadas no varejo, via Internet, com itens de diferentes fornecedores, seja grande e realizada 24 horas por dia. Para os modelos de negócios mais dinâmicos e que requerem a atualização do data warehouse no menor intervalo de tempo possível após a efetivação da venda, o valor deste parâmetro deve ser considerado como *mínimo possível (tempo real).* Conforme a discussão do parâmetro P1, acima, pode ocorrer que este dinamismo não seja necessário, portanto, para estes casos, o parâmetro irá conter o valor *intervalo determinado* ou *sincronismo postergado.* Além disso, existem os casos que, de acordo com a avaliação do parâmetro P2, não permitem nenhuma das opções dinâmicas e requerem, portanto, que o parâmetro P3, intervalo de tempo, contenha obrigatoriamente o valor *sincronismo postergado*, já que a carga do data warehouse é efetuada

estática e periodicamente, após o fechamento diário, semanal ou mensal das vendas.

- **P4 - Complexidade na propagação** - Como as características do sistema transacional deste estudo de caso não foram detalhadas, deve-se considerar as várias opções da complexidade na propagação. Este é um parâmetro que depende exclusivamente do modelo de dados da base transacional e da arquitetura do sistema transacional de *e-commerce* implementado. Por esta razão, os três valores, *baixa*, *média* e *alta* são considerados válidos.

- **P5 - Complexidade na consistência** - Pelas mesmas justificativas discutidas no parâmetro anterior, os valores *baixa*, *média* e *alta* são utilizados para a análise.

- **P6 - Utilização de agregados** - Conforme as questões apresentadas nas seções 6.2.1 e 6.2.2, a aplicação analítica deve utilizar as informações detalhadas, bem como as agregadas. Isto fica claro no trabalho de [SS99], que analisa detalhadamente e define as tabelas de agregação. Assim, o valor considerado para este parâmetro é *agregados*.

- **P7 - Implementação dos agregados** - Da mesma forma que o parâmetro anterior, [SS99] indica a implementação dos agregados através de *visões materializadas com agregados*, devido ao grande volume de informações no data mart e uma destacada preocupação com o desempenho das consultas analíticas.

Conforme a discussão acima, alguns dos parâmetros não possuem apenas um valor válido. Para considerar todas as possibilidades existentes, são apresentadas as combinações destes valores, com o objetivo de analisar as várias alternativas de implementação do modelo de negócios. É importante lembrar que apenas as combinações válidas, de acordo com a discussão apresentada na seção 5.4, estão representadas nas tabelas.

Para indicar qual a melhor opção de sincronismo para as possibilidades apresentadas, cada uma das combinações de parâmetros foi submetida para execução no protótipo que, após aplicar a função de análise, sugere uma das alternativas.

Os resultados obtidos, bem como os parâmetros que formam cada combinação estão representados nas tabelas 6.2, 6.3 e 6.4. Para melhor visualização das combinações e opções de sincronismo resultantes, as tabelas estão divididas e os valores foram representados conforme legenda, na tabela 6.1.

Parâmetro	Representação na tabela	Descrição
P1	E	Tipicamente estático
	D	Dinâmico
P2	P	Permite a implementação da carga dinâmica
	N	Não permite
P3	R	Mínimo possível (tempo real)
	D	Intervalo determinado
	P	Sincronismo postergado (carga estática)
P4	B	Baixa
	M	Média
	A	Alta
P5	B	Baixa
	M	Média
	A	Alta
P6	A	Agregados
P7	M	Visões materializadas com agregados
Opção	**Representação na tabela**	**Descrição**
E	1	Estático com cargas periódicas
Ddt	2	Dinâmico em determinado intervalo de tempo

Tabela 6.1: Legenda das tabelas de combinações de parâmetros

Para um melhor entendimento do conteúdo das tabelas 6.2, 6.3 e 6.4, segue-se uma breve explicação. Na primeira coluna de cada tabela estão os parâmetros a serem avaliados e nas colunas seguintes encontram-se cada uma das combinações válidas para estes

P1	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	D	D	D	D	D	D
P2	P	P	P	P	P	P	P	P	P	N	N	N	N	N	N	N	N	N	P	P	P	P	P	P
P3	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	R	R	R	R	R	R
P4	B	B	B	M	M	M	A	A	A	B	B	B	M	M	M	A	A	A	B	B	B	M	M	M
P5	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A
P6	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
P7	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Opção:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2

Tabela 6.2: Parte A - Avaliação das combinações de parâmetros para o estudo de caso de *e-commerce*

P1	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
P2	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	N	N	N
P3	R	R	R	D	D	D	D	D	D	D	D	D	P	P	P	P	P	P	P	P	P	R	R	R
P4	A	A	A	B	B	B	M	M	M	A	A	A	B	B	B	M	M	M	A	A	A	B	B	B
P5	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A
P6	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
P7	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Opção	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1

Tabela 6.3: Parte B - Avaliação das combinações de parâmetros para o estudo de caso de *e-commerce*

parâmetros. Estas combinações foram submetidas para execução no protótipo que, após receber cada combinação de valores para os parâmetros, retornou a opção de sincronismo mais adequada. Assim, como exemplo, a primeira coluna que contém uma combinação de valores (E, P, P, B, B, A, M) na tabela 6.2, ao ser submetida para execução no protótipo resultou na opção de sincronismo 1, que representa a opção de sincronismo *estático com cargas periódicas*, de acordo com a legenda na tabela 6.1.

P1	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
P2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
P3	R	R	R	R	R	R	D	D	D	D	D	D	D	D	D	P	P	P	P	P	P	P	P	P
P4	M	M	M	A	A	A	B	B	B	M	M	M	A	A	A	B	B	B	M	M	M	A	A	A
P5	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A	B	M	A
P6	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
P7	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Opção:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabela 6.4: Parte C - Avaliação das combinações de parâmetros para o estudo de caso de *e-commerce*

Vale salientar que as outras tabelas (6.3 e 6.4) apresentam os resultados das combinações restantes e podem ser interpretadas da mesma forma.

A execução do protótipo para uma das combinações de parâmetros é apresentada na figura 6.7.

6.3 Conclusão

Este capítulo descreveu a implementação de um protótipo, desenvolvido com o objetivo de validar, de forma prática, os resultados obtidos pela aplicação da função de análise, definida na seção 5.5. Este protótipo foi utilizado para avaliar um estudo de caso envolvendo o modelo de negócios de *e-commerce*. Neste estudo de caso, os parâmetros definidos nas seções 5.2 e 5.3 foram avaliados e submetidos para a execução no protótipo, com o objetivo de se obter a melhor opção de sincronismo para as informações que compõem o data mart de *e-commerce*.

Os resultados apresentados neste estudo levaram à opção de sincronismo estático para algumas das combinações de parâmetros e à opção de sincronismo dinâmico com um determinado intervalo de tempo, para outras.

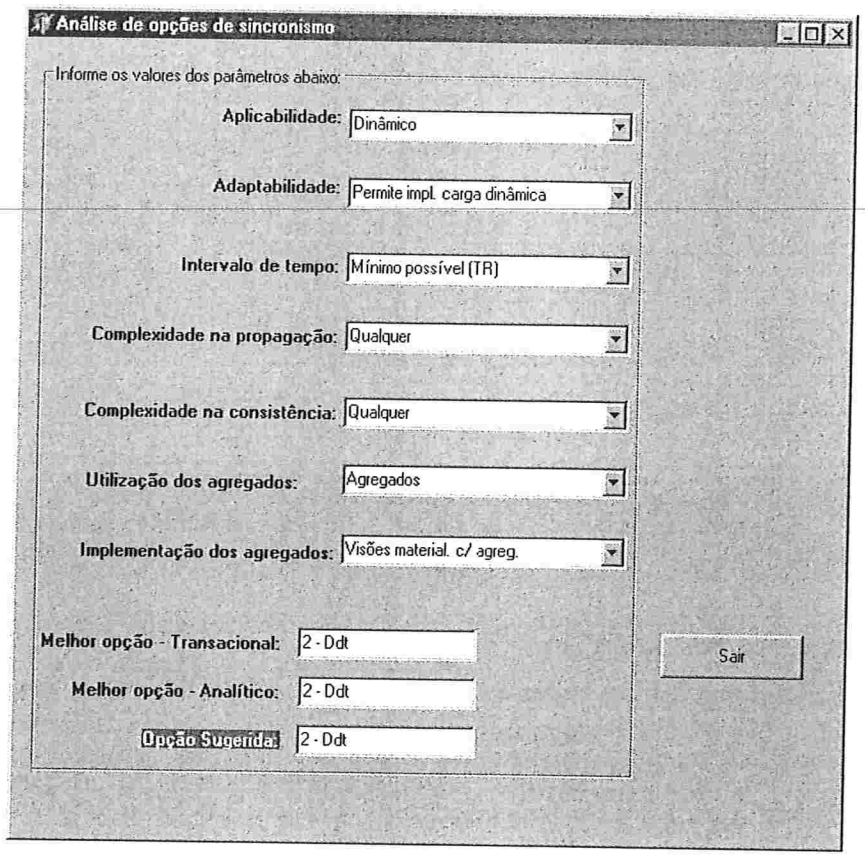


Figura 6.7: Estudo de Caso - escolha da opção de sincronismo para o modelo de *e-commerce*

Para os casos que apresentam como resultado a opção de sincronismo dinâmica com intervalo de tempo, alguns aspectos devem ser analisados quando da definição do tamanho do intervalo de tempo a ser implementado, como por exemplo:

- O volume de transações ocorridas em um determinado intervalo de tempo;
- O volume de dados a ser transportado;
- A disponibilidade dos meios de transporte entre os ambientes transacional e analítico;
- A rapidez na transferência das informações;
- A velocidade que o modelo de negócios requer.

Todos estes aspectos dependem do modelo de negócios específico que, com suas variações, pode levar à definição de diferentes intervalos de tempo para a manutenção do sincronismo. Outro aspecto importante envolve os parâmetros que avaliam a complexidade no ambiente transacional. É certo que quanto menor a complexidade do sistema transacional, menor pode ser o intervalo de tempo para o sincronismo.

Capítulo 7

Conclusão

7.1 A relevância do problema

Neste trabalho apresentamos uma solução personalizada para os processos diferenciados de atualização de dados entre os ambientes transacionais e analíticos. Partimos do pressuposto de que tanto os processos de atualização de dados como as necessidades de disponibilidade dos dados não são homogêneas em um projeto de data warehouse.

Algumas áreas de negócios requerem processos de decisão cada vez mais rápidos e precisos e, consequentemente, o modelo de sincronsimo entre o ambiente transacional e o analítico, normalmente implementado através de cargas periódicas se torna um fator crítico. Por outro lado, alguns modelos de negócios requerem sincronismo dinâmico - mantendo os ambientes sincronizados em tempo quase real ou em intervalos de tempo menores que aqueles implementados pelas cargas estáticas, executadas periodicamente. O objetivo deste trabalho foi o de definir a opção de sicronismo mais adequada para um determinado modelo de negócios, o que se reflete em cada uma das porções do data warehouse. Esta análise é baseada em uma função que avalia um conjunto de parâmetros, com o objetivo de definir qual a opção de sincronismo mais adequada para cada uma das porções.

Com isto, o data warehouse passa a implementar um modelo de sincronismo híbrido,

onde as porções de dados podem ser sincronizadas em intervalos de tempo diferentes, dependendo das características do ambiente transacional que deu origem às informações e dos requisitos de utilização da aplicação analítica. Esta análise, portanto, resulta em uma classificação de domínio. Assim propomos, nesse trabalho, um modelo híbrido de sincronismo de dados para os projetos de data warehouse. Para viabilizar tal hipótese, desenvolvemos diretrizes para classificação das transações tanto nos ambientes transacionais como analíticos. Estas diretrizes resultaram na definição de parâmetros de acordo com a seção 5.4 e seu respectivo formalismo foi apresentado na seção 5.5. Para validar tal formalismo foi desenvolvido um protótipo, apresentado no capítulo 6.

7.2 As contribuições do trabalho

Os resultados desse trabalho mostram que é viável a existência de um modelo híbrido de dados para projetos em data warehouse. Tal viabilidade torna os processos de atualização de dados mais eficientes. Isso contribui para uma diminuição do tempo de sincronismo entre os sistemas operacionais e de data warehouse e ainda uma diminuição dos exagerados recursos de hardware necessários para manter um sicronismo aceitável de dados dos ambientes em questão.

7.2.1 A classificação dos parâmetros

Para facilitar a avaliação da opção de sincronismo mais adequada para cada uma das porções do data warehouse, este trabalho apresentou as características dos ambientes transacional e analítico que devem ser analisadas.

Como características importantes do ambiente transacional, definimos a **necessidade da manutenção dinâmica** baseado no modelo de negócios que gera as informações daquela porção do data warehouse, identificando que algumas aplicações são tipicamente estáticas, no que se refere à carga de dados, enquanto outras requerem, por características funcionais, a atualização dinâmica da porção.

Outro aspecto avaliado foi sobre a **capacidade de adaptação do ambiente transacional**, que analisa a possibilidade de se implementar os processos necessários para sincronizar dinamicamente o data warehouse. Destacamos que algumas aplicações transacionais não possuem recursos que permitam a implementação destes processos, enquanto que outras possibilitam a inserção dos requisitos necessários para se obter e propagar as informações ao data warehouse.

Foi também avaliado o **intervalo de tempo desejável para o sincronismo**. Neste aspecto, o trabalho mostrou que deve-se avaliar a real necessidade do sincronismo dinâmico, em tempo real ou com um intervalo de tempo, já que o custo gerado por este tipo de implementação pode ser alto. Identificamos que algumas aplicações requerem o sincronismo do data warehouse em tempo real, ou no menor tempo possível, algumas se caracterizam por requerer um sincronismo em determinado período de tempo enquanto que outras necessitam manter o sincronismo em intervalos de tempo regulares, através de uma carga estática e periódica.

A **complexidade no ambiente transacional**, dividida em **complexidade na propagação** e **complexidade na consistência** é um fator que também deve ser avaliado. A complexidade na propagação se refere ao número de objetos distintos envolvidos na transação para realizar a atualização das informações do data warehouse. A complexidade na consistência analisa o nível de complexidade das verificações e validações que devem ser executadas pela transação imediatamente após a atualização na base transacional, para disponibilizar as informações para o processo de carga dinâmica. Estas complexidades foram consideradas em níveis altos, médios ou baixos, de acordo com as características da aplicação transacional avaliada.

O ambiente analítico se caracteriza pelas consultas efetuadas ao data warehouse. Estas consultas podem envolver apenas níveis de detalhe da informação ou incluir informações agregadas, em diferentes níveis. A existência de informações pré-agregadas na base analítica favorece o desempenho das consultas, porém dificulta o processo para a manutenção dinâmica do data warehouse, em função da forma como estes agregados são implementados. Para avaliar os aspectos do ambiente analítico, neste trabalho, definimos

dois parâmetros relacionados à agregação das informações.

O primeiro parâmetro está relacionado à **utilização dos agregados** nas consultas analíticas, que podem ser classificadas segundo este critério. Algumas consultas não utilizam utilizam agregados, buscando somente as informações em seu nível de granularidade mais baixo. Outras consultas fornecem informações agregadas ao usuário e, portanto, utilizam as pré-agregações disponíveis.

O segundo parâmetro descreve a **forma de implementação dos agregados** na base analítica e indica as várias alternativas para a implementação das agregações. A base analítica pode não conter pré-agregação das informações ou pode implementar estas agregações como visões não materializadas, visões materializadas ou cubos.

A avaliação, em conjunto, destes parâmetros indica a opção de sincronismo mais adequada para cada uma das porções de informações do data warehouse.

7.2.2 O formalismo da função de análise dos parâmetros

As características dos ambientes transacionais e analíticos, indicadas na seção anterior, foram consideradas como parâmetros e a interferência de um parâmetro no outro foi avaliada, através de uma função, definida na seção 5.4 deste trabalho. Para melhor clareza da análise, apresentamos os parâmetros sob a forma de árvores de decisão, onde cada folha da árvore representa o modelo de sincronsimo proposto para a porção do data warehouse.

Definimos, como opções de sincronismo, as alternativas de sincronismo em tempo real e sincronismo em determinado intervalo de tempo para o modelo dinâmico e, para o modelo estático, o sincronismo postergado, com cargas periódicas.

A função que avalia estes parâmetros em conjunto foi definida por:

$$\gamma(\mathbf{P(D)},\mathbf{A}) \rightarrow \mathbf{P(B)}$$

onde:

- $\mathbf{P(D)}$ representa o conjunto que contém as porções de informação escolhidas para o processo de análise;

- **A** é um conjunto de combinações de valores válidos para os parâmetros P1, P2, P3, P4, P5, P6 e P7 e
- **P(B)** é o conjunto de valores que representam os possíveis níveis de sincronismo.

Sendo X $\subset$ P(D), Y $\subset$ P(B) e (p_1, p_2, p_3, p_4, p_5, p_6, p_7) $\in$ A, temos:

$$\gamma(X, p_1, p_2, p_3, p_4, p_5, p_6, p_7) \rightarrow Y$$

Definimos que esta função γ deve ser aplicada para todos os subconjuntos de **P(D)**, ou seja, para todas as porções de informações do data warehouse, de forma a poder buscar a opção de sincronismo mais adequada para cada uma das porções.

Com isto, mostramos a viabilidade de um modelo híbrido para o data warehouse, onde as porções apresentam diferentes opções de sincronismo, escolhido com o objetivo de melhor utilização dos recursos e melhor desempenho dos processos de carga.

7.2.3 O protótipo para validação da proposta

Para validar a função de análise dos parâmetros desenvolvemos um protótipo que permitiu que situações do ambiente real fossem analisadas. Este protótipo tem como principais funções:

- Escolher a opção de sincronismo mais adequada para uma porção, dados os valores para os parâmetros definidos na seção 5.4;
- Armazenar o histórico de comportamento de cada porção do data warehouse, através da coleta periódica dos valores dos parâmetros e
- Permitir a simulação da avaliação deste histórico.

A função de análise dos parâmetros foi implementada em uma rotina que permite a entrada dos valores dos parâmetros, indicando a melhor opção de sincronismo para aquele conjunto de parâmetros.

A decisão de se implementar rotinas para acompanhar o comportamento de uma porção do data warehouse partiu da premissa de que a opção de sincronismo mais adequada para a porção de dados pode mudar ao longo do tempo, já que as características do modelo de negócios, da implementação do sistema transacional e a utilização do data warehouse através das consultas dos usuários também se alteram ao longo do tempo. A análise do histórico de comportamento de uma porção de dados permite analisar as opções escolhidas de acordo com o comportamento da porção ao longo do tempo e identificar ocorrência de uma mudança no modelo de sincronismo mais adequado. Esta identificação pode levar a uma possível alteração no modelo já implementado, buscando sempre melhor desempenho e melhor utilização dos recursos nos processos de carga.

Este protótipo tem como objetivo ser uma ferramenta de auxílio na definição do modelo do data warehouse, no que se refere aos aspectos de sincronismo.

7.3 Futuras pesquisas

Os próximos passos do trabalho podem ser divididos em quatro frentes:

1. *Incorporação do protótipo desenvolvido nesse trabalho pelas ferramentas de transferência de dados disponíveis no mercado*

   O protótipo apresentado mostrou a viabilidade de um componente de software para captar e armazenar as infomações das porções de dados com seus respectivos modos de atualização . Esse protótipo uma vez implementado nas ferramentas de transferência de dados disponíveis no mercado deverá fornecer valiosos dados históricos para melhor classificar o comportamento das formas de atualização entre os ambientes transacional e analítico. Essa ferramenta poderá contribuir para o conceito de sistemas adaptativos em algoritmos de sincronismo entre os ambientes transacional e analítico.

2. *Avaliação e extensão do modelo híbrido de dados para bases de dados heterogêneas*

   Considerando um ambiente de bases de dados heterogêneas como fonte de dados para

o data warehouse, temos uma diminuição da eficiência dos processos de carga em função das dificuldades de conversão de tipos de dados, formato e valor semântico. Uma próxima investigação possível é o de avaliar o impacto do ambiente heterogêneo de base de dados no modelo híbrido de sincronismo de dados proposto nesse trabalho.

3. *Elaboração de uma metodologia para utilização de modelos híbridos em projetos de data warehouse*

   Nesse trabalho apresentou-se diretrizes para elaboração de um modelo híbrido de dados. Entretanto para tornar tal proposta mais fundamentada é possível criar uma metodologia para tal modelo híbrido com utilização de técnicas de especificação formal de engenharia de software.

4. *Extensão da base de domínios dos parâmetros avaliados para uma base de dados fuzzy*

   Considerando uma base de dados fuzzy como uma extensão do esquema relacional, relações fuzzy podem ser utilizadas para armazenar tuplas que contenham, além dos valores dos parâmetros, valores ou pesos que indicam a probabilidade e o grau de incerteza da informação coletada. Estas probabilidades e graus de incerteza permitirão a transformação da informação em um valor numérico que será utilizado para calcular matematicamente a melhor opção de sincronismo para uma determinada porção de informações.

Referências Bibliográficas

[AAD+96] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. *On the computation of multidimensional aggregates.* Proceedings: 22nd VLDB, pages 506–521, Mumbai, Set 1996.

[Alb99] J. Albrecht e outros. *Management of multidimensional aggregates for efficient Online Analytical Processing.* Proceedings: International Database Engineering and Applications Symposium (IDEAS'99), 1999.

[AV98] C. Adamson, M. Venerable. *Data Warehouse Design Solutions*, John Wiley and Sons, 1998.

[BCL89] J.A. Blakeley, N. Coburnm, P. Larson. "Updating derived relations: detecting irrelevant and autonomously computable updates". *ACM Transactions on Database Systems*, 14(3):369-400, 1989.

[BLT86] J.A. Blakeley, P. Larson, F. Tompa. *Efficiently updating materialized views.* SIGMOD, 1986..

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

[CW91] S. Ceri, J. Widom. *Deriving incremental production rules for incremental view maintenance.* 17th VLDB, 1991.

[DEB95] IEEE Data Engineering Bulletin, *Special Issue on Materialized Views and Data Warehousing*, 18(2), junho 1995.

[DRSN98] P. M. Deshpande, K. Ramaswamy, A. Shukla, and J. F. Naughton. *Caching multidimensional queries using chunks.* Proceedings: ACM SIGMOD Conference on Management of Data, Seattle, WA, 1998.

[Elk90] C. Elkan."Independence of logic database queries and updates". *9th PODS*, pág. 154-160, 1990.

[GB95] A. Gupta, J. A. Blakeley. *Maintaining views using materialized views.* Documento não publicado.

[GBLP96] Jim Gray, Adam Bosworth, Andrew Layman and Hamid Pirahesh. *Data Cube: A Relational Operator Generalizing Group-By, CrossTab and Sub-Totals.* Proceedings: 12th Int. Conf. on Data Engineering, pp 152–159, 1996.

[GHRU96] H. Gupta, V. Harinarayan, A. Rajaraman, J. Ullman. *Index selection in OLAP.* 1996.

[GJM94] A. Gupta, H. V, Jagadish, I. S. Mumick. *Data integration using self-maintainable views.* Techical memorandum, AT&T Bell Laboratories, 1994.

[GM95] A. Gupta, I. S. Mumick. *Maintenance of materialized views: problems, techniques, and applications.* 1995.

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

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

[GMS93] A. Gupta, I. S. Mumick, V. S. Subrahmanian. *Maintaining views incrementally.* 1993

[Gup94] A. Gupta. *Partial information based integrity constraint checking.* Tese de doutoramento da Stanford University (CS-TR-95-1534).

[Gup97] H. Gupta. *Selection of views to materialize in a Data Warehouse*, 1997.

[HD92] J. Harrison, S. Dietrich. "Maintenance of materialized views in a deductive database: an update propagation approach". *Workshop on Deductive Databases*, JICSLP, 1992.

[HRU96] V. Harinarayan, A. Rajaraman, J. Ullman. "Implementing data cubes efficiently". *ACM Sigmod Intl. Conf. on Mngt. of Data*, 1996.

[IA99] *Information Advantage, Decision Path^{TM} Implementation Methodology.* "MyEureka!^{TM} data warehouse requirements guide", 1999.

[IH97] W. H. Inmon, R. D. Hackathorn. *Como usar o Data Warehouse.* Infobook, Rio de Janeiro, 1997.

[JMS95] H. V. Jagadish, I. S. Mumick, A. Silberschatz. "View maintenance issues in the chronicle data model". *14th PODS*, pág. 113-124, 1995.

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

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

[Kuc91] V. Kuechenhoff. "On the efficient computation of the difference between consecutive database states". *International Conference on DOOD*, 1991.

[LS93] A.Y. Levy, Y. Sagiv. "Queries independent of updates". *19th VLDB*, pág. 171-181, 1993.

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

[MQM97] I. Mumick, D. Quass and B. Mumick, *Maintenance of Data Cubes and Summary Tables in a Warehouse.* Proceedings: ACM SIGMOD Conf. on Management of Data, Arizona, Maio 1997.

[Pai84] R. Paige. "Applications of finite differencing to database integrity control and query/transaction optimization". *Advances in Database Theory*, pág. 170-209, Plenum Press, New York, 1984.

[PJD99] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. *Extending Practical Pre-Aggregation in On-Line Analytical Processing.* Proceedings: VLDB, pp. 663–674, 1999.

[QGMW97] D. Quass, A. Gupta, I. S. Mumick, J. Widom. *Making views self-maintainable for data warehousing.* 1997.

[Qua97] D. Quass. *Materialized views in data warehouses.* Tese de doutoramento do Departamento de Ciência da Computação da Stanford University. Agosto 1997.

[QW97] D. Quass, J. Widom. "On-line Warehouse View Maintenace". *Proceedings of ACM SIGMOD 1997 - International Conference on Management of Data*, pág. 393-404, 1997.

[QW91] X. Qian, G. Wiederhold. "Incremental recomputation of active relational expressions". *IEEE TKDE*, 3(1991), pág. 337-341.

[Rad96] N. Raden. *Modeling the Data Warehouse*, 1996.

[SBHD99] C. Sapia, M. Blaschka, G. Höfling, B. Dinter. "Extending the E/R model for the multidimensional paradigm". *Advances in database technologies - Journal of Computer Science and Information Management*, vol. 2, N. 3, 1999.

[SI84] O. Shmueli, A. Itai. *Maintenance of views. Sigmod Record*, 14(2):240-255, 1984.

[SMKK98] S. Samtani, M. Mohania, V. Kumar, Y. Kambayashi. *ER Workshops*, 1009, pág. 81-92.

[SS99] I. Song, K. LeVan-Shultz. "Data warehouse design for e-commerce environments". *Proceedings of ER Workshops*, pág. 374-387, 1999.

[Tan97] R. Tanler. *The Intranet Data Warehouse*, John Wiley and Sons, 1997.

[UO92] T. Urpi, A. Olive. "A method for change computation in deductive databases". *Proceedings of the Eighteenth International Conference on Very large Data Bases*, pág. 225-237, Vancouver, British Columbia, 1992.

[UW97] Jeffrey D. Ullman, Jennifer Widom. *A first course in database systems.* Prentice Hall, U.S.A, 1997.

[YKL96] J. Yang, K. Karlapalem, Q. Li. *A framework for designing materialized views in data warehousing environment.* Technical Report HKUST-CS96-35. Outubro 1996.

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