UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

A MODELAGEM DE DADOS NO AMBIENTE DATA WAREHOUSE

Daniele Del Bianco Hokama
Denis Camargo
Francine Fujita
João Luiz Valentim Fogliene

São Paulo
2004

Daniele Del Bianco Hokama
Denis Camargo
Francine Fujita
João Luiz Valentim Fogliene

A MODELAGEM DE DADOS NO AMBIENTE DATA WAREHOUSE

Trabalho de Graduação Interdisciplinar apresentado à Faculdade de Computação e Informática, da Universidade Presbiteriana Mackenzie, como exigência parcial para a obtenção do grau de Bacharel em Sistemas de Informação

Orientador: Prof. ROGÉRIO OLIVEIRA

São Paulo
2004

SUMÁRIO

LISTA DE QUADROS

LISTA DE FIGURAS

RESUMO

Modelos multidimensionais são largamente utilizados em processos analíticos que requerem complexos cruzamentos de informações, pois são capazes de processar com performance razoável um grande volume de dados. Este trabalho busca apresentar as principais técnicas da modelagem multidimensional utilizadas hoje. Essas técnicas têm o objetivo de otimizar a estrutura dos modelos multidimensionais. Para tanto, foi empregado um método baseado na apresentação de exemplos e na discussão dos problemas do modelo relacional para atender a demanda de informações analíticas e gerenciais.

ABSTRACT

Multidimensional models are wide used in analytical processes that require complexes data crosses because they can compute with moderate performance large amounts of data. This job presents the major multidimensional modeling techniques applied nowadays. These techniques have the objective to optimize multidimensional modeling structures. In order to do that, it was used a method based on the presentation of examples and on the discussion of the relational model's problems to supply the analytical and the business information demand.

INTRODUÇÃO

Quando surgiram, os sistemas e aplicações voltados para as funções operacionais tornaram o trabalho mais simples, pois, além de garantirem maior agilidade na execução de tarefas que antes requeriam um esforço muito grande, também minimizam possíveis erros humanos.

Além de vantagens como as citadas acima, essas aplicações trouxeram outras indiretamente, como a capacidade de manter informações históricas de forma agrupada e possível de serem consultadas. Essas informações poderiam ser usadas por áreas estratégicas da empresa (marketing, alta gerência, etc) para auxiliar em tomadas de decisão. Porém, agrupar essas informações, interpretá-las e tirar conclusões não é uma tarefa fácil. É preciso extrair de cada base de dados as informações que realmente interessam e padronizá-las para que possam ser analisadas.

O processo de *data warehousing* busca automatizar o processo de extração e padronização de dados, além de prover ao usuário maneiras mais fáceis e flexíveis de visualizar os dados. Um ***data warehouse*** é uma coleção de dados orientada por assuntos, integrada, variante no tempo, e não volátil, que tem por objetivo dar suporte aos processos de tomada de decisão [INMON, 1999].

De uma forma geral, sistemas de *data warehouse* compreendem um conjunto de programas que extraem e tratam dados do ambiente operacional da empresa, um banco de dados que os mantém, e sistemas que fornecem estes dados aos seus usuários, dando suporte a consultas *ad-hoc* (consultas com acesso casual único e tratamento dos dados segundo parâmetros nunca antes utilizados), relatórios analíticos e à tomada de decisão.

Com essas características voltadas à análise de negócios, é natural que o ambiente de *data warehouse* requeira mudanças constantes em seus relatórios e consultas. Essa flexibilidade é imprescindível, uma vez que ter as informações certas pode fazer a diferença na tomada de decisão. Porém, embora as necessidades por informações específicas mudem com freqüência, os dados associados não mudam. Imaginando-se que os processos de negócio de uma empresa permaneçam relativamente constantes, existe apenas um número finito de objetos e eventos com as quais uma organização está envolvida. Por esta razão, um modelo de dados é uma base sólida para identificar requisitos para um *data warehouse*.

Um modelo de dados bem estruturado é capaz de prover às empresas a capacidade de extraírem as informações certas das mais diferentes formas e maneiras, independente da ferramenta ou do grau de complexidade exigido nas consultas. Ou seja, a importância da modelagem de dados em *data warehouse* reside no fato de que, sem uma estrutura bem elaborada, a enorme quantidade de informações pode tornar as consultas demasiado lentas, e com a possibilidade de tornar inviáveis algumas operações de consulta.

O objetivo geral desse trabalho é mostrar a importância da modelagem de dados no ambiente *data warehouse,* apresentar os conceitos da modelagem dimensional e oferecer técnicas interessantes de construção de um modelo dimensional, visando oferecer ao usuário informações interessantes para a realização de consultas analíticas, e buscando a otimização da performance das consultas e das atualizações na base de dados.

O trabalho encontra-se organizado como segue: o capítulo um traz informações acerca da modelagem de dados relacional como uma base de entendimento para os próximos temas. Trata de informações acerca do ambiente *data warehouse*, seus termos e conceitos, e introdução sobre o “modelo dimensional de dados”, que será discutido mais profundamente

no capítulo dois, que dedica-se a modelagem dimensional. Os conceitos de tabela de fatos, o esquema estrela e floco de neve serão discutidos neste capítulo em detalhe. O capítulo três trata das técnicas de modelagem a serem exploradas na construção de um modelo dimensional, e discutem-se aspectos como a performance do modelo, a criação de chaves, o uso de agregações e a redundância controlada de dados. O capítulo quatro apresenta estratégias que podem ser adotadas para reduzir o tempo de processamento para recuperar e atualizar as informações contidas em um modelo dimensional. Um apêndice foi incluído para discutir o uso do OLAP (Processamento Analítico On-line), comparando os padrões ROLAP e MOLAP, contribuindo para um maior aprofundamento dos temas discutidos.

Capítulo 1 – Ambiente *Data Warehouse*

1.1 Conceitos

Um ***data warehouse*** nada mais é do que um banco de dados contendo dados extraídos do ambiente de produção da empresa, que foram selecionados e depurados, tendo sido otimizados para processamento de consulta e não para processamento de transações. Em geral, um *data warehouse* requer a consolidação de outros recursos de dados, além dos armazenados em banco de dados relacionais, como informações provenientes de planilhas eletrônicas, documentos textuais, etc. [INMON, 1999].

É importante considerar que um *data warehouse* não contém apenas dados resumidos, podendo conter também dados primitivos. Deve-se prover ao usuário a capacidade de aprofundar-se num determinado tópico, investigando níveis de agregação menores ou mesmo o dado primitivo, permitindo também a geração de novas agregações ou correlações com outras variáveis. Além do mais, é extremamente difícil prever todos os possíveis dados resumidos que serão necessários: limitar o conteúdo de um *data warehouse* apenas a dados resumidos significa limitar os usuários apenas às consultas e análises que eles puderem antecipar frente a seus requisitos atuais, não deixando qualquer flexibilidade para novas necessidades.

A especificação de requisitos do ambiente de suporte à decisão, associado a um *data warehouse,* é fundamentalmente diferente da especificação de requisitos dos sistemas que sustentam os processos usuais do ambiente operacional de uma empresa. Os requisitos dos sistemas do ambiente operacional são claramente identificáveis a partir das funções a serem executadas pelo sistema. Geralmente, os sistemas do ambiente operacional que apóiam os

usuários em suas funções do dia-a-dia são chamados OLTP (*On Line Transaction Processing*), e seu principal objetivo é executar o maior número de transações possíveis no menor tempo de processamento. Em geral, os sistemas OLTP são pouco flexíveis em relação à quantidade de relatórios e consultas, devido às limitações impostas por seu modelo de dados e à linguagem SQL. Em sistemas de suporte a decisão, onde o volume de dados costuma ser muito maior e as consultas altamente complexas, os requisitos são difíceis de determinar, o que culmina na necessidade de ferramentas altamente flexíveis e customizáveis. Para atender essa necessidade são adotados os sistemas OLAP (*On Line Analytical Processing*). Sistemas OLAP permitem aos usuários de alto nível, como gerentes e analistas de negócio, navegarem entre os dados da empresa com maior facilidade, proporcionando uma visão multidimensional desses dados.

Inmon (1999) nos dá uma definição bastante completa sobre o OLAP:

> *(...) é uma tecnologia de software que permite a analistas, gerentes e executivos a obterem os dados de uma forma rápida, consistente e com acesso interativo para uma grande variedade de possíveis visões da informação na empresa. Mais sucintamente, OLAP é um conjunto de funcionalidades que tem, como principal objetivo, facilitar a análise multidimensional.*

Sistemas OLAP fornecem uma visão multidimensional dos dados não importando como estes dados estão fisicamente armazenados. Os dados são percebidos pelo usuário como um cubo multidimensional onde cada célula contém um valor ou medida.

Cubo é uma estrutura multidimensional de dados que expressa a forma na qual os tipos de informações se relacionam entre si. O cubo de uma forma genérica armazena todas as informações relacionadas a um determinado assunto, de maneira a permitir que sejam

montadas várias combinações entre elas, resultando na extração de várias visões sobre o mesmo tema.

Alguns conceitos são importantes para que se possa compreender melhor o funcionamento de um ambiente *data warehouse* e são definidos a seguir.

Sistemas operacionais de origem: São considerados sistemas operacionais de origem qualquer sistema de registro que captura as transações da empresa. Sistemas de origem devem ser considerados como externos ao *data warehouse* porque se presume que se tenha um pouco ou nenhum controle sobre o conteúdo e formato dos dados nesses sistemas.

Metadados: Toda e qualquer informação no ambiente de *data warehouse* que não são os dados propriamente ditos, são chamados metadados. Estes são como uma enciclopédia para o *data warehouse*. Eles estão presentes em uma variedade de formas e formatos para suportar as necessidades desiguais dos grupos de usuários técnicos, administrativos e de negócio do *data warehouse*.

Quando os dados estão na área de armazenamento, encontramos metadados específicos para orientar os processos de transformação e carga, inclusive arquivos de teste e *lay-outs* de tabelas de destino, regras de transformação e filtragem, definições de dimensão e fato em conformidade, definições de agregação e resultados de *log* de execução e cronogramas de transmissão ETL (O termo "ETL" é tratado logo a seguir). Até códigos de programação personalizados que foram criados na área de armazenamento, são metadados.

Os metadados que permeiam o SGBD – Sistema Gerencial de Banco de Dados do *data warehouse* são responsáveis por itens como tabelas de sistemas, configurações de partição, índices, definições de exibição e concessões e privilégios de segurança no nível do

SGBD. Os metadados de ferramentas de acesso a dados identificam os nomes e as definições comerciais das tabelas e colunas da área de apresentação, bem como filtros de restrição, especificações de modelos de aplicação, estatísticas de uso e acesso e outras documentações de usuário.

O objetivo dos metadados é cercar, catalogar, integrar e melhorar essas diversas variedades de metadados, da mesma forma que os recursos de uma biblioteca [KIMBALL, 2002].

1.2 ETL – Extração, Transformação e Carga

No ambiente de *data warehouse*, os dados são inicialmente extraídos de sistemas operacionais e de fontes externas, posteriormente integrados e transformados (limpos, eliminados, combinados, validados, consolidados, agregados e sumarizados), antes de serem carregados no *data warehouse*. Finalmente, os usuários acessam o DW através de ferramentas de *front-end* ou aplicações submetendo suas consultas, de modo a obterem informações que permitam a tomada de decisões. Um DW contém dados sumarizados, históricos e detalhados para suportar a tomada de decisões táticas e estratégicas. A figura 1 apresenta o ambiente *data warehouse*, mostrando o fluxo realizado até a disponibilização dos dados aos usuários.

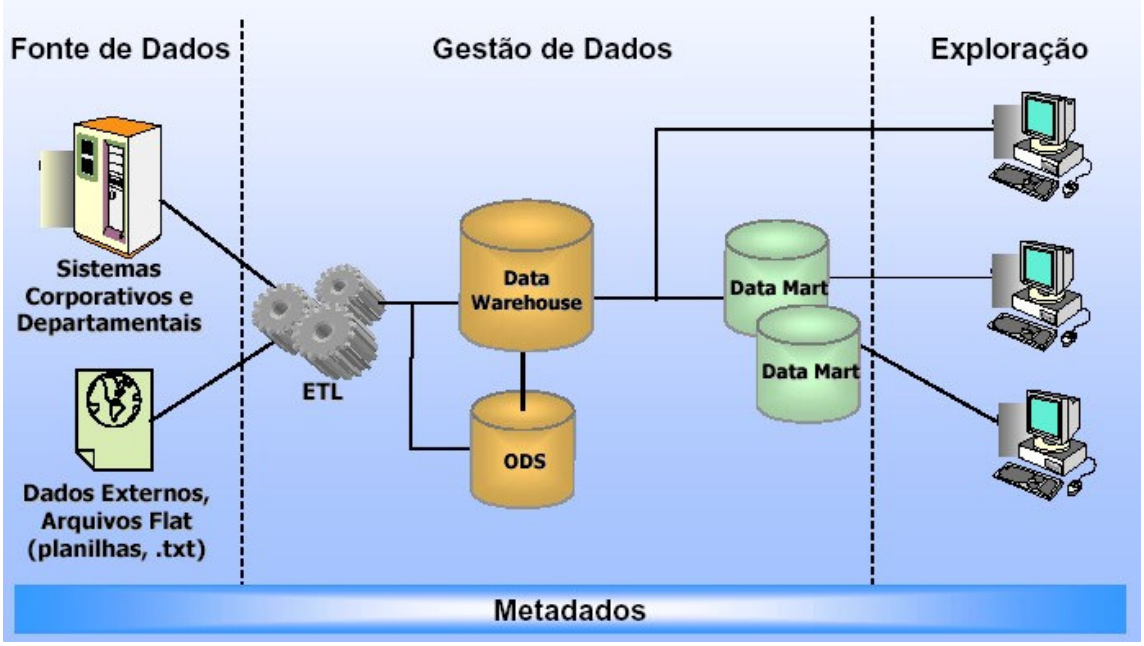


Figura 1: Ambiente *Data Warehouse*.

1.2.1 Extração

A extração é o primeiro passo na obtenção de dados para o ambiente do DW. Significa basicamente ler e entender as fontes de dados e copiar as partes necessárias para a área de transformação de dados, a fim de serem trabalhadas posteriormente. Na grande maioria dos DW, os dados provêm de várias fontes diferentes e independentes, podendo ser essas fontes as bases de dados dos sistemas transacionais, planilhas excel, etc.

Freqüentemente, o grande desafio é determinar quais dados extrair e que tipos de filtros aplicar, essa atividade é uma das que mais consomem tempo na construção do DW. A extração pode ser conduzida através da construção de programas cujo código é executado sobre um sistema fonte de modo a gerar arquivos com os dados desejados. Outra opção é utilizar ferramentas de extração específicas que geram código próprio, interno à ferramenta,

executado sobre o sistema fonte, de forma a obter os dados necessários, de preferência dentro de arquivos de formato não proprietário, como, por exemplo, arquivos texto.

1.2.2 Transformação de dados

Após a extração dos dados dos sistemas fontes, são realizadas uma série de atividades sobre esses dados de modo a convertê-los em formato adequado para carga no *data warehouse* e valioso para o negócio. A transformação dos dados poderá envolver um ou vários processos, dependendo da necessidade e situação. Seguem alguns dos processos mais comumente utilizados:

- limpeza: constitui no conjunto de atividades realizadas, sobre os dados extraídos, de modo a corrigir o uso incorreto ou inconsistente de códigos e caracteres especiais, resolver problemas de conflito de domínios, tratar dados perdidos, corrigir os valores duplicados ou errados. Independente do problema a ser solucionado pelo processo de limpeza, a finalidade é deixar os elementos de dados dentro de formatos padrões (uniformizados), não duplicados, corretos, consistentes e espelhando a realidade;
- eliminação: constitui em desconsiderar os campos e dados provenientes de sistemas legados que não são úteis ao DW;
- combinação: é realizada quando fontes de dados possuem exatamente os mesmos valores de chaves representando registros iguais ou complementares;
- desnormalização e normalização: o padrão no processo de transformação é reunir as hierarquias de dados, separadas em várias tabelas devido à normalização, dentro de uma única dimensão, de forma desnormalizada. Pode ocorrer, entretanto, que dados provenientes do processo de extração estejam

completamente desnormalizados dentro de arquivos texto, nesse caso é possível que seja necessário normalizar partes dos registros.

- cálculos, derivação e alocação: são transformações a serem aplicadas às regras do negócio identificadas durante o processo de levantamento de requisitos. É conveniente que as ferramentas a serem empregadas possuam um conjunto de funções, tais como manipulação de textos, aritmética de data e hora, entre outras.

1.2.3 Carga de dados

Após os dados serem transformados, eles são carregados no *data warehouse*. A parte de carga dos dados também possui uma enorme complexidade, e os seguintes fatores devem ser levados em conta:

- Integridade dos dados: no momento da carga, é necessário checar os campos que são chaves estrangeiras com suas respectivas tabelas para certificar-se de que os dados existentes na tabela da chave estrangeira estão de acordo com a tabela da chave primária;
- Tipo de carga a ser realizada - incremental ou a total: a carga incremental normalmente é feita para tabelas de fatos e a carga total é feita em tabelas de dimensão onde o analista terá que excluir os dados existentes e incluí-los novamente. Mas isso depende da necessidade do negócio em questão;
- Otimização do processo de carga: todo banco de dados possui um conjunto de técnicas para otimizar o processo de carga, tais como evitar a geração de *log* durante o processo, criar índices e agregar dados. Muitas dessas características podem ser invocadas dos bancos de dados ou registradas em scripts através da utilização de ferramentas sobre a área de organização de dados;

- Suporte completo ao processo de carga: o serviço de carga também precisa suportar as exigências antes e depois da carga atual, como eliminar e recriar índices e particionamento físico de tabelas e índices.

1.3 Modelo de Dados

A elaboração do modelo de dados concentra-se na observação dos fatos relevantes que ocorrem na realidade, com a finalidade de construir um sistema que possa automatizar as necessidades de informação da mesma. Neste momento, os documentos que registram estes fatos só devem ser utilizados como apoio de entendimento, e não como base para o desenvolvimento do sistema de informações, ou seja, não se deve ter a preocupação em simular o ambiente atual, seja ele manual ou automatizado [MACHADO, 1996].

O analista deve ter a preocupação de retratar as necessidades de informação que as pessoas (que agem sobre esta realidade) precisam para alcançar os objetivos desta mesma realidade. Ao coletar e selecionar os fatos relevantes, o analista deve identificar os elementos geradores de informação, as leis que regem esta realidade, bem como, as operações que incidem sobre os elementos básicos (dados).

Para registrar as necessidades de informação de uma realidade, é necessário fazer uso de um modelo, ou seja, algo que nos mostre como as informações estão relacionadas (fatos). E, com base no modelo criado, os analistas podem interagir com os usuários validando seus objetivos e metas, permitindo a construção de um sistema de informações, cada vez mais, próximo da realidade do usuário.

Como será visto a seguir, os modelos de dados podem ser classificados segundo a arquitetura que utilizam. O modelo relacional surgiu para atender os sistemas transacionais (OLTP), já o modelo dimensional surgiu para atender os sistemas analíticos (OLAP).

1.3.1 Modelo Relacional

Criado por Edgar F. Codd, nos anos 70, o Modelo Relacional, começou a ser utilizado nas empresas a partir de 1977. A abordagem relacional está baseada no princípio de que as informações em uma base de dados podem ser consideradas como relações matemáticas e que estão representadas de maneira uniforme, através do uso de tabelas bidimensionais. Este princípio coloca os dados dirigidos para estruturas mais simples de armazenar dados, que são as tabelas, e nas quais a visão do usuário é privilegiada.

O modelo relacional é um conjunto de dados visto segundo um conjunto de tabelas, e suas operações sobre elas são feitas por linguagens que manipulam a álgebra relacional, não sendo procedurais, ou seja, manipulam conjuntos de uma só vez [MACHADO, 1996].

Esse modelo surgiu para atender sistemas transacionais que possuem operações atômicas (que devem ocorrer por completo ou então serem desfeitas) pré-definidas, geralmente, com um grande número de usuários simultâneos realizando operações repetidamente.

Para se compreender melhor o modelo de dados relacional, é preciso o conhecimento de alguns conceitos que são definidos a seguir.

Chave: designa o conceito de item de busca, ou seja, um dado que será empregado nas consultas à base de dados. É um conceito lógico de aplicação.

Índice: é um recurso físico visando otimizar a recuperação de uma informação, via um método de acesso. Seu objetivo principal está relacionado com a performance de um sistema.

O modelo relacional deve estar preparado para ter um tempo de resposta pequeno, uma vez que, as transações são curtas e operam sobre poucos dados. Uma tabela é acessível por qualquer campo independentemente se este é declarado como chave ou não. O relacionamento entre conjunto de dados (tabelas) não existe fisicamente, pois este relacionamento é apenas lógico e representado através das chaves estrangeiras (elos de ligação entre as tabelas). Esse modelo visa a eliminação de redundância, deixando os dados em um único lugar.

A base de dados de um modelo relacional reflete imediatamente as operações realizadas, com atualizações que são visualizadas em tempo real, não existindo um processo de carga de dados. A inserção de um registro ou a atualização de um registro já existente é imediatamente vista pelos usuários do sistema e geralmente não existe a necessidade de consultas a dados históricos.

Por possuir atualizações em tempo real com usuários simultâneos, esse modelo exige controle de concorrência, por meio de mecanismos de bloqueio, *commit* e *rollback* para evitar qualquer tipo de problema com os dados, como a perda de informações.

Na criação de um modelo de dados relacional, são realizados aprimoramentos chamados de normalização, tornando o modelo menos redundante e menos inconsistente.

Normalização: é o processo de reunir todos os dados que serão armazenados em um certo banco de dados e separá-los em tabelas. Através do processo de normalização, pode-se substituir um conjunto de entidades e relacionamentos por um outro, o qual se apresenta "purificado" em relação às anomalias de atualização (inclusão, alteração e exclusão), as quais, podem causar certos problemas, tais como: grupos repetitivos (atributos multivalorados) de dados, redundância de dados desnecessários, perdas acidentais de informação, dificuldade na representação de fatos da realidade [MACHADO, 1996].

Pode-se citar alguns benefícios do processo da Normalização: *estabilidade do modelo*, mantendo-o inalterado face a mudanças que venham a ser percebidas ou introduzidas no ambiente que tenha sido modelado; *flexibilidade,* com o processo de normalização das tabelas aumenta a capacidade de adaptação a demandas diferenciadas, a expansão e redução, omissão ou presença (das tabelas); *integridade,* refere-se à qualidade do dado, se um dado sobre certo objeto aparecer mapeado em mais de um local de modo diferente, pode-se ter indícios de que não há integridade entre eles; *economia,* com a existência de redundâncias nos dados, o custo de armazenamento, torna-se muito maior, além de também aumentar o custo de manipulação dos dados.

1.3.2 Modelo Dimensional

Modelagem dimensional é um nome para uma técnica antiga que permitia tornar os bancos de dados fáceis e compreensíveis. Caso após caso, a partir da década de 1970 as empresas de Tecnologia da Informação, as consultorias, os usuários finais e os fornecedores

migraram para uma estrutura dimensional simples para atender a necessidade humana fundamental de simplicidade [KIMBALL, 1998].

O modelo dimensional surgiu para atender sistemas de processamento analítico, com consultas para planejamento tático e estratégico da empresa. Atualmente a utilização desses sistemas pelo nível operacional das empresas vem crescendo, auxiliando o processo de tomada de decisões diárias. Normalmente, ele atende um pequeno número de usuários que realizam consultas planejadas (relatórios pré-definidos) e *ad-hoc*.

O tempo de resposta é maior, devido a consultas longas que operam sobre grande volume de dados. Para melhor desempenho nas consultas, há redundância planejada dos dados, compensando os gastos com armazenamento e atualização das informações. O resultado é uma estrutura simples, com modelos que refletem o processo de análise de negócios.

A base de dados não é atualizada pelo usuário, portanto, não disponibiliza as alterações realizadas entre uma atualização e outra. As atualizações são feitas periodicamente em *batch,* não havendo a necessidade de controle de concorrência. Os usuários somente realizam consultas na base de dados, podendo extrair e formatar seus próprios relatórios, não dependendo da equipe de tecnologia para isso.

Para realização de análises, a base de dados é planejada para armazenar muitos dados históricos, tipicamente de 12 a 60 meses.

Os sistemas analíticos suportam a gestão da empresa, como planejamento de marketing (expansão de lojas, lançamento de ofertas), análise de vendas (melhores e piores clientes / produtos / vendedores), etc.

Processamento Transacional:	**Processamento Analítico:**
Dados normalizados	Dados consistentes
Atualização em tempo real	Desempenho compatível com o volume de dados (ou fórmulas, etc)
Controle de concorrência	Clara representação do modelo de negócio (camada semântica)
Dados correntes	Ferramentas especiais para os usuários finais
Respostas imediatas	

Quadro 1: Requisitos de processamento transacional e analítico.

1.3.3 - A escolha da modelagem

O quadro 2 a seguir mostra as principais diferenças entre sistemas de processamento transacional (OLTP - *On Line Transaction Processing*) e sistemas de processamento analítico (OLAP - *On Line Analytical Processing*).

Se OLAP e OLTP são tão distintos, a modelagem de dados para cada função deve ser também distinta, pois OLAP e OLTP se propõem a resolver problemas completamente diferentes.

É um erro pensar que técnicas de projeto que servem para sistemas convencionais serão adequadas para a construção de um *data warehouse* [INMON, 1999]. Os requisitos para um *data warehouse* podem não ser conhecidos até que ele esteja parcialmente carregado e já em uso.

O principal objetivo da modelagem relacional em um sistema OLTP é eliminar ao máximo, a redundância, de tal forma que uma transação que promova mudanças no estado do banco de dados, atue o mais pontualmente possível. Com isso, nas metodologias de projeto

usuais, os dados são "fragmentados" por diversas tabelas, o que traz uma considerável complexidade à formulação de uma consulta por um usuário final. Por isso, esta abordagem não parece ser a mais adequada para o projeto de um *data warehouse*, onde estruturas mais simples, com menor grau de normalização devem ser buscadas [KIMBALL, 2002].

	Transacional - OLTP	Analítico - OLAP
Usuários típicos	Usuários em geral	Gerentes, analistas de negócio
Aplicação do sistema	Operações do dia-a-dia	Análises do negócio
Interação do usuário	Pré-determinado	***Ad-hoc***
Características de trabalho	Leitura/gravação	Leitura
Unidade de trabalho	Transação	Consulta
Processamento	Orientado a processos	Orientado a assuntos
Atualização	Um registro por vez	Vários registros por vez

Quadro 2: Diferenças entre OLAP e OLTP
Fonte: Vaisman (1998, p. 5)

Obter respostas a questões típicas de análise dos negócios de uma empresa geralmente requer a visualização dos dados segundo diferentes perspectivas. Por exemplo: uma agência de automóveis que esteja querendo melhorar o desempenho de seu negócio necessita examinar os dados sobre as vendas disponíveis na empresa. Uma avaliação deste tipo requer uma visão histórica do volume de vendas sob múltiplas perspectivas: volume de vendas por modelo, volume de vendas por cor, volume de vendas por montadora, volume de vendas por período de tempo. Uma análise do volume de vendas utilizando uma ou mais destas perspectivas permitiria responder questões do tipo:

Qual a tendência, em termos de volume de vendas, para o mês de dezembro, de modelos Corsa Sedan pretos?

A capacidade de responder a este tipo de questão em tempo hábil é o que permite aos gerentes e altos executivos das empresas formularem estratégias efetivas, identificar tendências e melhorar sua habilidade de tomar decisões de negócio. O ambiente tradicional de bancos de dados relacional certamente pode atender a este tipo de consulta. No entanto, usuários finais que necessitam de consultas deste tipo via acesso interativo aos bancos de dados se frustram devido a tempos de resposta ruins e pela falta de flexibilidade oferecida por ferramentas de consulta baseadas em SQL. Daí a necessidade de utilizar abordagens específicas para atender a estas consultas.

São chamadas de “dimensões” as diferentes perspectivas envolvidas. Em nosso exemplo: loja, montadora, mês, essas dimensões usualmente correspondem a campos não numéricos em um banco de dados. Considerando um conjunto de “medidas”, tal como vendas ou despesas com promoção, essas medidas correspondem geralmente a campos numéricos em um banco de dados. A seguir, as agregações dessas medidas são avaliadas segundo as diversas dimensões e armazenadas para acesso futuro. Por exemplo, calcula-se a média de todas as vendas por todos os meses por loja. A forma como essas agregações são armazenadas pode ser vista em termos de dimensões e coordenadas, dando origem ao termo multidimensional.

Ao contrário de aplicações convencionais, como uma folha de pagamento ou inventário, a classificação de instâncias em problemas multidimensionais é dependente do objetivo da análise do usuário, não sendo consideradas propriedades próprias das entidades envolvidas. Os tipos de classificação usados fazem surgir as dimensões descritivas, segundo as quais observações dos objetos ou eventos são observadas e medidas.

Cada eixo no espaço multidimensional é um campo/coluna de uma tabela relacional e cada ponto um valor correspondente à interseção das colunas. Por exemplo, o valor para o

campo vendas, correspondente a mês igual a novembro e loja igual a Iguatemi é um ponto com coordenada (“novembro”, “Iguatemi”). Nesse caso, mês e loja são duas dimensões e vendas é uma medida.

Teoricamente, quaisquer dados podem ser considerados multidimensionais. Entretanto, o termo normalmente se refere a dados representando objetos ou eventos que podem ser descritos por dois ou mais de seus atributos. Estruturas relacionais podem ser usadas para a representação e o armazenamento de dados multidimensionais. Nesse caso, as abordagens encontradas incluem desde a adoção de formas específicas de modelagem (os chamados esquemas estrela e floco de neve) até mecanismos sofisticados de indexação.

Neste capítulo foram tratados conceitos básicos de *data warehouse* e modelo de dados, mostrando as diferenças básicas entre o modelo relacional e o modelo dimensional, que será tratado com maior profundidade no capítulo que segue.

Capítulo 2 - A Modelagem Dimensional

A modelagem dimensional é uma metodologia que permite modelar logicamente dados para melhorar o desempenho de consultas e prover facilidade de utilização a partir de um conjunto de eventos básicos de medição. Os modelos dimensionais são compreensíveis, previsíveis, ampliáveis e resistentes ao ataque específico de grupos de usuários de negócio, por se manter fiel à simplicidade, ter uma perspectiva voltada para as necessidades analíticas da empresa, e especialmente ao seu formato simétrico, em que todas as dimensões normalmente são iguais pontos de entrada na tabela de fatos [KIMBALL, 2002]. Os modelos dimensionais são a base de muitos aprimoramentos de desempenho SGBD, inclusive agregações e métodos de indexação avançados.

2.1 Exemplo de Modelos de dados

O sistema empregado para apresentação dos exemplos de modelos de dados é um sistema de vendas de uma rede de lojas de departamento. Essa rede possui diversas lojas distribuídas em diferentes estados do Brasil. O sistema transacional gerencia todas as vendas efetuadas em cada uma das lojas.

O modelo relacional utilizado nesse sistema utilizado nesse sistema, ilustrado na figura 2, possui uma tabela Faturas que contém todos os dados relacionados a uma fatura, com seus itens vendidos sendo armazenados na tabela Itens_Fatura. Para gerenciar as vendas, possui também um cadastro de Lojas, de Clientes, de Funcionários, de Produtos, de Categorias e de Fornecedores.

Cada venda ocorrida representa uma linha na tabela Faturas e *n* linhas na tabela Itens_Fatura, sendo que *n* é o número de itens (produtos) diferentes vendidos nessa transação.

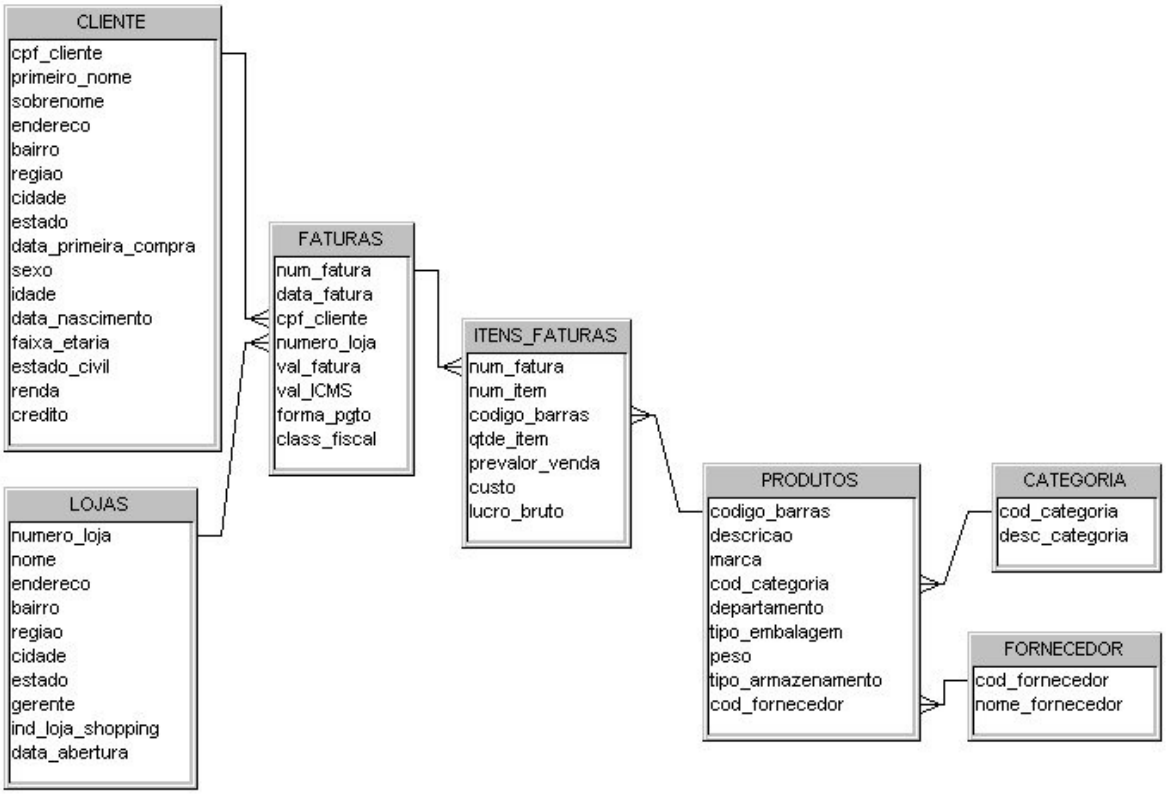


Figura 2: Modelo relacional do sistema transacional de vendas de lojas de departamento.

Para se realizar consultas analíticas, o modelo relacional não oferece o desempenho adequado, pois cada informação se encontra em uma tabela diferente. A normalização é um dos princípios de tal modelo, sendo assim, o grande número de junções é inevitável, tornando o processamento muito lento.

Por exemplo, um analista de marketing quer saber qual o faturamento com a venda de cosméticos do fornecedor "Johnson & Johnson", em lojas da zona sul da cidade de São Paulo no período do dia das mães. No modelo relacional a consulta precisaria fazer a junção das tabelas Categorias, Produtos, Fornecedores, Itens_Fatura, Faturas e Lojas. Seria feita uma

busca por "cosméticos" na tabela Categorias, uma busca por "Johnson & Johnson" na tabela Fornecedores, e uma junção dessas tabelas com a tabela Produtos para identificar quais são os produtos dessa categoria e desse fornecedor. Na tabela Lojas seria feita uma busca por "zona sul" e cidade "São Paulo", e as datas referentes ao período do dia das mães teriam que ser limitadas manualmente na tabela Faturas. Com isso, é possível retornar o faturamento total da tabela Itens_Fatura.

Para realizar essa consulta no modelo relacional, foram utilizadas 6 tabelas, realizando um total de 5 junções, o que em um modelo dimensional poderia ser otimizado, como é apresentado a seguir.

A figura 3 apresenta o modelo dimensional desse sistema, preparado para atender necessidades de análises das vendas, a partir do cruzamento das informações das lojas da rede, com os produtos vendidos, os clientes e a data da venda.

Esse modelo possui uma tabela central chamada Vendas que armazena os dados principais a serem analisados da venda, como o faturamento e o lucro. Essa tabela tem relacionamento com as demais tabelas necessárias para identificar um item de venda com base no cliente, data, loja e produto.

A tabela Clientes possui os dados necessários para identificação do comprador, e dados importantes para filtrá-los, como idade, faixa etária, renda, sexo, etc. A tabela Data armazena um registro para cada dia durante o período de existência da rede de lojas de departamento, com campos para facilitar a busca por uma determinada data ou período, como a data no ano calendário ou fiscal, se o dia é um feriado ou não, qual a temporada de vendas dessa data (dia dos namorados, dia das mães, Natal), etc.

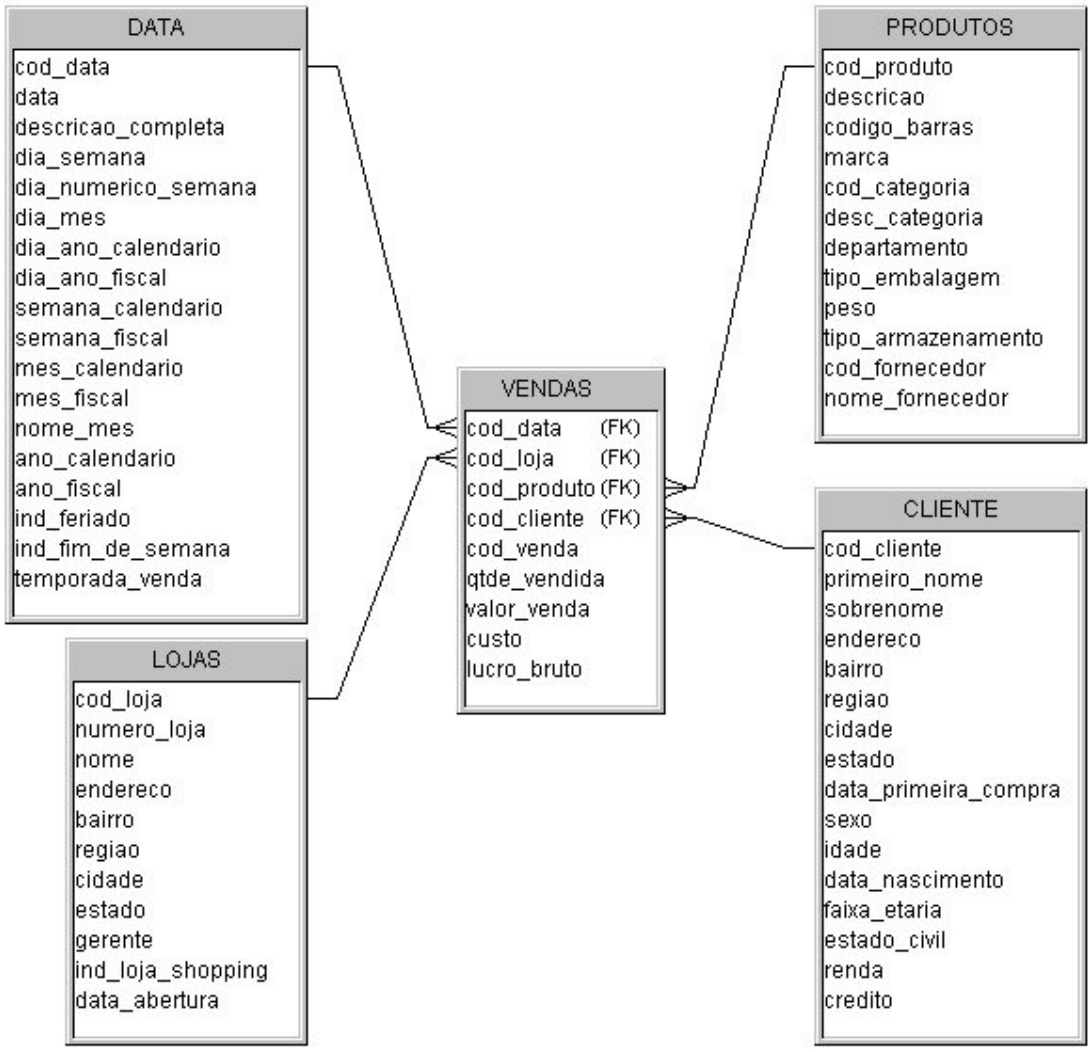


Figura 3: Modelo dimensional do sistema analítico de Vendas de Lojas de departamento.

A tabela Lojas possui os dados de identificação para cada uma das lojas, e dados para filtro, como um campo para indicar se uma loja pertence ou não a um shopping center. Por fim a tabela Produtos armazena os dados de cada um dos produtos comercializados pelas lojas da rede, sua categoria e fornecedor, e campos de caracterização do produto, como o tipo de embalagem (garrafa, caixa, pacote) e peso.

Nesse esquema, a busca pelo faturamento dos cosméticos do fornecedor "Johnson & Johnson" vendidos na zona sul de São Paulo no período do dia das mães envolveria a junção

apenas das tabelas Vendas, Lojas, Produtos e Data. A busca por categoria e fornecedor envolveria uma única tabela, já que todos estes dados estão armazenados na tabela Produtos. Na tabela Data seriam consultados todos os registros que tivessem o campo temporada de venda com o valor "dia das mães", deste ano calendário. Essa é uma maneira mais simples do que identificar quais foram os dias que fizeram parte desse período no momento da consulta, evitando possíveis distorções por parte dos usuários. Na tabela Lojas é feita a seleção por região "zona sul" e cidade "São Paulo", e, através da junção dessas três tabelas na tabela Vendas, é retornado o faturamento total.

Como o modelo relacional trabalha com normalização, suas tabelas possuem menos registros e não têm redundâncias, apresentando assim uma melhor performance nas tarefas do dia a dia, como inclusões, alterações e exclusões de registros, mas ele só é adequado para consultas simples de poucos registros. Para análises mais complexas com um universo de registros maior, o modelo dimensional oferece uma melhor alternativa, economizando em junções com várias tabelas, e armazenando dados que facilitam a análise das informações.

O modelo dimensional, além de atender a consulta sobre as vendas no período do dia das mães com uma performance melhor, facilita outros tipos de análise como comparações entre os resultados das vendas do dia das mães desse ano com os resultados de anos anteriores, avaliação média do perfil dos compradores (faixa etária, renda média, etc.), etc.

2.2 Esquema Estrela

O esquema estrela é uma estrutura simples, com poucas tabelas e ligações (relacionamentos) bem definidas [POE, KLAUER, BROBST, 1998], assemelha-se ao modelo

de negócio, o que facilita a leitura e entendimento, não só pelos analistas, como por usuários finais não familiarizados com estruturas de banco de dados. Permite a criação de um banco de dados que facilita a execução de consultas complexas, podendo ser realizadas de modo eficiente e intuitivo pelo usuário.

O modelo dimensional requer a utilização de ferramentas de consultas analíticas, desenvolvidas especialmente para consultar esse tipo de modelo, o que permite aos usuários a exploração de todos os dados disponíveis durante a elaboração das consultas.

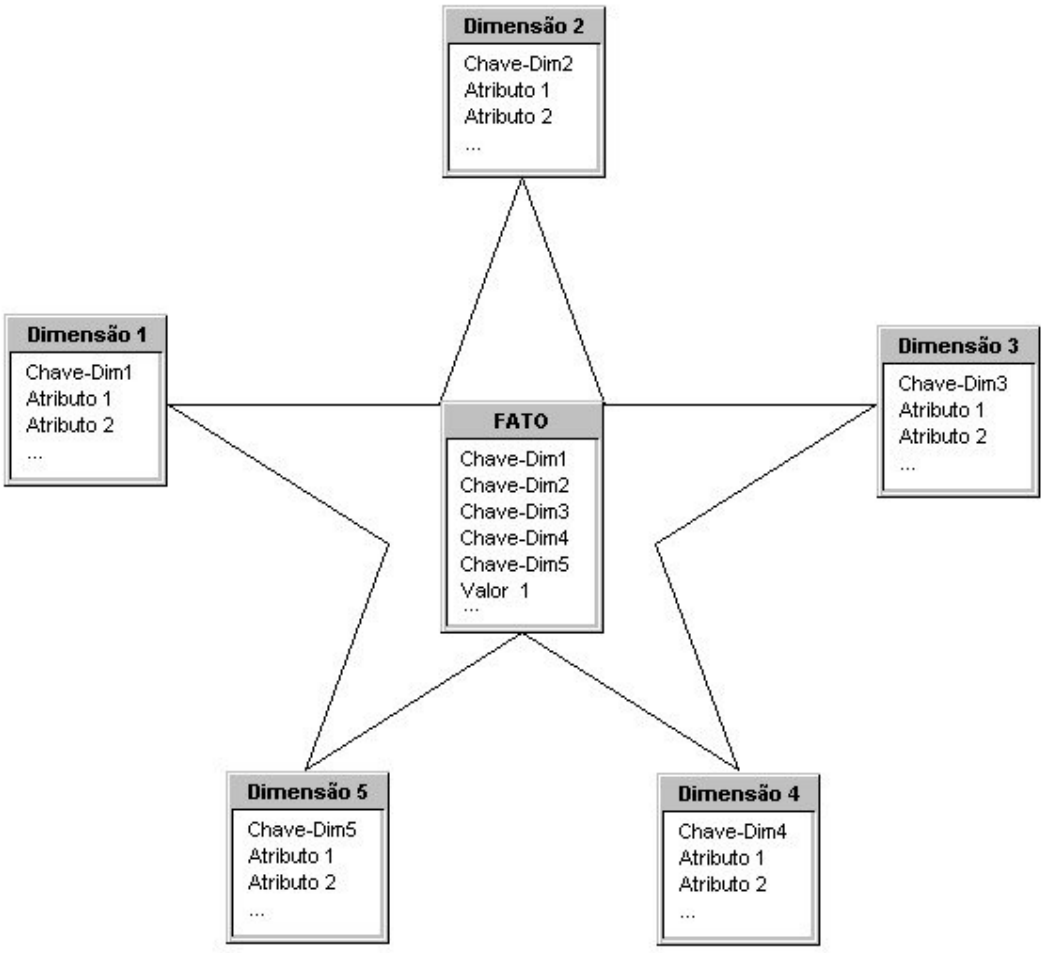


Figura 4: Representação da estrutura do Esquema Estrela.

O nome "estrela" está associado à disposição das tabelas no modelo, que consiste de uma tabela central, a tabela de fatos, que se relaciona com diversas outras tabelas, as tabelas de dimensão (os conceitos de tabelas de fatos e tabelas de dimensão são apresentados em seguida). A figura 4 apresenta a estrutura geral de um esquema estrela.

O esquema estrela pode representar tanto o modelo lógico, como o modelo físico do banco de dados. A representação mais simples de um modelo dimensional contém um esquema estrela com uma tabela de fatos relacionada com tabelas de dimensão, mas um modelo dimensional pode ter uma ou mais tabelas de fatos, relacionadas com tabelas de dimensão. Entretanto, a visão de um esquema por vez torna o modelo mais claro.

2.2.1 Tabela de Fatos

A tabela de fatos é a principal tabela de um modelo dimensional, onde as medições numéricas de interesse da empresa estão armazenadas [KIMBALL, 2002]. A palavra "fato" representa uma medida dos processos que estamos modelando, como quantidades, valores e indicadores. A tabela de fatos registra os fatos que serão analisados. É composta por uma chave primária (formada por uma combinação única de valores de chaves de dimensão) e pelas métricas de interesse para o negócio.

As dimensões indicam a forma como as medidas serão vistas, os seja, são os aspectos pelos quais se pretende observar as métricas. A intersecção das chaves de dimensão define a granularidade da tabela de fatos, e é importante que todas as medidas na tabela de fatos tenham a mesma granularidade.

A granularidade diz respeito ao nível de detalhe ou de resumo contido nas unidades de dados existentes no *data warehouse* [INMON, 1997]. Quanto mais detalhe, mais baixo o nível de granularidade. Quanto menos detalhe, mais alto o nível de granularidade.

Os modelos dimensionais devem armazenar a informação mais detalhada no processo do negócio, preferencialmente dados que não podem ser subdivididos, como uma linha de item de uma venda, por exemplo. Por isso, normalmente uma tabela de fatos é grande, com milhões de registros.

A tabela de fatos deve ser sempre preenchida com as medidas referentes ao fato. Não se deve preencher uma linha da tabela fato com zeros para representar que nada aconteceu (por exemplo, que não houve vendas de um produto em determinada data), pois isso faria com que a tabela de fatos crescesse demais.

A tabela de fatos é sempre esparsa, ou seja, possui um número relativamente pequeno de todas as combinações possíveis de valores de chaves. Por exemplo, no caso de um banco de dados de uma companhia aérea, a presença de todas as combinações possíveis representaria que todos os clientes voam todos os dias, e em todos os vôos feitos pela companhia, o que na prática é impossível. Por isso podemos dizer que esse banco é extremamente esparso, pois uma porcentagem muito pequena de todas as combinações possíveis de clientes, número do vôo e dia aparecerão nele.

Outro ponto importante é que a tabela de fatos deve representar uma unidade do processo do negócio, não devendo-se misturar assuntos diferentes numa mesma tabela de fatos.

Os atributos mais comuns em uma tabela de fatos são valores numéricos. Estes valores são, em sua maioria, aditivos. As métricas aditivas são as que permitem operações como adição, subtração e média de valores por todas as dimensões, em quaisquer combinações de registros, como "total de itens vendidos" por combinação de data, produto e loja. Métricas aditivas são importantes porque normalmente as aplicações de *data warehouse* não retornam uma linha da tabela de fatos, mas sim centenas, milhares e até milhões.

Existem também métricas não-aditivas e métricas semi-aditivas. As métricas não-aditivas são valores que não podem ser manipulados livremente, como valores percentuais ou relativos. Para esses valores, os cálculos devem ser realizados nos dados absolutos nos quais se baseiam. Exemplos de métricas não-aditivas são preço de custo e preço de venda de um produto em uma venda. Por fim, as métricas semi-aditivas são valores que não podem ser somados em todas as dimensões. Por exemplo: numa tabela com o registro diário do saldo bancário dos clientes de uma agência, não faz sentido somar os saldos bancários diários de um cliente durante um mês, mas pode-se somar os saldos de todos os clientes de uma agência em determinada data.

Um exemplo de uma tabela de fatos é a tabela Vendas da rede de lojas de departamentos vista anteriormente, apresentada na figura 5.

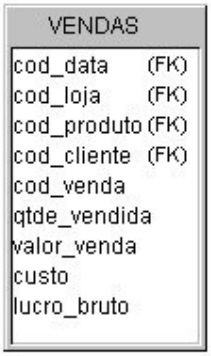


Figura 5: Tabela de fatos Vendas.

2.2.2 Modelagem da Tabela de Fatos

Uma técnica para modelar a tabela de fatos é responder as seguintes perguntas:

- Que processo estamos modelando?
- O que usamos para medir este processo?
- Quais os indicadores críticos de desempenho desse processo?

Um processo é uma atividade de negócio natural executada na empresa. Ouvir os usuários é o meio mais eficiente de selecionar o processo de negócio. É importante lembrar que não estamos nos referindo a um departamento ou função de negócio de uma empresa quando falamos de processos de negócio. Se forem estabelecidos modelos dimensionais para cada departamento, inevitavelmente os dados serão duplicados com diferentes rótulos e terminologias [KIMBALL, 2002].

Os valores necessários para se medir o processo podem ser encontrados prontos no sistema origem, ou então podem ser calculados baseados em medidas do sistema origem. Os fatos típicos são valores numéricos aditivos.

Os indicadores críticos de desempenho são os indicadores em que a empresa se baseia para fazer suas análises. A tabela de fatos deve conter fatos que permitam que esses indicadores sejam construídos.

Por exemplo: o usuário quer saber quais foram as vendas no período de janeiro a março deste ano na região sudeste e nordeste, quais os produtos mais rentáveis, quem foram os maiores clientes, quais foram os melhores vendedores, onde vendi mais que o concorrente, e que produtos vendem menos que os do concorrente. Como resposta às perguntas, temos:

- Que processo estamos modelando? - Processo de vendas

- O que usamos para medir este processo? - Usamos o valor das vendas e quantidade vendida
- Quais os indicadores críticos de desempenho desse processo? - Atingimento da cota de venda, margem de lucro, *benchmark* com o mercado

2.2.3 Classificação dos Fatos

Os fatos podem ser classificados em transações individuais, em *snapshots* e em linhas de itens [KIMBALL et al, 1998].

As chamadas transações individuais representam uma transação completa, normalmente, apresentam uma estrutura muito simples, possuem um campo acumulado que contém o valor total de uma transação.

Os fatos *snapshots* representam medidas de atividades extraídas em tempo determinado, como, por exemplo, final do dia ou final do mês, com medições acumuladas durante esse período.

Os fatos do tipo linhas de itens são aqueles que representam exatamente uma linha de item, são armazenados no nível mais detalhado dos fatos, como, por exemplo, itens de pedido, itens de entrega e itens de apólice de seguro.

2.2.4 Tabela de Dimensão

A tabela de dimensão contém as descrições textuais do negócio, e possui as informações necessárias para análises ao longo de dimensões. Seus atributos são fonte das restrições das consultas, agrupamento dos resultados, e cabeçalhos para relatórios. As

dimensões são os aspectos pelos quais se pretende observar as métricas relativas ao processo que está sendo modelado.

A qualidade do banco de dados é proporcional à qualidade dos atributos de dimensões, portanto deve ser dedicados tempo e atenção a sua descrição, ao seu preenchimento e a garantia da qualidade dos valores em uma coluna de atributos [KIMBALL, 2002].

Cada dimensão é definida com uma única chave primária. Essa chave é a base da integridade referencial no relacionamento com a tabela de fatos. Uma tabela de dimensão é composta também de atributos, os melhores atributos são textuais e distintos, e devem consistir de palavras reais, evitando-se o uso de códigos e abreviações. Esses atributos descrevem as linhas na tabela de dimensão. A figura 6 mostra um exemplo de tabela de dimensão Produtos.

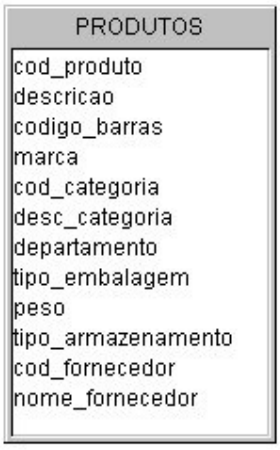


Figura 6: Tabela de dimensão Produtos.

A tabela de dimensão costuma ser bem menor que a tabela fato, geralmente com muito menos do que um milhão de registros, no entanto é comum existirem tabelas de dimensão com muitos atributos (de 50 a 100).

É muito importante que os atributos das tabelas de dimensão sejam preenchidos com valores descritivos ao invés de códigos sem sentido, criptografados ou abreviados [KIMBALL, 2002]. Por exemplo, em uma tabela de dimensão Alimentos, o campo "perecível" deve ser preenchido com valores como "É perecível" ou "Não é perecível" ao invés de usar simplesmente "S" e "N". Em um relatório com a listagem de milhares de produtos, um valor descritivo tem muito mais utilidade do que códigos. Ao invés de usar uma aplicação para decodificar esses códigos e mostrar uma descrição, é melhor armazenar essas descrições no banco de dados, tornando a informação disponível ao usuário independentemente de seu aplicativo de acesso aos dados.

2.2.5 Hierarquia de Dimensões

As hierarquias descrevem a lógica dos relacionamentos entre os dados são a base para a navegação entre os diferentes níveis de detalhe em uma estrutura multidimensional [MEYER, CANNON, 1998]. A figura 7 apresenta os níveis de agregações, ou seja, agrupamentos que podem ser aplicados à dimensão Produto. Muitas dimensões apresentam uma estrutura hierárquica ou multinível.

Algumas estruturas hierárquicas são facilmente identificadas, como por exemplo, uma estrutura de tempo representada por horas, dias, semanas, meses, trimestres e anos ou uma estrutura geográfica representada por cidades, municípios, estados, regiões e países [KIMBALL, 1996]. Dois tipos de hierarquia podem ser considerados para uma dimensão: explícita e implícita.

- Hierarquias explícitas: As hierarquias são caracterizadas por uma seqüência de entidades interligadas, cujos relacionamentos, entre cada par de entidades na

seqüência, é N:1. A figura 7 representa a hierarquia explícita para a dimensão **Produto**. Essa hierarquia é constituída pelas dimensões **Tipo** e **Categoria**.

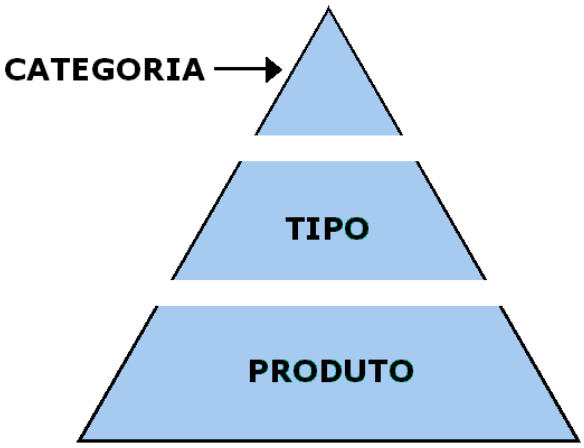


Figura 7: Hierarquia explícita de Produto.

- Hierarquias implícitas: também conhecidas como múltiplas hierarquias, representam as hierarquias embutidas nos atributos das dimensões. Um exemplo para múltiplas dimensões é representado na figura 8, com a classificação de produtos de acordo com Tipo de Armazenamento e Tipo de Embalagem. Os alimentos podem ser subcategorizados, quanto ao armazenamento refrigerado ou "não refrigerado", ou quanto ao tipo de embalagem, em "caixa" e "pacote". Do mesmo modo, os materiais de limpeza podem ser classificados, quanto à sua fórmula, em tóxica ou não tóxica, ou, quanto à sua consistência, em líquida, pastosa ou em pó.

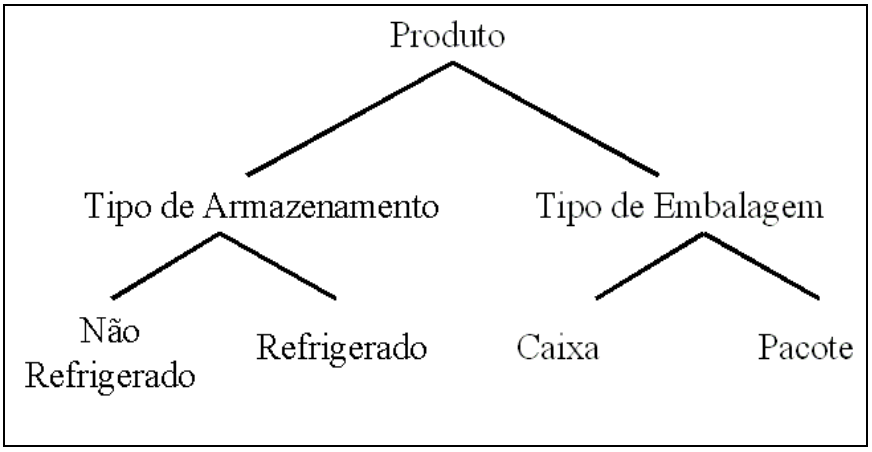


Figura 8: Hierarquia implícita para Produto.

Uma questão importante a ser abordada diz respeito à influência da hierarquia das dimensões sobre a tabela de fato. A tabela de fatos deve refletir a menor granularidade das dimensões, de modo a garantir que não sejam armazenados registros que representem totais referentes a um nível mais alto na hierarquia de uma dimensão.

Dessa forma, se a dimensão **Produto,** na figura 7, apresenta a hierarquia:

Categoria / Tipo / Produto, os registros na tabela de fatos devem indicar totais no nível de produto. Os registros que totalizem por **Categoria** ou **Tipo** não devem ser armazenados.

2.2.6 *Drill-down* e *Roll-up*

Uma característica bastante interessante nas análises dimensionais é a possibilidade de detalhamento e de sintetização das informações, que chamamos respectivamente de *drill-down* e *roll-up*. A figura 9 apresenta um exemplo de *drill-down* e *roll-up* através de uma informação de data.

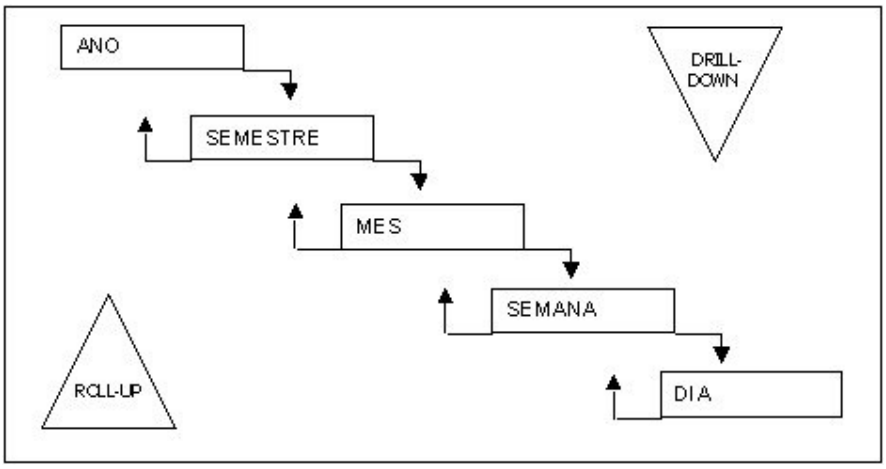


Figura 9: Exemplo de *drill-down*, detalhando uma informação de data e *roll-up*, sintetizando a informação.

Drill-down não significa descer em uma hierarquia predeterminada. Significa a possibilidade de obter rapidamente cabeçalhos de linha de qualquer uma das dimensões associadas a uma tabela de fatos. Significa também a possibilidade de remover cabeçalhos e pesquisar em direções diferentes [KIMBALL, 1996].

Determina também o detalhamento de um consulta, as consultas são mais restritas se existirem mais detalhes nos critérios de seleção. O usuário pode querer começar sua análise no nível mais alto de agregação, e então aprofundar a pesquisa passando pelos diferentes níveis até o nível inferior de detalhe a fim de obter uma perspectiva diferente do que compôs os valores do nível mais alto. A acumulação e o aprofundamento de pesquisa são recursos úteis do ambiente OLAP para a realização da análise multidimensional, porém, diferentes níveis de uma hierarquia não representam dimensões diferentes por si mesmas, desse modo, eles não são considerados requerimentos para se fornecer capacidades multidimensionais.

A figura 10 mostra o suporte OLAP ao processamento de *drill-down*. Há dois tipos de processamento de *drill-down* relevantes: processamento de *drill-down* entre-OLAP e

processamento de *drill-down* OLAP-para-dados estruturados organizacional. O *drill-down* entre-OLAP é usado para mostrar o relacionamento de resumo entre as diferentes instâncias de dados dentro do ambiente OLAP, é também conhecido como *drill-across*. Dados mais detalhados existem no nível estruturado organizacional do d*ata warehouse*, o que dá suporte a um nível de *drill-down* que vai além do projeto de cada instância de OLAP departamental [INMON, 1999].

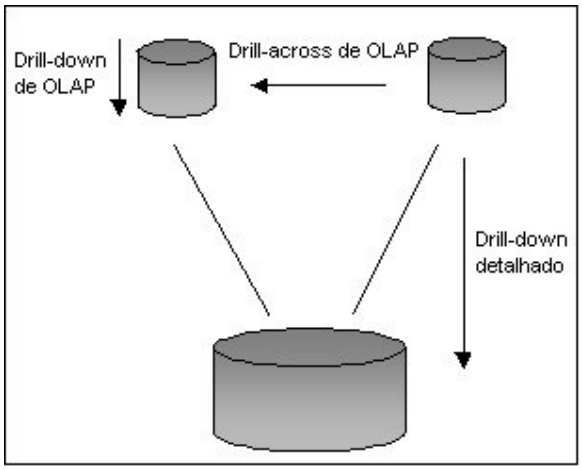


Figura 10: *Drill-down* e *drill-across* de OLAP.

O *roll-up* é o processo inverso do *drill-down***,** permite que o usuário reduza o escopo da análise. Ele sobe o nível de detalhe, ou seja, incrementa o nível de agregação.

2.2.7 Dimensões Descaracterizadas

As dimensões descaracterizadas, também conhecidas como dimensões degeneradas, são tipos de atributo normalmente empregados em tabelas de fatos onde a granularidade da tabela representa uma transação propriamente dita ou uma linha de item da transação, e pode ser usada para agrupar itens de linha em uma única ordem.

A dimensão descaracterizada representa um identificador do registro. São representadas na tabela de fatos como chaves de dimensão sem que exista a tabela de dimensão. Muitas vezes dimensões descaracterizadas compõem a chave primária da tabela de fatos junto com as chaves estrangeiras das outras dimensões. A figura 11 ilustra um exemplo de dimensão descaracterizada na tabela de fatos Vendas, com o atributo número da venda. Como as outras informações, como Cliente e Data, já estão em outra dimensão, o número da venda se torna uma dimensão descaracterizada.

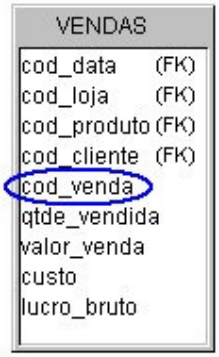


Figura 11: Dimensão descaracterizada “código da venda” na tabela de fatos Vendas.

2.3 Esquema Floco de Neve

O esquema floco de neve é uma variação do esquema estrela, no qual todas as tabelas de dimensão são normalizadas na terceira forma normal (3FN), ou seja, são retirados das tabelas os campos que são funcionalmente dependentes de outros campos que não são chaves. Recomenda-se utilizar o esquema floco de neve apenas quando a linha de dimensão ficar muito longa e começar a ser relevante do ponto de vista de armazenamento. A figura 12

representa a implementação do modelo floco de neve no modelo do sistema de vendas da loja de departamentos apresentado no tópico 2.1.

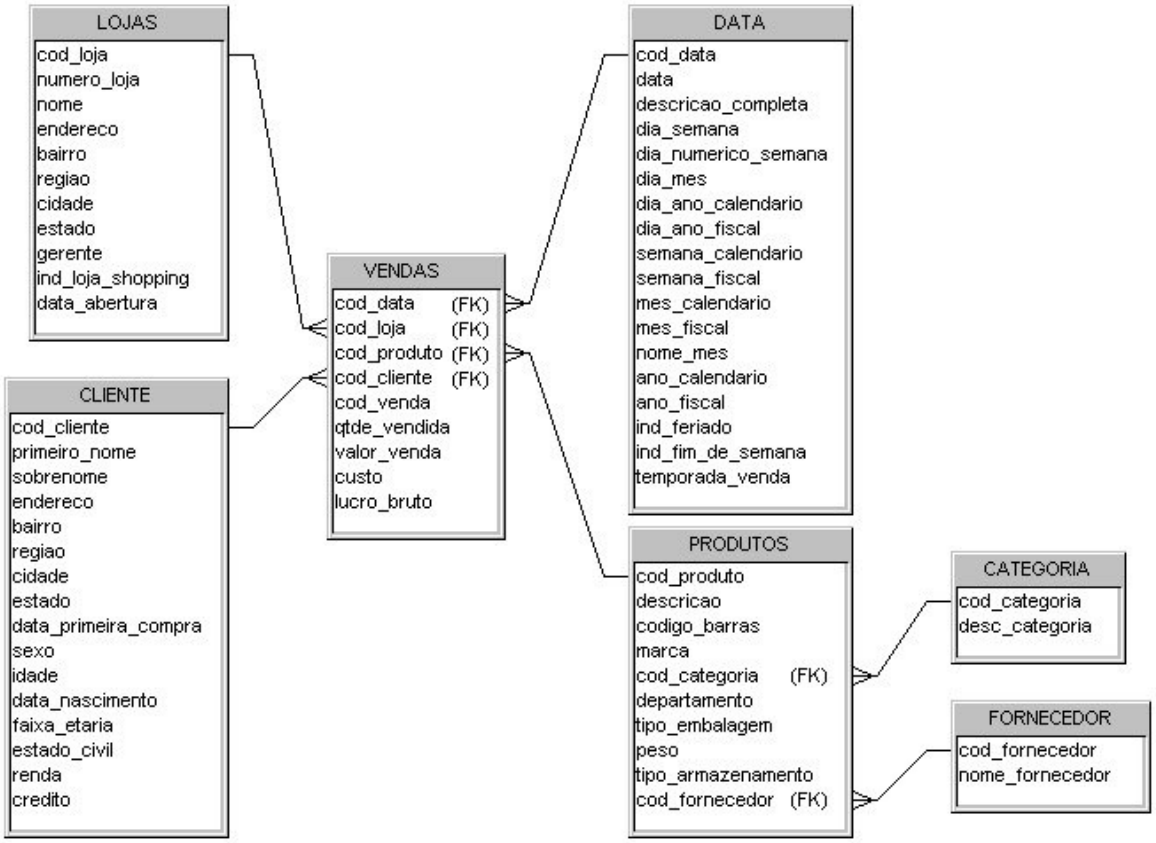


Figura 12: Modelo de Dados do sistema de vendas de lojas de departamento utilizando o esquema floco de neve.

Se as tabelas de dimensão forem normalizadas podem surgir alguns problemas. A complexidade do modelo de dados aumenta e, conseqüentemente, diminui a compreensão desse modelo por parte dos usuários.

A implementação do floco de neve frustra o uso de esquemas de indexação mais eficientes como o índice de mapa de bits [KIMBALL, 2002]. Esses índices são muito úteis para indexar campos de baixa cardinalidade, agilizam muito o desempenho de uma consulta ou restrição à única coluna em questão, sendo assim ideais para tabelas desnormalizadas. Esse

esquema inevitavelmente interfere na possibilidade de exploração dessa técnica de ajuste de desempenho.

O desempenho durante a fase de atualização dos dados do *data warehouse* será melhor com as tabelas normalizadas, mas isso não é tão importante em sistemas de apoio à decisão, uma vez que esta operação é feita normalmente durante a noite ou em momentos em que não estejam sendo executadas consultas por parte dos usuários. Mesmo assim, alguns projetistas utilizam o argumento de melhorar este desempenho para justificarem a necessidade de normalizar as dimensões.

Já a performance das consultas realizadas pelos usuários tende a diminuir devido o aumento do número de junções, e essa sim pode ser crucial para o sucesso do ambiente visto que os usuários que esperam retornos cada vez mais rápidos.

Ralph Kimball [1996] aconselha os projetistas "bem-intencionados" a resistirem à tentação de transformar esquemas estrela em esquemas floco de neve, devido ao impacto da complexidade deste tipo de estrutura sobre o usuário final, enquanto que o ganho em termos de espaço de armazenamento seria pouco relevante.

A decisão de optar pelo esquema estrela ou pelo floco de neve deve ser tomada levando-se em consideração o volume de dados, o SGBD, as ferramentas utilizadas, etc.

Outra possibilidade é mesclar o esquema estrela com o esquema floco de neve, normalizando apenas algumas dimensões. Essas dimensões normalizadas podem ou não ter um relacionamento direto com a tabela de fatos. A figura 13 mostra como ficariam relacionadas as tabelas no caso do modelo ter um relacionamento direto das dimensões normalizadas com a tabela de fato.

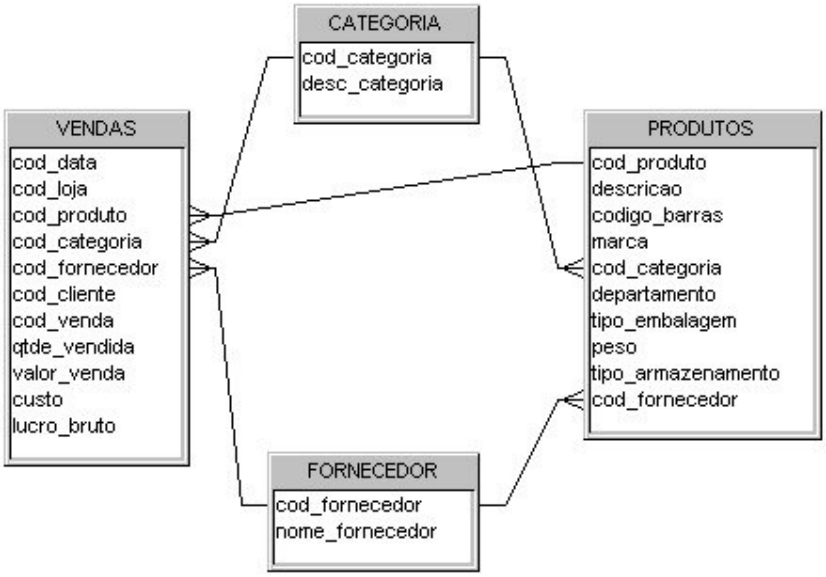


Figura 13: Modelo de dados floco de neve com as dimensões normalizadas se relacionando diretamente com a tabela de fatos.

Dessa forma, pode-se economizar com espaço de armazenamento e melhorar a performance em alguns tipos de consultas, do que se fosse utilizado o esquema floco de neve puro. Por exemplo, no caso de se efetuar uma análise de vendas por categoria.

2.4 Cubo

Uma idéia fundamental da modelagem dimensional é que quase todos os tipos de dados de negócio podem ser representados por um tipo de cubo de dados, onde as células deste cubo contêm valores medidos e os lados do cubo definem as dimensões dos dados. Pode-se ter mais que três dimensões, tecnicamente chamado de hipercubo, apesar de normalmente os termos cubo e cubo de dados serem usados como sinônimos de hipercubo [KIMBALL et al, 1998].

Cubo é a estrutura multidimensional de dados que expressa a forma na qual os tipos de informações se relacionam entre si. É formado pela tabela de fatos e pelas tabelas de dimensão que a circundam e representam possíveis formas de visualizar e consultar os dados. O cubo armazena todas as informações relacionadas a um determinado assunto, de maneira a permitir que sejam montadas várias combinações entre elas, resultando na extração de várias visões sobre o mesmo tema.

Por exemplo, voltando ao exemplo da rede de lojas de departamento, podemos considerar o total de vendas de um determinado produto em um período do ano (como um mês) em uma loja específica. Nós podemos imaginar estes dados através de um cubo tridimensional, onde uma dimensão representa os produtos disponíveis, a outra o período e a última as lojas da rede, como mostrado na figura 14:

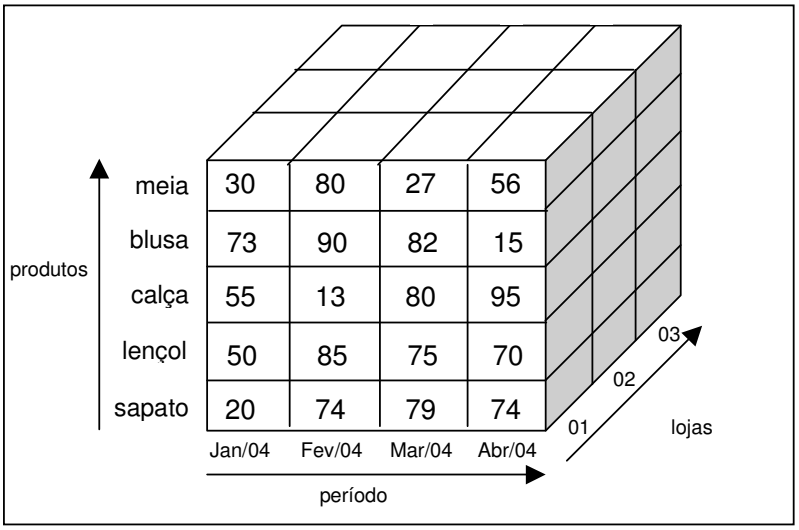


Figura 14: Exemplo de um cubo aplicado ao sistema de vendas de uma rede de lojas de departamento

No exemplo acima, demonstramos um cubo de três dimensões. Se acrescentarmos mais uma dimensão (cliente, por exemplo), teremos então um hipercubo de quatro dimensões.

Para se visualizar a análise multidimensional em cubo utiliza-se a técnica de *slice* e *dice*, ou seja, fatiar e cortar o cubo separando partes de um cubo [INMOM, 1999]. O uso integrado dos conceitos *slice* e *dice* permite rotacionar os lados de um cubo de dados (dimensões) em qualquer sentido, possibilitando a combinação de quaisquer dimensões e a obtenção de informações correspondentes sobre vários enfoques.

Entende-se que com a modelagem dimensional é possível melhorar desempenho de consultas e facilitar análises através das medidas armazenadas nas tabelas fatos e das descrições das dimensões. Esse capítulo apresentou os conceitos e terminologias da modelagem dimensional, apresentou as tabelas de fatos e tabelas de dimensão e os esquemas estrela e floco de neve. Porém existem técnicas para se desenvolver o modelo dimensional e melhorar seu desempenho. Essas técnicas serão apresentadas no próximo capítulo.

Capítulo 3 - Técnicas de Modelagem Dimensional

As técnicas de modelagem dimensional, ou multidimensional, representam uma maneira de se desenvolver um modelo dimensional. Estas técnicas têm o propósito de gerar um modelo dimensional que seja correto do ponto de vista do negócio e apresente um bom desempenho para a execução de consultas. A seguir são apresentadas algumas técnicas interessantes de implementação da modelagem dimensional.

3.1 Granularidade

A mais importante questão de projeto que o desenvolvedor do *data warehouse* precisa enfrentar, refere-se à definição da granularidade do *data warehouse*, ou seja, o nível de detalhe ou de resumo dos dados existentes no *data warehouse*.

Quando a granularidade de um *data warehouse* é apropriadamente estabelecida, os demais aspectos de projeto e implementação fluem tranqüilamente; quando ela não é estabelecida, todos os outros aspectos se complicam [INMON, 1997].

A razão pelo qual a granularidade é a principal questão de projeto consiste no fato de que ela afeta profundamente o volume de dados que residem no *data warehouse* e, ao mesmo tempo, afeta o tipo de consulta que pode ser atendida.

Com um nível de granularidade mais alto o volume de dados é bem menor e menos índices serão necessários, como a quantidade de registro é geralmente bem grande, a força de processamento necessária para acessar os dados é um fator importante.

O problema é que à medida que o nível de granularidade aumenta, o número de consultas que podem ser atendidas diminui, sendo que numa granularidade mínima as consultas mais detalhadas podem ser respondidas.

3.1.1 Níveis duais de granularidade

As organizações geralmente buscam uma eficiência de armazenamento e acesso a dados, assim como querem ter a possibilidade de consultar os dados em maior detalhe. Por isso, deve-se pensar em dois, ou mais, níveis de granularidade no *data warehouse*, ou seja, níveis duais de granularidade.

Um ambiente com níveis duais de granularidade consiste em ter dados a respeito de um mesmo assunto em granularidades diferentes. Por exemplo, em um sistema bancário, pode-se armazenar os lançamentos individuais em contas correntes nos últimos 60 dias, e armazenar o histórico resumido desses lançamentos nos últimos 5 anos, com o valor total de lançamentos sumarizados por mês. Tanto os dados resumidos, quanto os detalhados estarão disponíveis para o usuário.

O ponto de partida para a definição do nível de granularidade apropriado consiste em se fazer uma estimativa do número de linhas de dados que o *data warehouse* conterá. Se para o primeiro ano ultrapassar o total de 1.000.000 de linhas, níveis duais de granularidade se farão necessários [INMON, 1997].

Níveis duais de granularidade permitem que você processe eficientemente a enorme quantidade de solicitações e atenda a qualquer questão que possa ser respondida. Essa é a melhor de todas as situações e deveria ser a opção de projeto padrão [INMON, 1997].

3.1.2 Tabelas Agregadas

Em aplicações de análise de dados, um dos fatores mais críticos é o tempo de resposta ao usuário devido ao grande volume de dados envolvido nas consultas desse tipo de aplicação. Uma técnica utilizada para obter ganhos significativos de performance é a criação de tabelas agregadas, esse é um dos principais recursos para ajuste de desempenho de *data warehouses*. Consiste em criar novas tabelas com os dados da tabela de fatos, mas alterando a granularidade da mesma, gerando assim tabelas bem menores, com dados sumarizados.

É evidente que se o usuário quiser fazer um *drill-down* na informação que estiver analisando, isto é, detalhar mais a informação, ele acabará tendo que usar tabelas não-agregadas, arcando, portanto, com um tempo maior de resposta.

Todo *data warehouse* deve conter tabelas de agregação pré-calculadas e pré-armazenada. Como deve ser evitada a granularidade mista na tabela de fatos, cada agregação de tabela de fatos deve ocupar sua própria tabela de fatos física. A figura 15 mostra algumas agregações que poderiam ser feitas da tabela de fatos Venda.

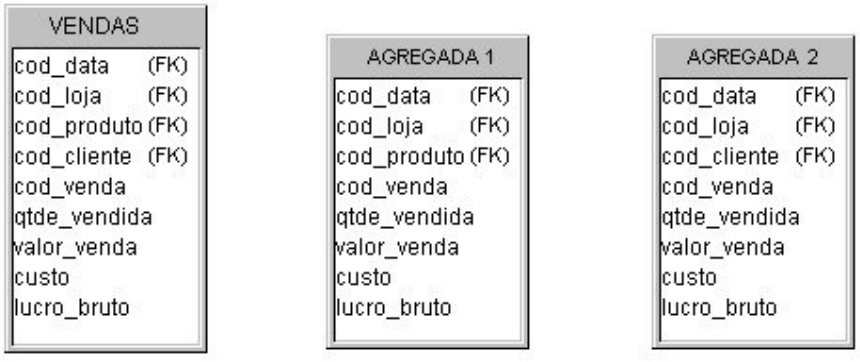


Figura 15: Tabela de fatos Vendas e duas tabelas agregadas originadas da agregação da tabela Vendas.

É importante avaliar bem o ambiente para definir quais agregações devem ser criadas. É preciso considerar o modelo de dados (relacionamentos, hierarquias e cardinalidades), realizar estatísticas de consultas para descobrir quais os requisitos mais freqüentes e quais consultas são mais críticas, e assim definir quais agregadas serão mais úteis, mais utilizadas.

É impraticável pensar na criação de todas as combinações de agregação em potencial. A utilização de tabelas agregadas requer um esforço adicional de manutenção, além de aumentar o gasto com armazenamento, por isso deve-se sempre tentar criar agregadas que correspondam a múltiplas consultas.

Outro ponto a ser considerado é a distribuição estatística dos dados, ou seja, deve ser avaliado quantas instâncias exclusivas existem em cada nível da hierarquia e qual a compactação será obtida ao se passar para o nível seguinte. Por exemplo, se uma tabela tiver 50 produtos, e esses fossem agrupados em 10 marcas, estaremos resumindo 5 linhas da tabela base (na média) para calcular a agregação de marca. Nesse caso não vale o esforço de pré-armazenar a agregação fisicamente. Por outro lado, se podemos evitar a consulta de 100 linhas base acessando a agregação, isso já pode ser vantajoso.

O uso de tabelas agregadas pode ser temporário, por exemplo, se determinadas análises são feitas freqüentemente apenas em um determinado período, essas tabelas podem ser criadas apenas nesse período, economizando com armazenamento e manutenção. Ou ainda tabelas que serão utilizadas apenas uma vez, em análises especiais.

A utilização dessas tabelas deve ser sempre monitorada para verificar se realmente está se obtendo vantagem com a sua existência. O benefício gerado pela criação de uma agregada é calculado baseado na redução do volume de dados e na freqüência de sua

utilização. Deve-se sempre levar em conta a possibilidade de uma agregada ser extinta e a manutenção que isso causaria a todo o processo.

Cada nova carga de dados no *data warehouse* acarretará no recálculo de toda ou pelo menos parte das tabelas agregadas, para que ela contemple os novos dados incluídos. Existem duas formas de atualizar uma tabela agregada:

- Agregação completa: consiste em recriar a tabela agregada toda vez que houver uma carga na tabela de fatos;

- Agregação incremental: os dados existentes na tabela são alterados e apenas os dados novos são inseridos.

Quando a agregação não for completa, mecanismos de limpeza precisam ser desenvolvidos para que os dados mais antigos sejam excluídos. O período de retenção de uma tabela agregada nem sempre é o mesmo que o da tabela de fatos. Muitas empresas adotam um período maior na tabela agregada, pois os dados estão sumarizados e gastam menos com o armazenamento. Por exemplo, uma tabela de fatos tem dados com visão diária de três meses, mas o usuário pode acessar os dados dos últimos doze meses em uma visão mensal.

A decisão por qual tabela agregada utilizar para responder cada consulta não é uma tarefa tão simples, pois muitas vezes a consulta pode ser executada em mais de uma tabela agregada retornando o mesmo resultado. Em algumas ferramentas analíticas, durante o desenvolvimento do projeto são definidas as possibilidades de tabelas a serem utilizadas e a prioridade de cada uma, sendo a escolha feita baseada nos campos que o usuário utilizar na a consulta. Existem também ferramentas que determinam automaticamente qual tabela será utilizada na consulta, baseados no nível de granularidade de cada uma.

Outra forma de implementar agregações é utilizando as visões materializadas, apresentadas a seguir.

3.2 Visões materializadas

As visões materializadas são um tipo especial de visão, que existe fisicamente dentro do banco de dados. Elas existem para melhorar o tempo de execução de consultas.

Nos ambientes de *data warehouse* as visões materializadas são usadas para o pré-cálculo e armazenamento de dados gerados, como totais e médias. Também podem ser usadas para pré-calcular relacionamentos com ou sem agregações.

O otimizador de consultas do SGBD pode utilizar a visão materializada para aumentar a performance de acesso aos dados, reconhecendo automaticamente quando uma visão materializada pode ser utilizada para determinada requisição. O otimizador transparentemente direciona a consulta para a visão materializada em vez da tabela original.

Essa característica do otimizador pode trazer muitos benefícios para ambientes com grandes volumes de dados, pois sem alterar a consulta do usuário, que utiliza tabelas com grandes volumes, o otimizador interpreta consultas e obtém o mesmo resultado mais rapidamente em uma das visões materializadas presentes no ambiente [FERNANDES, 2002].

Alguns SGBDs atualizam os dados das visões materializadas imediatamente após as alterações serem efetivadas na tabela original, não havendo necessidade de processamentos especiais durante o processo de carga do *data warehouse*.

3.3 Técnicas de Rastreamento de Alterações

No projeto de modelagem do *data warehouse*, as dimensões são normalmente projetadas para serem independentes de tempo. Infelizmente isso não reflete o que acontece no mundo real. Apesar de não haver mudanças constantemente nas dimensões, elas não possuem um valor sempre fixo (o estado civil de um cliente pode alterar de solteiro para casado, por exemplo).

Durante a fase de projeto do *data warehouse*, os projetistas devem determinar quais estratégias tomar para gerenciar as alterações nas dimensões. Esse item geralmente é esquecido pelos usuários durante a preparação dos requisitos do projeto, porém é melhor estar preparado para estas situações o quanto antes para desenvolver o modelo de dados adequado a essas mudanças.

É preciso analisar cada atributo das dimensões, prevendo quais atitudes tomar se (e quando) o valor deste atributo mudar no mundo real [KIMBALL, 2002], e analisar qual é a melhor alternativa para o modelo.

3.3.1 Sobrescrever o valor

Com essa abordagem, o valor antigo é simplesmente substituído pelo novo valor na tabela de dimensão responsável por guardar este dado, sem afetar a tabela de fatos. Essa é a solução de implementação mais simples e é útil em situações de acertos no cadastro (correção de informações erradas, por exemplo), ou quando não é importante armazenar o valor antigo do atributo alterado. Um exemplo é a alteração da categoria do endereço do cliente José

Aguiar da Rua Pedro Álvares Cabral, 299 na Vila Barros para Rua XV de Novembro, 14 no Centro, ilustrado na figura 16 com alguns dos campos da tabela Clientes.

cod_cliente	primeiro_nome	sobrenome	endereco	bairro
241	José	Aguiar	Rua XV de novembro, 14	Centro

Figura 16: Alteração de dimensão sobrescrevendo o valor antigo.

Porém essa solução não mantém o histórico das informações, e os fatos perdem esse acompanhamento. Caso haja alguma agregação ou consulta criada baseada no antigo valor deste atributo precisará ser refeita para que esta referencie o novo valor, do contrário ela passará a não encontrar mais registros com o valor utilizado.

3.3.2 - Adicionar uma nova linha na tabela de dimensão

Com esta abordagem, é adicionado um novo registro para a dimensão que foi alterada, contendo uma nova chave e o novo valor. Por exemplo, na alteração da residência de um cliente, seria inserido um novo registro desse cliente, com uma nova chave e o endereço alterado.

Essa alternativa permite haver quantas alterações quantas forem necessárias na tabela de dimensão, porém pode ocorrer um crescimento muito grande dessa tabela, não sendo uma alternativa muito adequada quando se atinge o patamar de milhões de linhas de dimensão [KIMBALL, 2002].

Uma complicação dessa abordagem é a necessidade de utilização de chaves substitutas (não poderia ser usado o RG do cliente, por exemplo), já que haverá dois registros referenciando o mesmo item.

Nas necessidades de busca de todas as Vendas para esse cliente (independente de sua alteração), bastaria restringir a consulta pelo RG do cliente, ou pelo seu nome, retornando então todos os fatos que referenciam o item.

No processo de carga do *data warehouse*, todos os novos fatos que referenciem essa dimensão alterada precisam conter a chave da nova dimensão, ocasionando numa partição natural da tabela de fatos cronologicamente. Essas chaves substitutas devem estar descritas nos metadados e serem tratadas pelas aplicações do usuário final, que devem considerar que se trata do mesmo item, que sofreu uma alteração em algum atributo.

A figura 17 ilustra a alteração do endereço de um cliente através da inserção de um novo registro na dimensão (somente são mostrados alguns atributos da tabela).

cod_cliente	**primeiro_nome**	**sobrenome**	**endereco**	**bairro**
241	José	Aguiar	Rua Pedro Álvares Cabral, 299	Vila Barros
854	José	Aguiar	Rua XV de novembro, 14	Centro

Figura 17: alteração de dimensão por inserção de novo registro.

Para a detecção e manipulação das alterações durante a carga das dimensões, é interessante adicionar um campo na tabela de dimensão contendo o resultado de um algoritmo de *checksum* em cada registro. Quando os novos registros forem carregados, executa-se o mesmo algoritmo nestes registros, caso o resultado de ambos seja igual, significa que o registro não foi alterado e não precisa de atualização na tabela de dimensão. Caso tenha diferença, significa que houve alteração e que é preciso adicionar uma nova dimensão para este item, com nova chave.

3.3.3 Adicionar uma nova coluna de dimensão

Apesar de haver o particionamento do histórico com a abordagem anterior, ela não permita a associação do novo valor com o valor antigo da dimensão. Muitas vezes, porém, os usuários podem querer buscar os fatos como se a alteração não tivesse ocorrido.

Por exemplo, a pesquisa por vendas para clientes do bairro "Centro" só retornaria dados a partir da data em que o novo valor entrou em vigor, e o usuário pode querer saber como as vendas do dia se comportariam sob a estrutura do dia anterior, com os dados antigos.

Para resolver esse tipo de necessidade, uma solução é adicionar uma nova coluna na tabela de dimensão, contendo o valor inicial de determinado atributo, e um novo atributo com a data em que o novo valor entrou em vigor. Essa data é necessária porque com essa abordagem a chave da dimensão permanece inalterada, sendo através da data a única maneira de identificar a alteração.

Essa solução é mais complexa do que as duas anteriores, e torna necessária a manutenção de dois campos para um atributo: um com o valor corrente, e outro com o valor original. Essa solução também não atende o objetivo de acompanhar o histórico dos dados, pois não há o armazenamento dos valores intermediários, apenas do atual e do original.

3.3.4 Artefatos de dados

Esta técnica é usada para preservar o relacionamento dos dados quando um ou mais atributos são dinâmicos por natureza [INMON, 1999]. Consiste na criação de uma tabela de dimensão adicional com os dados que são alterados com o tempo, para manter o histórico dos

registros. Essa nova tabela contém a chave original do produto, a data de alteração, e os demais atributos que são dinâmicos.

A cada alteração na dimensão em questão, um novo registro é adicionado nessa tabela de histórico, conforme mostra a figura 18.

Cliente

cod_cliente	**241**
data_status	14/05/2004
primeiro_nome	José
sobre_nome	Aguiar

Histórico Cliente

cod_cliente	**241**	**241**	**241**
data_alteracao	15/02/2002	18/03/2003	14/05/2004
endereco	Rua Salvador, 344	Rua Pedro Álvares Cabral, 299	Rua XV de Novembro, 14
bairro	Jardim Aurora	Vila Barros	Centro

Figura 18: Rastreamento de alterações por meio de artefatos de dados.

Essa abordagem difere da abordagem que prega a inserção de um novo registro na tabela de dimensão porque alguns dados são dinâmicos por natureza, enquanto outros são estáticos. Capturar todos os dados de uma dimensão a cada alteração em um atributo iria gerar redundância demais. Outra vantagem dessa abordagem é a persistência da chave original do item da dimensão.

3.4 Criação de Minidimensões

No gerenciamento de alterações nas dimensões, encontramos alguns problemas quando uma tabela é muito grande e sofre alterações freqüentemente. A utilização da técnica de adicionar uma nova linha na tabela de dimensão a cada alteração se torna inviável numa

tabela que já contém milhões de registros, e sobrescrever tais valores faria com que o histórico dessas alterações fosse perdido. A criação de minidimensões é uma técnica que permite vencer os desafios de análise de desempenho e controle de alterações em tabelas de dimensão muito grandes e que mudam rapidamente.

A técnica de criação de minidimensões consiste em dividir uma dimensão muito grande em uma dimensão separada ou em minidimensões compostas por pequenos conjuntos de atributos que estão sempre sendo mudados ou analisados [KIMBALL, 2002]. Esses atributos devem estar estruturados para conter um número limitado de valores.

Por exemplo, no caso da tabela de Clientes, pode-se criar uma minidimensão separada por atributos demográficos como idade, sexo, número de filhos e renda familiar, partindo do princípio que essas colunas seriam muito utilizadas em análises.

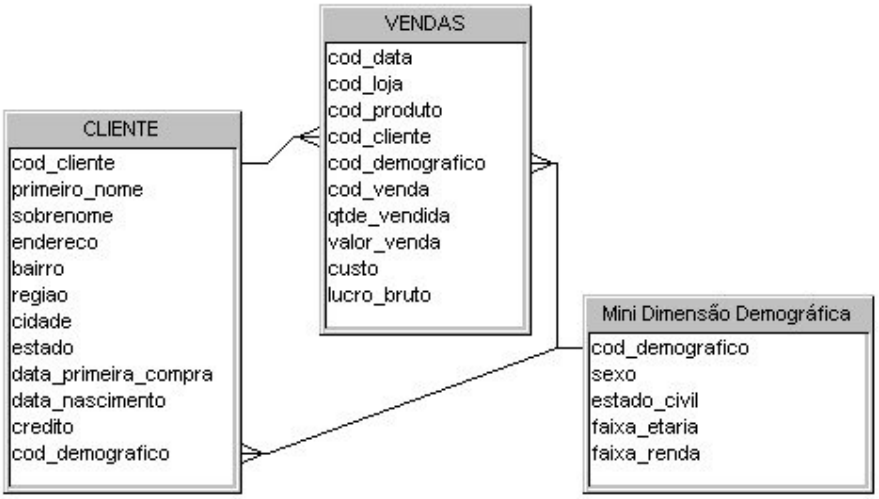


Figura 19: Minidimensão DEMOGRÁFICA.

Cada registro da tabela de fatos deve conter duas chaves externas relacionadas a essa dimensão, a chave da dimensão normal e a chave da minidimensão, como mostra a figura 19.

A presença da chave da minidimensão garante um bom desempenho em consultas por atributos demográficos.

Quando se cria uma minidimensão, os atributos devem assumir um número relativamente pequeno de valores discretos, por isso, os atributos que mudam com freqüência, como idade e receita, devem ser associados em faixas de valores. A utilização de faixas com associações é provavelmente o compromisso mais importante associado à técnica da minidimensão porque, depois que decidido sobre as associações de valor, será difícil o registro da tabela dimensão mudar para um conjunto diferente de associações mais adiante. A figura 20 ilustra como ficariam as linhas de uma minidimensão de dados demográficos.

CHAVE DE DADOS DEMOGRÁFICOS	ESTADO CIVIL	IDADE	SEXO	RENDA FAMILIAR
1	Solteiro	18-22	Masculino	< R$2.000,00
2	Casado	18-22	Masculino	< R$2.000,00
3	Divorciado	18-22	Masculino	< R$2.000,00
4	Solteiro	18-22	Masculino	R$2.000,00- R$4.000,00
5	Casado	18-22	Masculino	R$2.000,00- R$4.000,00
20	Solteiro	30-35	Feminino	R$5.000,00- R$8.000,00

Figura 20: Exemplo de linhas de uma minidimensão de dados demográficos.

No caso do usuário precisar acessar o valor dos dados brutos específicos, ele deverá ser incluído também na tabela de fatos além de ser representado como uma associação de valor na minidimensão de dados demográficos, isso atenderia o usuário em consultas específicas como essa e seria performático em análises mais globais.

Uma outra vantagem da presença da chave da minidimensão na tabela de fatos é que os perfis demográficos históricos de um cliente, por exemplo, podem ser levantados a qualquer momento consultando a tabela de fatos. A tabela de dimensão armazena apenas o valor de chave da minidimensão referente aos valores atuais, não há necessidade de armazenar os valores antigos, se isso fosse na própria tabela de dimensão o problema de aumentar muito a tabela que já é muito grande continuaria a existir.

3.5 Criação de Novas Chaves

Também conhecidas como *chaves sem significado*, *chaves inteiras*, *chaves não naturais*, as chaves substitutas são, basicamente, um valor inteiro, atribuído seqüencialmente a cada registro inserido, conforme a necessidade. Por exemplo: o primeiro produto inserido na dimensão Produto teria a chave "1", o próximo teria a chave "2", e assim sucessivamente. Sua funcionalidade é exclusivamente para o relacionamento das tabelas de dimensão com a tabela de fatos [KIMBALL, 2002].

Manter uma chave substituta para as dimensões em vez de utilizar as chaves naturais operacionais é vantajoso porque isso livra o ambiente *data warehouse* de alterações operacionais. Manter uma chave sem significado possibilita ao *data warehouse* ser independente de alterações nos sistemas transacionais para manter a compatibilidade dos dados.

Um exemplo de chave substituta é o atributo cod_loja na tabela de dimensão "Lojas". Enquanto a chave original do sistema transacional é o número da loja (numero_loja),

no ambiente *data warehouse* foi criada a chave substituta cod_loja para evitar o gerenciamento de alterações na tabela de dimensão lojas, conforme ilustra a figura 21.

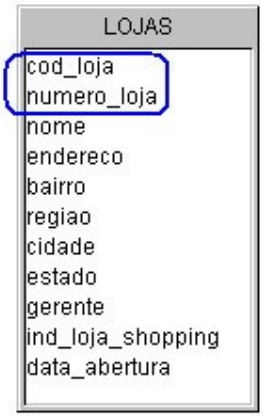


Figura 21: Chave substituta "código da loja" na tabela de dimensão Lojas no lugar da chave original "número da loja".

Outra vantagem de chaves substitutas é a performance. Muitas vezes as chaves operacionais são volumosos textos alfanuméricos. Para manter o relacionamento com a tabela de fatos, muito espaço em disco é desperdiçado por causa do tamanho destas chaves. No entanto, uma chave numérica de 4 bytes consegue armazenar aproximadamente 2 bilhões de valores, o suficiente para guardar todos os registros de uma tabela de dimensão.

A figura 22 representa a utilização da chave substituta cod_produto utilizada no lugar da chave original codigo_barras na tabela de dimensão Produtos. Ao se utilizar a chave substituta (de 4 bytes) neste caso, é possível economizar cerca de 10 bytes por registro da tabela de fatos (economia de aproximadamente 1Gb a cada 1 bilhão de registros).

Figura 22: Chave substituta "código do produto" para economizar espaço em disco da chave original "código de barras".

Como alternativa à chave inteira seqüencial, pode-se usar a chave operacional do item, adicionando-se dois ou mais dígitos ao final para indicar a versão do item [INMON, 1997]. Essa alternativa não resolve o problema da performance, já que o tamanho da chave ficará ainda maior do que a chave original, porém ajuda a rastrear as modificações geradas nos itens através da chave primária, gerando um controle de versão de fácil manipulação.

3.6 Tratamento de Dimensões e Fatos com Cardinalidade M:N

Segundo Kimball [KIMBALL et al, 1998], apesar de, normalmente, a cardinalidade entre as dimensões e a tabela de fatos ser 1:N (um-para-muitos), podem acontecer casos em que essa cardinalidade seja M:N (muitos-para-muitos). A identificação deste tipo de cardinalidade empregando um desenvolvimento baseado apenas em técnicas dimensionais não é uma tarefa trivial. Neste tipo de desenvolvimento as dimensões e a própria tabela de fatos são definidas de acordo com a especificação do usuário final. Para identificar a existência da

cardinalidade M:N é necessária uma análise mais minuciosa, cruzando o modelo gerado com as regras do negócio [SOARES, 1998].

Na abordagem de Kimball [KIMBALL et al, 1998], após a definição do modelo dimensional é necessária uma análise de cada dimensão existente, verificando-se a cardinalidade de seu relacionamento com a tabela de fatos. O propósito desta análise é identificar as dimensões que apresentem um relacionamento M:N com a tabela de fatos.

Quando esse tipo de cardinalidade é identificado, Kimball [KIMBALL et al, 1998] recomenda a adição de uma nova dimensão que representará uma ponte entre a dimensão original e a tabela de fatos. Essa dimensão ponte representa efetivamente a cardinalidade M:N, tendo além de sua chave normal, uma chave que permite identificar o conjunto de informações. Essa chave, única para um conjunto, será inserida na tabela de fatos. Dessa forma, será possível realizar as consultas pela dimensão origem, garantindo que não haverão repetições na tabela de fatos. Essa nova dimensão, mais simples, deve permitir o desenvolvimento de consultas a partir dela para a tabela de fatos.

3.7 Tabela de fatos sem fatos

Uma das premissas na criação e população da tabela de fatos é a obrigatoriedade das medidas serem preenchidas. Não se pode ter um registro na tabela de fatos com as medidas contendo o valor zero, pois isso iria aumentar bastante a quantidade de registros.

No *data warehouse* de exemplo que registra Vendas, pode haver a necessidade de saber quais produtos não tiveram nenhuma venda em determinada data, em determinada loja.

Não se pode inserir um registro com essas três dimensões e preencher as medidas com zeros.

Para suprir necessidades de registrar um fato onde não temos medidas (nada aconteceu), são utilizadas tabelas de fatos sem fatos. Uma tabela de fatos sem fatos consiste basicamente de uma tabela que servirá de relacionamento entre todas as dimensões necessárias para atender a uma necessidade básica, porém não contendo nenhuma medida. Seus registros apenas relacionam as dimensões relacionadas a ela. A tabela de fatos sem fatos tem como linhas todo o universo possível de relacionamento entre as dimensões envolvidas, sem nenhum atributo de medida.

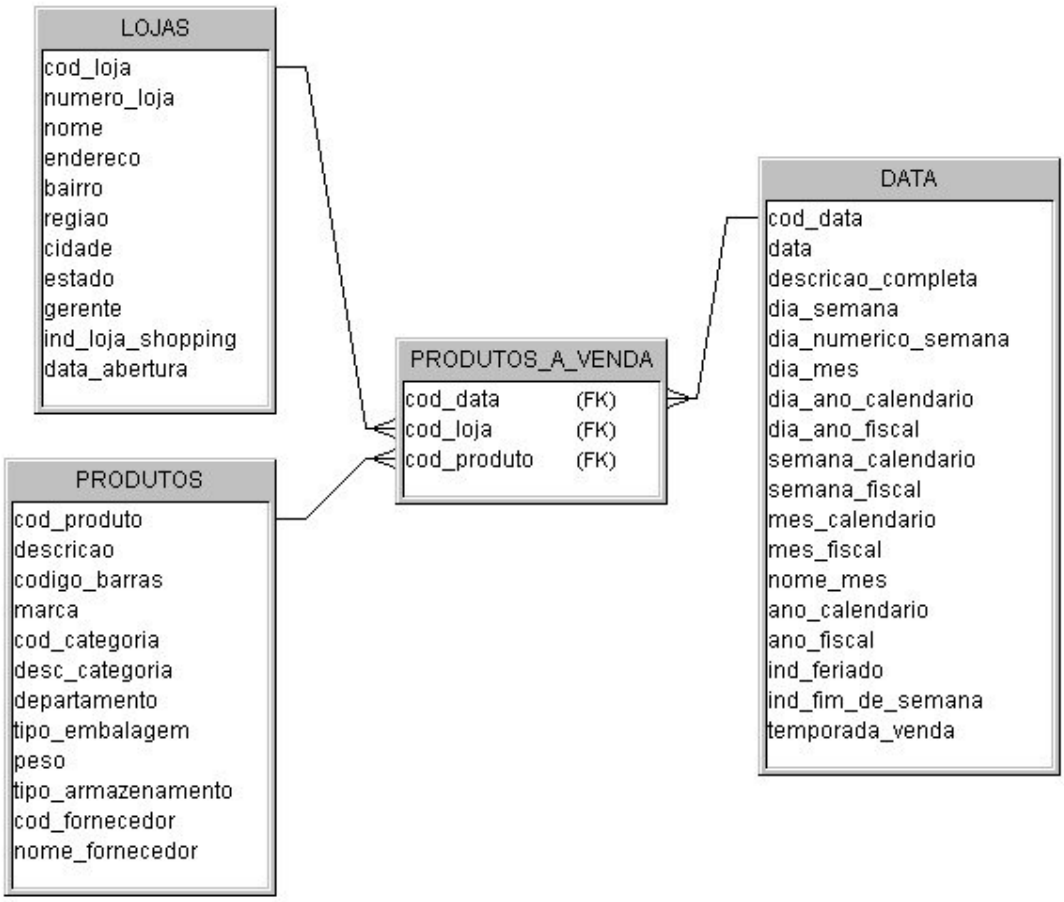


Figura 23: Tabela de fatos sem fatos “Produtos a Venda”.

No caso dos produtos que não foram vendidos, seria criada uma tabela fato sem fatos chamada Produtos a Venda. Essa tabela possui apenas os atributos para relacionamento com as tabelas de dimensão Produtos, Lojas e Data. É importante notar que ela não possui relacionamento com a tabela de dimensão Clientes, e também não possui a dimensão descaracterizada Venda, já que a granularidade desta tabela é o produto por loja por data, conforme mostra a figura 23.

Essa tabela é então preenchida com todo o universo de produtos disponíveis para venda em cada loja, a cada dia, contendo os registros com esses relacionamentos. Para saber quais produtos não foram vendidos, basta consultar na tabela Produtos a Venda o universo de produtos nas datas/lojas desejadas e, após consultar a tabela de fatos Vendas com as mesmas condições, a diferença entre os dois resultados é a resposta ao problema.

Como a tabela de fatos sem fatos armazena um registro para cada associação possível entra as dimensões, ela geralmente contém uma quantidade de registros muito grande. Não é necessário existir uma relação entre a tabela de fatos sem fatos e a tabela de fatos, podendo, no nosso exemplo, a granularidade da tabela de fatos sem fatos ser produtos disponíveis por semana, ao invés de usar granularidade diária. Isso reduziria a quantidade de registros da tabela, porém continuaria a atender à necessidade.

Uma tabela de fatos sem fatos também pode ser usada como uma tabela associativa entre as dimensões, onde não é necessário armazenar nenhuma medida. O registro de eventos normalmente é feito desta forma, onde não existem medidas para ser analisadas, apenas a relação entre as dimensões.

Um exemplo dessa abordagem é um *data warehouse* de uma instituição de ensino, com a necessidade de analisar os alunos, as matérias que eles cursam, e em quais campi. Para

isso, pode-se criar uma tabela de fatos sem fatos Matrícula possuindo relacionamentos com as dimensões Alunos, Matérias e Campi. Essa tabela armazenaria os registros de quais alunos cursam quais matérias, e em qual campus. Apenas estas são as informações importantes para este *data warehouse*, não existe nenhuma medida para ser armazenada e analisada, portanto a tabela de fatos somente teria as chaves das tabelas de dimensão.

3.8 Dados desnormalizados

Redundância gerenciada é uma característica do *data warehouse* que ajuda no suporte ao processamento de informações, para dar ao usuário um conjunto de dados adequados à necessidade do negócio, permitindo visualizar os dados como informações mais facilmente [INMON, 1999].

Os dados redundantes normalmente são desnormalizados armazenando-se as definições dos registros juntamente com os próprios códigos, na mesma tabela. Dessa maneira, não são necessárias repetidas junções entre tabelas de dados (tabelas de fatos) e tabelas de referência (tabelas de dimensão) para obter as definições descritivas dos dados, apenas a consulta a uma tabela já traz essas informações, ao invés de códigos confusos.

Isso permite ao usuário fazer uma consulta SQL no banco de dados qualificando sua consulta pelo código (método mais eficiente), e exibindo o resultado pelas suas descrições (método mais informativo) sem precisar juntar tabelas distintas, melhorando a performance das consultas.

No tabela de dimensão Produtos, estão armazenados os campos de código da categoria (cod_categoria) e também o campo descrição da categoria (desc_categoria). Com isso, o usuário pode limitar a consulta pelo código, enquanto que o resultado é exibido pela descrição, sem relacionar outras tabelas, conforme exemplifica a figura 24.

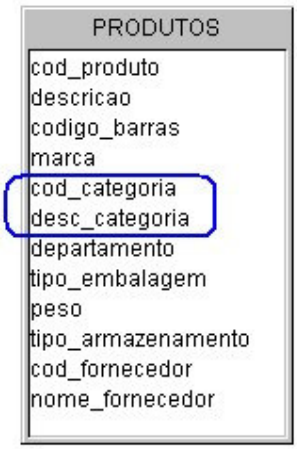


Figura 24: Dados desnormalizados na tabela de dimensão Produtos.

Deve ser aplicado um controle sobre quais dimensões são boas candidatas a serem desnormalizadas na tabela de fatos. Quando o valor é sempre preenchido na tabela de dimensão, e quando a tabela relacionada existe somente para descrever este dado, é uma boa idéia fazer a desnormalização. Porém é melhor deixar uma tabela de dimensão normalizada quando ocorrer uma das seguintes situações:

- a dimensão não é preenchida constantemente, ou seja, o dado é preenchido na tabela de dimensão somente em uma pequena porcentagem de registros [INMON, 1999];

- a dimensão armazena dados importantes além da descrição do campo, isto é, outros dados da tabela relacionada são importantes para o esquema, além da definição descritiva do dado [INMON, 1999];

- quando existe uma hierarquia que pode ser usada para diferentes modos de agregação e classificação dos dados para diferentes propósitos [INMON, 1999].

3.9 Dimensões de tempo

Todo *data mart* geralmente é uma relação de fatos baseados em um tempo, por isso é recomendado a criação de uma dimensão de tempo, como a dimensão Data criada no modelo exemplo, ao invés de armazenar a data diretamente na tabela de fatos.

Uma dimensão Data explícita é importante porque muitas vezes uma consulta SQL não permite um filtro de datas por dias de semana e finais de semana, ou por feriados, períodos fiscais, temporadas e eventos importantes. Com isso, uma tabela de dimensão Data permite armazenar todas as opções de filtragem desejadas e facilitar consultas e análises de dados.

Um exemplo de tabela de dimensão Data é a utilizada no exemplo do capítulo 2, conforme ilustrado na figura 25.

Na tabela de dimensão Data é armazenado um registro por dia, não sendo utilizada para armazenar dados de hora. Isso permite armazenar registros de um período de dez anos em uma tabela de dimensão Data, totalizando em apenas 3.650 registros, sendo uma tabela de dimensão relativamente pequena.

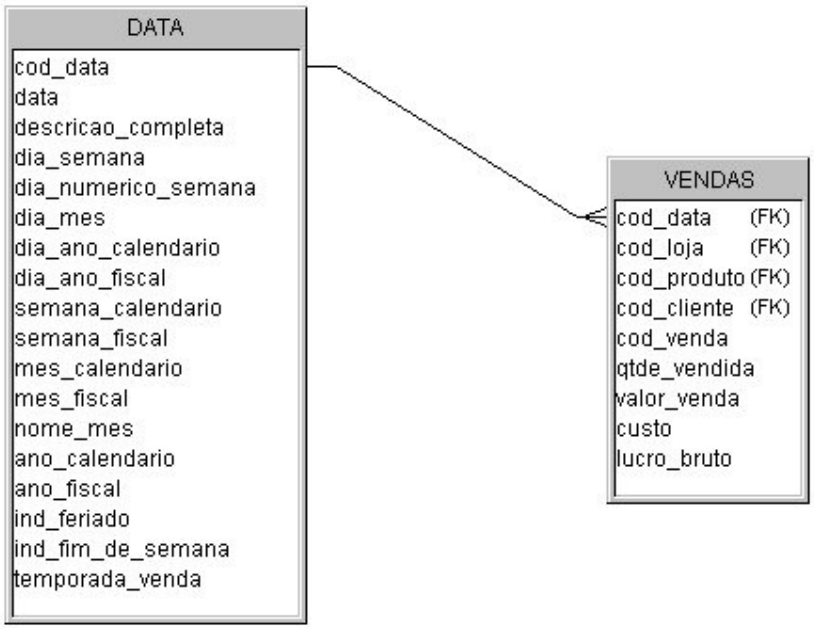


Figura 25: Tabela de dimensão Data se relacionando com a tabela de fatos Vendas no modelo do sistema de vendas de lojas de departamento, o que permite a visualização dos fatos por diversos critérios diferentes de data.

Nas necessidades de armazenar a hora da transação da análise em questão, é mais conveniente a criação de uma dimensão Hora do dia, separada da dimensão Data, contendo registros para cada minuto do dia, resultando em uma tabela com apenas 1.440 registros. Caso fosse armazenada a hora na dimensão Data, com granularidade de minutos, a quantidade de registros necessários para o armazenamento de um período de dez anos aumentaria para 5.256.000, um valor extremamente alto e difícil de manipular [KIMBALL, 2002].

A chave primária dessa tabela pode ser um campo numérico inteiro, ao invés de um campo de data. Os campos de data baseados no SQL normalmente são armazenados em 8 bytes, enquanto um campo inteiro é armazenado em apenas 4 bytes. Isso representa uma economia de 4 bytes em cada registro da tabela de fatos, onde existe uma chave estrangeira referenciando essa tabela.

Observe na figura 26 como ficariam preenchidos alguns atributos de uma dimensão Data, para as datas "16/02/2004" e a "21/04/2004".

CHAVE DA DATA	DIA DA SEMANA	DIA NUMÉRICO DA SEMANA	DIA DO MÊS	DIA DO ANO	MÊS CALENDÁRIO	NOME DO MÊS	INDICADOR DE FERIADO
1	segunda-feira	2	16	47	2	fevereiro	não é feriado
50	quarta-feira	4	21	112	4	abril	é feriado

Figura 26: Exemplo de linhas em uma dimensão Data, a primeira linha corresponde à data "16/02/2004" e a segunda a "21/04/2004".

A dimensão Data deve conter todos os atributos que podem ser utilizados para análise e seleção dos dados. É melhor armazenar todos esses atributos de tempo na tabela ao invés de usar as aplicações de acesso para converter e classificar as datas da maneira desejada.

Na dimensão Data do exemplo, foram colocados atributos como dia da semana, dia no mês, dia no ano, mês calendário, nome do mês, semana fiscal, indicador de feriado, etc. Mas uma dimensão Data pode conter uma infinidade de campos, todas as interpretações de datas úteis para o negócio devem ser armazenadas. Por exemplo, para determinado negócio pode ser interessante saber o número do dia na época, ou seja, um número de dia consecutivo começando pelo início de alguma época. A figura 27 mostra uma dimensão Data com mais alguns atributos que podem auxiliar nas análises.

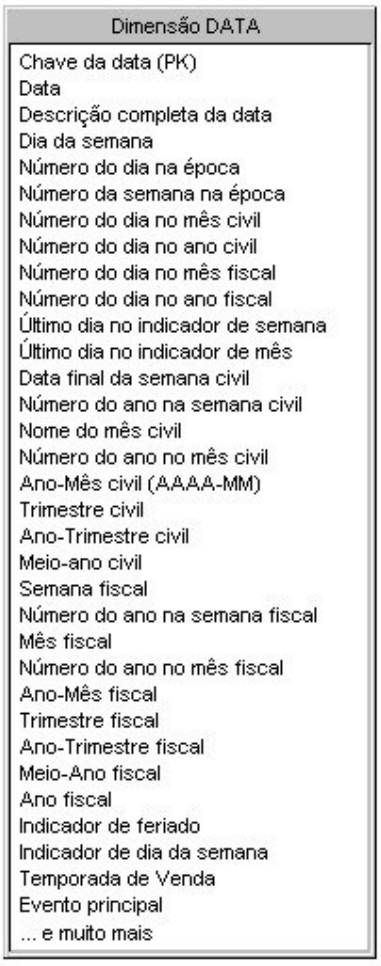


Figura 27: Dimensão Data com diversos atributos, todas as interpretações de datas úteis para o negócio devem ser armazenadas.

Entende- se que as técnicas de modelagem dimensional são importantes para gerar um modelo correto do ponto de vista do negócio e com um bom desempenho. Neste capítulo foi apresentada a importância de adotar a granularidade adequada, além de abordar tabelas agregadas, técnicas de rastreamento de alterações, minidimensões, desnormalização, entre outras técnicas.

Capítulo 4 - Questões sobre acesso a dados multidimensionais

Já foi apresentado como uma modelagem dimensional pode prover as informações necessárias a processos analíticos. Também foi visto como construir esse modelo e suas vantagens sobre o modelo relacional para este tipo de aplicações. Porém, com este modelo, uma quantidade muito maior de informações precisa ser armazenada e, conseqüentemente, recuperadas. Não é tarefa simples recuperar e atualizar as informações contidas em modelo dimensional, agregando, derivando ou simplesmente selecionando-as sem que seja necessário um tempo de processamento elevado.

As estratégias de que este capítulo trata podem ser adotadas para reduzir este tempo de processamento. Embora muitas vezes não seja possível obter um tempo de resposta consideravelmente rápido, se comparada a operações transacionais, a aplicação destas técnicas pode trazer um ganho significativo neste tempo, reduzindo o custo de processamento de muitas atividades em um ambiente de processamento analítico.

4.1 Estratégias de Processamento de Consultas

Conforme foi visto, as consultas em um sistema OLAP são muito diferentes das consultas realizadas em um sistema OLTP. Enquanto um sistema OLTP é desenvolvido para retornar resultados de um número limitado de consultas, este não é o caso nos sistemas OLAP. Os modelos dimensionais tentam abranger essas novas características. Como conseqüência, fornecedores e pesquisadores começaram a pensar em maneiras de resolver consultas envolvendo agregações, *rankings*, etc, em bancos de dados com muitos gigabytes de

dados. O uso correto de índices é uma estratégia de otimização no processamento das consultas.

Usualmente, os bancos de dados dos sistemas OLTP e os *data warehouses* residem em diferentes sistemas, sendo que não existem limitações de índices nos *data warehouses*. Pode-se criar índices em quantas colunas das tabelas forem necessárias. Em geral, os *data warehouses* podem ser indexados com índices de alta seletividade.

Por exemplo, uma consulta para saber quantas vendas houve para homens de São Paulo com escolaridade universitária, num sistema OLTP, geraria problemas com índice, já que a coluna "sexo" não poderia ser indexada, devido à sua alta seletividade. Além disso, ter índices em muitas colunas levaria a problemas de performance durante atualizações da tabela. Então, uma consulta desse tipo seria otimizada, normalmente, por apenas um índice. Em um banco de dados com bilhões de vendas para milhares de clientes, o tempo de resposta não seria aceitável.

Na consulta do exemplo, o processador de consultas precisa selecionar dados na tabela de Clientes, que vai retornar as linhas de acordo com os atributos "estado", "sexo" e "educação", e então fazer a junção com a tabela de Vendas.

Para esse tipo de problema, foram criadas novas técnicas de indexação e processamento da consulta.

4.1.1 Índices *bitmap*

Em um índice B-tree tradicional, os nós-folhas possuem um valor e a lista de posições das linhas que satisfazem esse valor. Em alguns casos, é possível utilizar uma

alternativa melhor que essa: em vez de uma lista com as posições, pode-se utilizar um vetor de bits [VAISMAN, 1998].

Um número é associado a cada linha de uma tabela (por exemplo, Clientes). Considerando a necessidade de criar um índice no atributo "educação", presumindo que haja 5 possíveis valores para esse atributo. O índice seria um vetor de bits de tamanha N (sento N a quantidade de registro em Clientes). Em cada posição desse vetor, o bit é preenchido com 1 ou 0 caso o cliente na linha em questão possua educação universitária ou não, respectivamente. Para ter um índice *bitmap*, basta criar um vetor de bits para cada valor possível do atributo, e armazenar o vetor nos nós folhas do B-tree.

Como exemplo, pode-se considerar o seguinte *bitmap*:

```
0011011001001.....1
```

Esse vetor afirma que os clientes nas linhas 3, 4, 6, 7, 10, 13 e N possuem educação universitária.

É fácil estimar o tamanho desse tipo de índice: considerando como exemplo a tabela Clientes com 250.000 linhas, e 5 possíveis valores para o atributo "educação". Um índice *bitmap* neste atributo (considerando somente os nós folhas) iria ocupar 250.000 * 5 / 8 = 0.14 mb. Um índice B-tree tradicional, como este usado no exemplo, iria ocupar 250.000 * 4kb = 0.95 mb. Essa não é uma diferença significativa na tabela de Clientes, mas em uma tabela com bilhões de linhas (como a tabela Vendas), passa a ser um ganho considerável.

O espaço necessário para índices *bitmap* é proporcional à quantidade de valores do índice (enquanto índices tradicionais são independentes dessa quantidade). Para sistemas de suporte a decisão, que necessitam de índices em atributos de grande seletividade, esses índices são os mais adequados [VAISMAN, 1998].

A maior vantagem dos índices *bitmap*, no entanto, é a alta performance alcançada no processo de consulta [VAISMAN, 1998]. Quando for necessário realizar uma operação com AND, OR ou NOT, o sistema apenas fará uma comparação de bits.

No exemplo, a seleção busca sexo = "M" e educação = "Universitário", supondo dois vetores de bits para cada valor como:

`0110010110010110` para sexo = “M” e
`1001010110110101` para educação = “Universitário”

É simples para um programa em qualquer linguagem, com um custo de processamento muito baixo, gerar um vetor que armazene o resultado da operação AND, comparando bit a bit:

`0000010110010100` = sexo = “M” e educação = “Universitário”

4.1.2 Índices com junção

Uma junção é conhecida por ser a operação com maior custo de performance em um SGBD. Existem algumas estratégias de processamento de junções em modelos relacionais, como *hash-join*, *sort-merge join*, entre outras. Para estratégias focadas para o suporte a decisão, as principais estratégias são o uso de índices com junção e *star join* [VAISMAN, 1998].

Essas estratégias tiram vantagem do esquema estrela do *data warehouse*, no qual a tabela de fatos está relacionada com as dimensões por chaves estrangeiras. As junções que serão requeridas pelas consultas irão usar essas chaves estrangeiras.

Considerando a seguinte consulta:

π sum(Vendas) (Vendas |X| (σ estado="SP" (Cliente)))

É possível criar um índice com uma junção na chave estrangeira, na tabela de Vendas, junto com o atributo "estado".

Isso parece estranho, porque o estado não pertence a "Vendas". Na verdade, é criado é um índice apontando para os registros de Vendas que armazenam vendas para clientes de SP. Isso é feito primeiramente com a junção das tabelas pela chave estrangeira, e então criando um *bitmap* em Vendas para cada estado. A figura 28 ilustra como ficaria um índice *bitmap* para o exemplo de Vendas.

Cada *bitmap* terá N bits, sendo N o número de linhas na tabela Vendas, e no caso de ter mais condições, pode-se criar um índice com junção para cada condição.

Vendas

cod_venda	**cod_produto**	**cod_cliente**	**qtde_vendida**
1	5	3	50
2	45	3	100
3	8	2	200
4	13	4	150
5	2	1	300
6	9	4	50

Cliente

cod_cliente	**estado**
1	SP
2	RJ
3	SP
4	PR

O índice ficaria como no exemplo:

```
SP 110010
RJ 001000
PR 000101
```

Figura 28: Exemplo de índices com junção no *data warehouse* de Vendas.

Esse conceito pode ser generalizado, não sendo restrito a apenas um relacionamento, e também é útil quando as condições são colocadas em diferentes tabelas de dimensões. A vantagem desse tipo de índice é sua flexibilidade, facilitando a alteração ou adição de condições.

Alguns sistemas permitem o uso de índices para junções com mais de duas tabelas. A idéia é a mesma do índice explicado anteriormente, ou seja, filtrar valores da tabela de fatos. Mas, neste caso, um índice multi-colunas é criado.

Como exemplo, pode-se utilizar a tabela de fatos Vendas e as tabelas de dimensões Tempo e Cliente, e a necessidade de buscar a quantidade de vendas dos clientes de São Paulo, no mês de dezembro de 2003, com essa consulta representada abaixo:

π sum(Vendas) (Vendas |X| (σ estado="SP" (Cliente)) |X|
(σ mes_calendario=12 $\wedge$ ano_calendário=2003(Data)))

Para criar um índice, é preciso fazer a junção das três tabelas e criar o índice na forma (estado, mes_calendario), com um registro no índice para cada par de valores da tabela Vendas.

Os sistemas que implementam esse tipo de estratégia de indexação precisam saber da existência desses índices ao processas as consultas, para tirar melhor proveito.

4.1.3 Junções estrela (*Star Join*)

Star Join é uma técnica que aumenta a performance de uma junção sem criar índices com junção, considerando-se que há um índice em cada um dos atributos envolvidos nas restrições.

Para a consulta de vendas de dezembro de 2003, para clientes de “SP”, apresentada anteriormente, o sistema iria primeiro selecionar todos os códigos dos clientes em "SP", armazenando-os em uma tabela temporária (T1). Então seriam selecionados todos os dias do mês de dezembro de 2003 e armazenados em outra tabela temporária (T2), e finalmente, seria feito o produto cartesiano T1 X T2 para fazer a junção com o índice (cod_cliente, cod_data) já existente na tabela Vendas.

Um problema dessa técnica acontece quando a tabela de fatos é esparsa (o que é bastante comum em grandes organizações). Neste caso, os recursos são divididos para calcular o produto cartesiano, para formar linhas que não participarão da junção. Cabe utilizar a técnica mais adequada em cada situação.

4.2 Operador CUBE na Agregação Relacional

Aplicações de análise de dados, geralmente, fazem agregação dos dados através de muitas dimensões. Essas aplicações sumarizam valores dos dados, extraem informações estatísticas e então contrastam uma categoria com outra.

A agregação dos dados é realizada através de funções de agregação SQL e o operador GROUP BY. Através destes operadores são produzidas respostas de zero-dimensão ou uma-dimensão. Para serem produzidas respostas com n-dimensões é utilizado o chamado operador CUBE.

4.2.1 Agregação no SQL

O SQL padrão conta com cinco funções de agregação de valores: **COUNT ( )** usado para contar a quantidade de registros em uma tabela de acordo com o critério, o **SUM ( )** que soma os valores em uma tabela de acordo com critérios, o **MIN ( )** que seleciona o valor mínimo em uma tabela, o **MAX ( )** que seleciona o valor máximo em uma tabela e o **AVG ( )** que obtém o valor médio em uma seleção. Além dessas, o SQL também é capaz de agregar valores que se repetem em uma seleção, usando o comando **DISTINCT.**

Por exemplo, poderíamos considerar o código abaixo aplicado sobre a tabela EMPREGADOS (figura 29):

```
SELECT AVG(salario) AS media_sal
  FROM Empregados;
```

Nome	depto	salario
Paulo Mendes	**121**	**3.000,00**
Antonio Gusmão	**121**	**5.000,00**
Patrícia da Silva	**010**	**1.500,00**
Marisa Santana	**005**	**8.000,00**

Figura 29: Tabela Empregados.

E como resultado obteríamos o seguinte (figura 30):

Media_sal
4.375,00

Figura 30: Média dos salários.

Da mesma forma, os outros operadores podem ser aplicados obtendo-se os resultados de acordo com cada operador utilizado. A exceção é o comando **DISTINCT** usado na sintaxe da seguinte maneira:

```
SELECT COUNT(DISTINCT Depto) AS qtd_dep
  FROM  Empresa;
```

Obtendo-se o seguinte resultado (figura 31):

Qtd_dep
3

Figura 31: Quantidade de departamentos.

Essas funções são extremamente úteis, mas quando tratamos de consultas analíticas de dados, elas não são suficientes. Existem alguns tipos de consulta muito complexas de serem implementadas com o SQL tradicional e são desses problemas que o próximo tópico trata com mais detalhes.

4.2.2 Problemas com o "group by"

Como vimos, o SQL possui algumas funções de agregação que, embora atendam muitas demandas, não são suficientes para atender a complexidade de um processo de consulta analítica.

Certas agregações analíticas são extremamente complexas, embora não sejam impossíveis de serem construídas com o SQL tradicional. Dentre os problemas mais comuns encontrados nestes casos estão pertinentes ao uso de *roll-up/drill-down* [GRAY, 1995].

Roll-up e *drill-down* usam totais e sub-totais para relatórios. Relatórios comumente agregam dados em um nível menos granular e então, sucessivamente, diminuem essa granularidade até obterem um resultado mais detalhado sobre o dado. O relatório de vendas de carro na figura 32 pode dar a idéia deste problema.

A tabela representada na figura 32 não pode ser relacional, uma vez que possui valores nulos na chave primária. A figura 33 é uma tabela relacional e mostra uma representação mais conveniente que a do exemplo anterior.

Modelo	Ano	Cor	Vendas por Modelo, Ano e Cor	Vendas por Modelo e Ano	Vendas por Modelo
GM	1994	Preto	50		
		Branco	40		
				90	
	1995	Preto	85		
		Branco	115		
				200	
					290

Figura 32: Tabela de *Roll Up* de Vendas de carro por Modelo, por Ano e por Cor.

Modelo	Ano	Cor	Unidades
GM	1994	Preto	50
GM	1994	Branco	40
GM	1994	ALL	90
GM	1995	Preto	85
GM	1995	Branco	115
GM	1995	ALL	200
GM	ALL	ALL	290

Figura 33: Tabela de Vendas.

Para se construir essa tabela a solução proposta foi incluir um valor chamado *ALL* para denotar a agregação dos campos. Desta forma, um *Roll-up* pode ser calculado com este código:

```
  SELECT  Modelo, 'ALL', 'ALL', SUM(Unidades)
    FROM  Vendas
   WHERE  Modelo = 'GM'
GROUP BY  Modelo
```

```
UNION

  SELECT  Modelo, Ano, 'ALL', SUM(Unidades)
    FROM  Vendas
   WHERE  Modelo = 'GM'
GROUP BY  Modelo, Ano

UNION

  SELECT  Modelo, Ano, Cor, SUM,(Unidades)
    FROM  Vendas
   WHERE  Modelo = 'GM'
GROUP BY  Modelo, Ano, Cor;
```

A representação da tabela de vendas (figura 33), com a união dos GROUP BY's, "resolve" o problema da representação dos dados agregados em um modelo de dados relacional.

Este exemplo é um *roll-up* de 3 dimensões. Agregações de n-dimensões requerem também n-uniões (*Union*), ou seja, quanto mais dimensões a serem agregadas, mais complexo será o código SQL para acessar os dados.

Note que a tabela de vendas (figura 33) não contempla a agregação das linhas que representam o total de vendas por ano. As linhas que não foram incluídas são as seguintes (figura 34):

Modelo	Ano	Cor	Unidades
GM	ALL	Preto	135
GM	ALL	Branco	155

Figura 34: Linhas não incluídas na agregação da tabela de Vendas.

Para que as mesmas sejam incluídas na consulta, é necessário inserir ao final do código utilizado na obtenção da tabela de vendas da figura 34 as seguintes linhas:

```
UNION

  SELECT  Modelo, 'ALL', Cor, SUM (Vendas)
    FROM  Vendas
   WHERE  Modelo = 'GM'
GROUP BY  Modelo, Cor;
```

Cross tabulation é a agregação de duas ou mais dimensões. Por exemplo, na figura 35 podemos observar uma *cross-tab* de duas dimensões.

GM	1994	1995	Total (ALL)
Preto	50	85	135
Branco	40	115	155
Total (ALL)	90	200	290

Figura 35: *Cross tabulation* de vendas da GM.

A representação de uma *cross tab* é equivalente a uma representação relacional, usando o valor *ALL*, generaliza-se uma *cross tab* de n-dimensões.

4.2.3 Operador CUBE

Podemos dizer que a dimensão 0 de um cubo de dados é um ponto, a dimensão 1 é uma linha com um ponto, a dimensão 2 é uma *cross tabulation,* um plano, duas linhas e um ponto enquanto a dimensão 3 é um cubo com três intersecções de *cross tabulations* de duas dimensões. A Figura 36 apresenta, dentro do exemplo que estamos utilizando, um cubo de dados [GRAY, 1995].

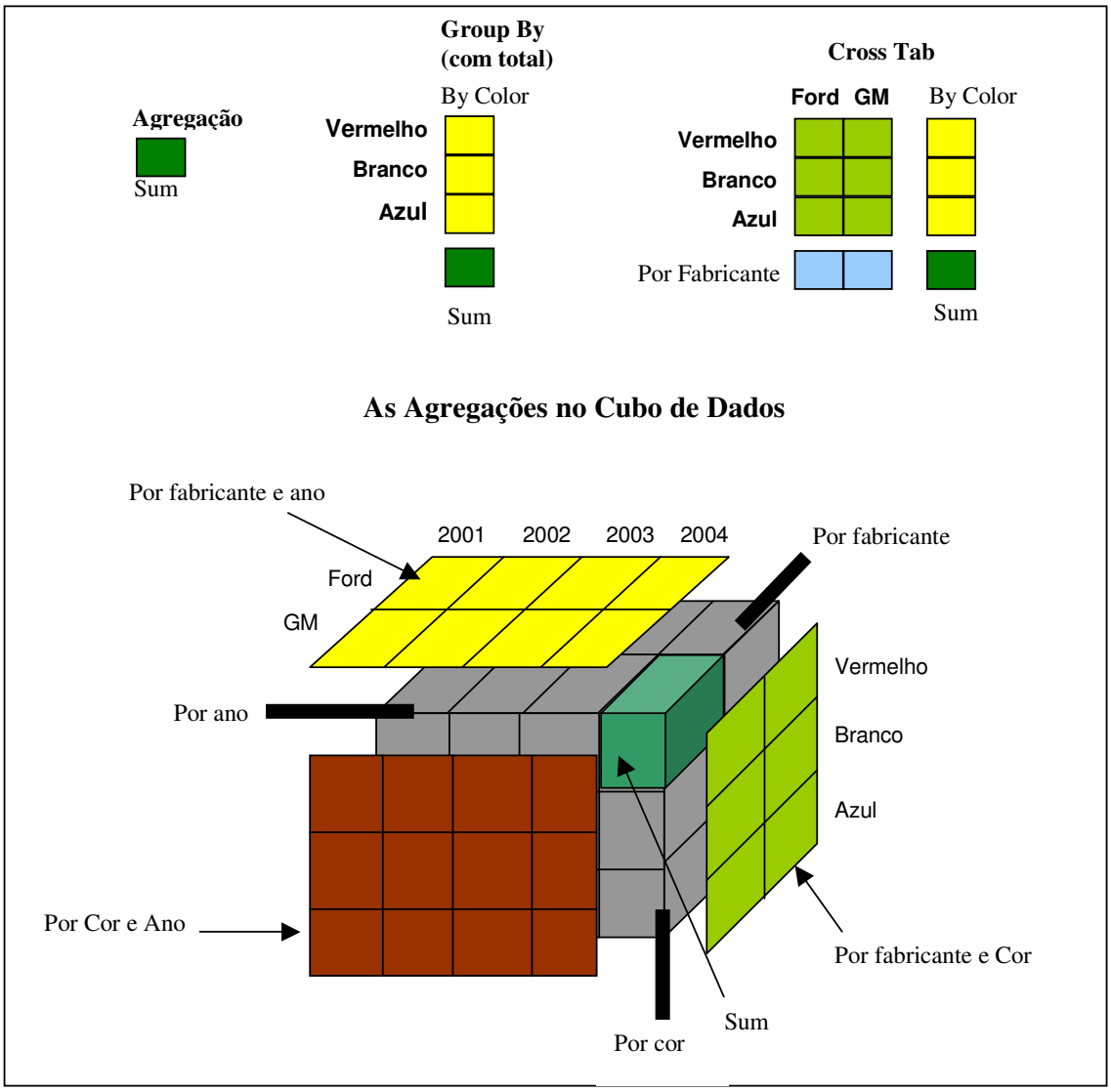


Figura 36: Agregações em um cubo de N Dimensões.

O operador *Data Cube* ou cubo de dados é a generalização em N-Dimensões de funções simples de agregação. Esse operador constrói uma tabela contendo todos os valores agregados e o total agregado é representado como uma tupla:

```
ALL, ALL, ALL, ..., ALL, f(*)
```

O GROUP BY tradicional pode gerar um cubo de dados de n-dimensões. Agregações com poucas dimensões aparecem como pontos, linhas, planos, cubos ou hipercubos. Pontos em planos com mais dimensões, possuem menos valores ALL.

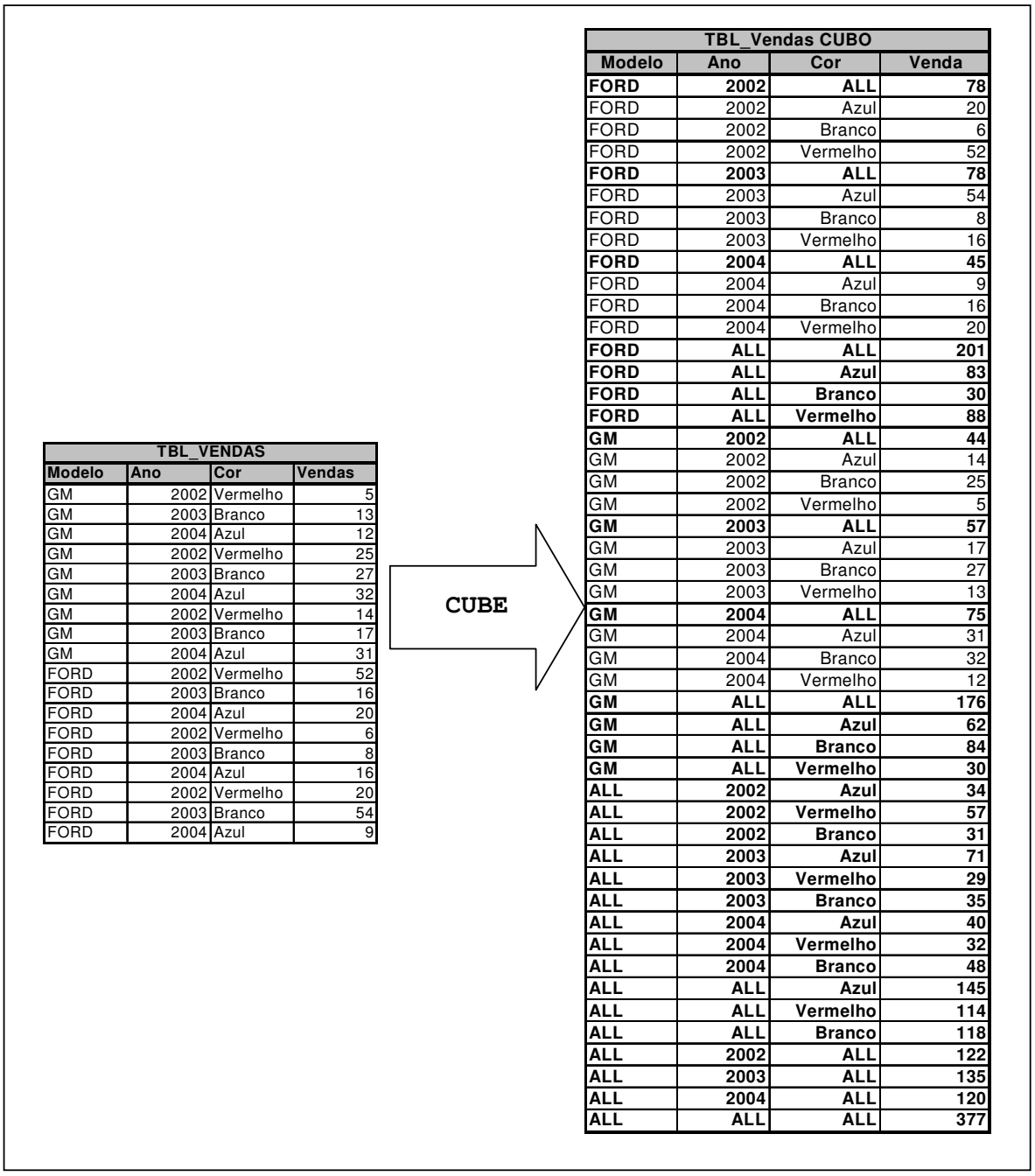
A sintaxe SQL usando o operador CUBE é a seguinte:

```
GROUP BY (
 {   ( < nome-da-coluna>|<expressão>)
    [ AS <nome correlato> ]
[<cláusula>]
  , ...)
[WITH  (CUBE |ROLLUP)]
  }
```

A figura 37 representa o uso da sintaxe SQL abaixo:

```
SELECT   Modelo, Ano, Cor, SUM(Vendas) AS Vendas
   FROM  TBL_VENDAS
   WHERE Modelo IN ('Ford' , 'GM')
      AND     Ano BETWEEN 2002 AND 2004
   GROUP BY   Modelo, Ano, Cor
   WITH  CUBE;
```

Podemos considerar que a cardinalidade de N atributos são C1, C2..., CN, então, a cardinalidade do cubo resultante, pode ser expressa pela projeção de (C1+1). O Valor extra em cada domínio é *ALL*. Por exemplo, a tabela TBL_VENDAS possui 2x3x3=18, considerando 2 possíveis valores para o atributo Modelo ('FORD' e 'GM'), 3 para Ano ('2002', '2003' e '2004') enquanto a tabela resultante após a aplicação do operador CUBE, possui 3x4x4=48, incluindo o valor *ALL*.



TBL_VENDAS			
Modelo	Ano	Cor	Vendas
GM	2002	Vermelho	5
GM	2003	Branco	13
GM	2004	Azul	12
GM	2002	Vermelho	25
GM	2003	Branco	27
GM	2004	Azul	32
GM	2002	Vermelho	14
GM	2003	Branco	17
GM	2004	Azul	31
FORD	2002	Vermelho	52
FORD	2003	Branco	16
FORD	2004	Azul	20
FORD	2002	Vermelho	6
FORD	2003	Branco	8
FORD	2004	Azul	16
FORD	2002	Vermelho	20
FORD	2003	Branco	54
FORD	2004	Azul	9

TBL_Vendas CUBO			
Modelo	Ano	Cor	Venda
FORD	**2002**	**ALL**	**78**
FORD	2002	Azul	20
FORD	2002	Branco	6
FORD	2002	Vermelho	52
FORD	**2003**	**ALL**	**78**
FORD	2003	Azul	54
FORD	2003	Branco	8
FORD	2003	Vermelho	16
FORD	**2004**	**ALL**	**45**
FORD	2004	Azul	9
FORD	2004	Branco	16
FORD	2004	Vermelho	20
FORD	**ALL**	**ALL**	**201**
FORD	**ALL**	**Azul**	**83**
FORD	**ALL**	**Branco**	**30**
FORD	**ALL**	**Vermelho**	**88**
GM	**2002**	**ALL**	**44**
GM	2002	Azul	14
GM	2002	Branco	25
GM	2002	Vermelho	5
GM	**2003**	**ALL**	**57**
GM	2003	Azul	17
GM	2003	Branco	27
GM	2003	Vermelho	13
GM	**2004**	**ALL**	**75**
GM	2004	Azul	31
GM	2004	Branco	32
GM	2004	Vermelho	12
GM	**ALL**	**ALL**	**176**
GM	**ALL**	**Azul**	**62**
GM	**ALL**	**Branco**	**84**
GM	**ALL**	**Vermelho**	**30**
ALL	**2002**	**Azul**	**34**
ALL	**2002**	**Vermelho**	**57**
ALL	**2002**	**Branco**	**31**
ALL	**2003**	**Azul**	**71**
ALL	**2003**	**Vermelho**	**29**
ALL	**2003**	**Branco**	**35**
ALL	**2004**	**Azul**	**40**
ALL	**2004**	**Vermelho**	**32**
ALL	**2004**	**Branco**	**48**
ALL	**ALL**	**Azul**	**145**
ALL	**ALL**	**Vermelho**	**114**
ALL	**ALL**	**Branco**	**118**
ALL	**2002**	**ALL**	**122**
ALL	**2003**	**ALL**	**135**
ALL	**2004**	**ALL**	**120**
ALL	**ALL**	**ALL**	**377**

Figura 37: Data Cube de 3 dimensões construído a partir da tabela tbl_vendas.

Cada valor **ALL** representa um conjunto de todos os valores que foram agregados.

Na tabela resultante os conjuntos respectivos são:

```
Modelo.ALL  = ALL (Modelo) = ('Ford','GM')
Ano.ALL    = ALL (Ano)    = ('2002','2003','2004')
Cor.ALL = ALL (Cor)    = ('Vermelho','Branco','Azul')
```

O valor *ALL* aparece como um valor essencial na agregação usando o comando CUBE. Sem ele, seria impossível obter super agregações como visto na figura 37 (totais por Modelo, Ano e Cor exibidos simultaneamente, assim como totais gerais), porém, ele cria uma complexidade substancial. Ele é um "não-valor" como o *NULL* e durante sua implementação alguns aspectos devem ser observados. Abaixo citamos alguns:

- O *ALL* torna-se uma palavra-chave denotando um conjunto de valores.

- A sintaxe *ALL* [NOT] ALLOWED deve ser adicionada à sintaxe de definição da coluna e aos atributos da coluna no catálogo do sistema, para permitir ou não a inclusão de valores *ALL*.

- *ALL*, como NULL, não participa de nenhuma agregação, exceto COUNT().

Embora existam outros aspectos, estes dão uma idéia das particularidades ao se utilizar o valor *ALL* [GRAY, 1995].

Embora o SQL padrão conte com diversas funções de agregação muito úteis no dia-a-dia de um usuário, estas não são suficientes para atender completamente a demanda de agregações exigidas por processos analíticos. Nesta demanda pode-se considerar a necessidade de se realizar *Roll-Up* e *Drill-Down* das informações. Para que o SQL consiga agrupar as informações para exibi-las de maneira eficiente para um processo que envolva *Roll-Up/Drill-Down*, complexas consultas precisam ser realizadas. O operador CUBE reduz a complexidade destas consultas, uma vez que, por si só, consegue extrair as informações no formato adequado com as devidas agregações.

4.3 Manutenção de Visões

Uma visão é uma relação derivada, definida a partir das relações bases, ou seja, relações que estão fisicamente armazenadas. Portanto, uma visão define uma função de um conjunto de tabelas base em uma tabela derivada, essa função é recomputada toda vez que uma visão é referenciada.

As visões materializadas são um tipo especial de visão, onde os registros são armazenados fisicamente dentro do banco de dados. Nos ambientes de *data warehouse* elas são utilizadas para pré-cálculos e armazenamento dos dados gerados, são usadas também para pré-calcular relacionamentos com ou sem agregações. A utilização dessas visões pode gerar uma melhoria significativa no tempo de execução de consultas.

Em contrapartida, quando um dado base for atualizado, as visões materializadas derivadas desse dado também devem ser atualizadas. O processo de atualizar uma visão materializada em resposta a modificações em um dado base é chamado de manutenção de visões. Esse processo nem sempre envolve recalcular a visão completamente, na maioria das vezes isso seria um desperdício de processamento, pois é possível computar nas visões apenas as modificações ocorridas nas relações base. Isto é chamado de manutenção incremental de visões.

Existem quatro dimensões através das quais o problema da manutenção de visões pode ser analisado [GUPTA, 1995]:

- Dimensão Informação: Refere-se à quantidade de informações que estão disponíveis para manutenção de visões. Essa dimensão trata de quais dados são

usados para computar uma alteração em uma visão, como quais relações base têm que ser acessadas, as restrições de integridade, chaves, etc.

- Dimensão Modificação: Refere-se a quais modificações um algoritmo de manutenção de visões pode tratar. Inclusão e exclusão de registros nas relações bases, alteração de registros, se são tratados diretamente ou são modelados como uma exclusão seguida de uma inclusão, as mudanças na definição da visão, conjuntos de alterações, etc.

- Dimensão Linguagem: Refere-se a fatores como se a visão pode ser expressa como uma consulta seleção-projeção-junção, ou em algum outro subconjunto de álgebra relacional, se utiliza SQL ou subconjunto de SQL, se pode ter duplicidades, usar agregação, recursividade, etc.

- Dimensão Instância: Refere-se à possibilidade dos algoritmos de manutenção de visões trabalharem ou não sobre todas as instâncias de um banco de dados e se eles trabalham ou não para todas as instâncias de modificação. Sendo assim uma instância pode ser de dois tipos: instância de banco de dados e instância de modificação.

A seguir são apresentados exemplos que ilustram as dimensões utilizadas para a classificação do problema da manutenção de visões.

Exemplo 1: Dimensões Informação e Modificação

Em uma empresa de desenvolvimento de sistemas, os módulos desenvolvidos podem ser genéricos, sendo, nesses casos, reaproveitados em diversos sistemas, tendo apenas

um custo de adaptação. Considerando a seguinte relação que trata o módulo, custo da adaptação e o sistema em que o módulo foi incorporado:

Sistema_modulo(modulo, custo, sistema)

E a seguinte visão que contém todos os módulos que custaram mais do que R$ 1000:

Modulos_caros(modulo) = π modulo σ custo>1000(Sistema_modulo)

Se for inserido um registro como (mod1, 500, CW), isso não gera nenhum impacto na visão, pois o custo é menor que R$1000. Porém, se for incluído o registro (mod2, 1250, WH) será necessário contemplar esse registro na visão, pois o custo é maior que R$1000. Diferentes algoritmos de manutenção de visões podem ser designados, dependendo das informações disponíveis para determinar se o mod2 deve ser inserido na visão:

- Se a visão materializada antiga estiver disponível, basta consultá-la para verificar se o módulo mod2 já está na visão, se estiver a visão não precisa ser modificada, senão o mod2 deverá ser inserido.

- Se a relação base Sistema_modulo estiver disponível, ela pode ser utilizada para verificar se existe algum registro com o mesmo módulo, mas com o custo igual ou maior. Se for encontrado algum registro, o novo registro inserido não precisa ser contemplado na visão.

- Se o módulo for chave na relação Sistema_modulo e esse módulo está sendo incluído agora, significa que ele não pode já estar na visão, portanto deve ser inserido.

Outro problema da manutenção de visões é responder às exclusões realizadas. Supondo que o registro (mod1, 6000, IF) seja excluído. Como o custo dele é maior que 1000, esse módulo está na visão, mas não se pode simplesmente excluir o mod1 da visão, é preciso saber se não existe outro registro na relação base que determine com que o mod1 continue na visão, como, por exemplo, o registro (mod1, 2000, DN). Portando, o algoritmo de manutenção não é capaz de resolver o problema da exclusão utilizando apenas a visão materializada, sempre serão necessárias outras informações como os próprios dados da relação base, restrições ou chaves.

Exemplo 2: Dimensões Linguagem e Instância

O exemplo 1 considerou uma linguagem de definição de visão consistida de operações de seleção e projeção, no exemplo 2, será ampliada a linguagem de definição da visão com a operação junção (*join*). A visão definida a seguir mostra os módulos distintos que fazem parte de sistemas em fase de homologação, definindo uma junção entre as relações Sistema_modulo(modulo, custo, sistema) e Sistema(sistema, status):

Mod_hom (modulo) = π modulo (σ status=”Hom”(Sistemas) |X| Sistemas_modulo)

No caso de inclusão de um registro (mod2, 250, “IF”), se já existir o módulo mod2 na visão Mod_hom, essa inclusão não afeta a visão, mas no caso de não existir, não é possível determinar o efeito dessa inclusão utilizando apenas a visão. Na visão Modulos_Caros do exemplo 1, era possível determinar a manutenção em resposta às inclusões utilizando apenas a visão, mas quando se utilizam junções isso se torna impossível.

A visão Mod_hom é mantida se ela já tiver o módulo mod2, mas não caso contrário. Assim, a manutenibilidade de uma visão depende também de instâncias particulares dos bancos de dados e das modificações.

4.3.1 Informações completas

Os casos em que os algoritmos utilizam informações completas, são aqueles que as relações bases e as visões materializadas estão disponíveis durante o processo de manutenção. Esses casos se aplicam a maioria dos trabalhos de manutenção de visões, e o foco tem sido buscar técnicas eficientes para manter visões expressadas em diferentes linguagens (iniciando de seleção-projeção-junção à álgebra relacional, e SQL), considerando funcionalidades como agregações, duplicidades, recursividade, e *outer-join*.

No que diz respeito aos casos de utilização de informações completas, são aplicadas a três tipos de visões: visões não-recursivas, visões *outer-join* e visões recursivas.

Visões não-recursivas

Utiliza o algoritmo de contagem, o qual pode ser definido usando união, negação e agregação. A idéia básica é contar o número de alternativas de derivações pra cada registro de uma visão e armazenar isso como uma informação extra na própria visão. Dessa forma, se for excluído um registro em uma relação base, é possível verificar se ele deve ou não ser excluído da visão.

Por exemplo, considerando a seguinte visão:

```
CREATE VIEW Visao1 as
  SELECT DISTINCT t1.Campo1, t2.Campo2
    FROM tab_base t1, tab_base t2
```

WHERE t1.Campo2 = t1.Campo1

E os valores abaixo para a relação tab_base:

Campo1	Campo2
a	b
b	c
b	e
a	d
d	c

Figura 38: Valores da relação tab_base.

A visão Visao1 ficará dessa forma ao ser incluído o número de derivações para cada linha, nomeado de “Conta”:

Campo1	Campo2	Conta
a	c	2
a	e	1

Figura 39: Valores da visão Visao1 com o número de derivações para cada linha

Supondo a exclusão do registro (a, b) da relação base, embora o registro (a, c) da visão seja derivada de (a, b), ele tem duas alternativas de derivações, como mostra o atributo Conta, ou seja, o registro (a ,b) não é o único que gera o (a, c) na visão, portanto, não deve haver a exclusão do registro na visão. A expressão abaixo aplica a diferenciação para definir a visão:

$$\Delta\text{-}(tab\_base) = (a,b)|X| \; tab\_base \; U \; tab\_base \; |X|(a,b) \; U \; (a,b)|X|(a,b) = \{(a,c), (a,e)\}$$

O algoritmo Contagem computa ∆-(tab_base) (∆+ para inserções). Cada registro em ∆ é associado com um valor de contagem (negativo para exclusões). Nesse exemplo, ficaria

(a, c, -1), (a, e, -1). Combinando isso com a visão materializada Visao1 atual, o resultado seria a seguinte visão:

Campo1	Campo2	Conta
a	c	1

Figura 40: Valores da visão materializada Visao1 após a exclusão do registro (a, b) da relação base.

O algoritmo de contagem executa esse processo para cada visão materializada presente no sistema.

Visões *outer-join*

Para definir uma visão *outer-join* é utilizada a seguinte sintaxe:

```
CREATE VIEW Visao1 AS
  SELECT x1, x2, ..., xn
    FROM tab1 FULL OUTER JOIN tab2 ON g(y1, y2, ...)
```

Onde g(y1, y2, ...) é uma conjunção de predicados.

Registros podem ser inseridos ou excluídos de tab1 ou tab2. Isso é representado por:

$\triangle+$ (tab1) e $\triangle-$(tab1)

O *outer-join* pode ser reescrito como um par de *outer joins* pela esquerda e pela direita:

```
s1 - SELECT x1, x2, ..., xn
       FROM △(tab1) LEFT OUTER JOIN tab2 ON g(y1, y2, ...)
s2 - SELECT x1, x2, ..., xn
       FROM tab1_novo RIGHT OUTER JOIN △(tab2) ON g(y1, y2, ...)
```

Obs.: tab1_novo é a relação tab1 depois das alterações.

Considerando tab1(A, B) e tab2(A, C), e as seguintes instâncias:

A	B
10	15
20	25

A	C
10	bb
30	dd

Figura 41: Instância das relações tab1 e tab2.

O *outer-join* completo é:

A	B	A	C
10	15	10	bb
20	25	null	Null
null	null	30	dd

Figura 42: Valores da visão com *outer-join* completo Visao1.

Inserindo (40, 45) em tab1, é inserido um registro (40, 45, null, null) na visão. Se for inserido um registro que já tenha um correspondente em tab2 como (30, 35), é obtida a combinação (30, 35, 30, dd). Porém, o algoritmo deve excluir o registro (null, null, 30, dd) da visão. Assim como se o registro (30, 35) for excluído de tab1, deve ser adicionado na visão (null, null, 30, dd).

Em resumo, s1 computa na visão Visao1 as mudanças de tab1, e s2 faz o mesmo com as mudanças ocorridas em tab2.

Visões Recursivas

O algoritmo utilizado na manutenção dessas visões é chamado de DRed – Exclusão (*Deletion*) e Rederivação (*Rederivation*). Como seu nome já revela, esse algoritmo é formado por duas fases. A primeira exclui todos os registros afetados pela expressão da visão. No exemplo do algoritmo de contagem, seriam excluídos ambos os registros, (a, c) e (a, e), não

importando quantas alternativas de derivação cada registro tem. Nesse caso, uma superestimação de registros derivados excluídos é obtida. Então, essa superestimação é aprimorada procurando as alternativas de derivação de cada registro excluído. Dessa forma, (a, c) será reinserido na visão. A inclusão é executada usando a visão (uma vez atualizada parcialmente) e as relações bases.

4.3.2 Informações parciais

Ao contrário da manutenção de visões que utiliza informações completas, uma visão nem sempre exige uma manutenção quando a modificação utiliza somente informações parciais. Assim, o algoritmo primeiro verifica se a visão deve ser atualizada, depois, caso isso seja possível, efetivamente faz a manutenção.

Consultas independentes de alteração

Um algoritmo verifica se uma alteração na relação base afeta a visão. Se a visão não é afetada pela alteração, nada é feito. Se ela é afetada, então os algoritmos de manutenção são aplicados.

Visões auto-atualizáveis

Uma visão é chamada de auto-atualizável, quando sua manutenção pode ser feita utilizando apenas a visão materializada e as restrições de chave. Essa é uma idéia importante quando as visões materializadas são aplicadas em *data warehouses*, porque assim, não é necessário varrer os dados base para atualizar as tabelas sumarizadas.

Pode-se dizer que uma visão é auto-atualizável com respeito à modificação tipo T em uma relação base R se a visão pode ser auto-atualizada para todas as instâncias de um banco de dados em resposta a todas as modificações de tipo T sobre a relação R.

No segundo exemplo apresentado no início dessa seção de manutenção de visão:

Mod_hom (modulo) = π modulo (σ status="Hom"(Sistemas) |X| Sistemas_modulo)

Se for excluído um registro em Sistemas_modulo, como (mod2, 250, "IF"), não se pode excluir o módulo mod2 da visão sem verificar se esse módulo foi implementado também em outro sistema da empresa. Sendo assim, a visão Mod_hom não é auto-atualizável com respeito a exclusões em Sistemas_modulo. Assim como essa visão não é auto-atualizável com respeito a inclusões em qualquer uma das duas relações bases.

A maioria das visões seleção-projeção-junção não é auto-atualizável com respeito a inclusões, mas muitas vezes são auto-atualizáveis com respeito a exclusões e alterações. Por exemplo:

- Uma visão seleção-projeção-junção que faz a junção de duas ou mais relações distintas não é auto-atualizável com respeito a inclusões.

- Uma visão seleção-projeção-junção é auto-atualizável com respeito a exclusões em tab1 se os atributos chaves de cada ocorrência de tab1 na junção também estão incluídos na visão, ou são igualadas a uma constante na definição da visão.

- Uma visão Visao1 com *outer-join* pela esquerda ou completo definida usando duas relações tab1 e tab2 é auto-atualizável com respeito a todos os tipos de

> modificações na relação tab2, desde que as chaves de tab1 e tab2 sejam distintas e todos os atributos expostos de tab2 sejam distintos. Um atributo é distinto em uma visão Visao1, se ele aparece na cláusula SELECT de definição da visão. Um atributo A pertencente a uma relação tab1 é exposto numa visão Visao1, se A for utilizado no predicado da definição.

Com as técnicas apresentadas neste capítulo, é possível melhorar o desempenho em tarefas que recuperam dados em um *data warehouse*. Seja através de índices como os usados em consultas, como na manutenção correta e otimizada de visões ou ainda com a agregação relacional, um tempo e um custo razoável de processamento podem ser economizados, propiciando maior eficiência na entrega de resultados ao usuário.

CONCLUSÃO

Uma modelagem de dados eficientemente construída é fundamental na construção de um *data warehouse* e, para permitir o acesso a informações em grandes bases de dados. Observou-se que, o modelo relacional apresenta um ótimo desempenho para qualquer tipo de operação transacional (inserções, alterações e exclusões), que é seu objetivo, mas não atende plenamente a demanda por informações, já que sua performance é baixa para executar consultas complexas, envolvendo períodos de tempo ou grandes quantidades de tabelas e registros.

Com o modelo multidimensional é possível obter-se um desempenho superior, no que se refere às consultas e análise de grandes volumes de dados. Esse modelo permite, ainda, que as consultas sejam realizadas de maneira mais intuitiva e flexível pelo usuário.

Esses resultados são obtidos, em geral, pela utilização de esquemas do tipo estrela e floco de neve. A composição formada por uma única tabela de fatos, onde se encontram as medidas dos valores analisados, e circundada por quantas tabelas dimensão que trazem as descrições textuais do negócio (esquema estrela), fornece uma estrutura desnormalizada e mais adequada a consultas a ser definida de acordo com a área de negócio que irá atender.

A modelagem deve permitir a obtenção do mais baixo nível de detalhe possível sobre os dados, armazenando-se não só dados sumarizados tais quais valores mensais ou quinzenais, mas também informações de cada operação de negócio. Essa capacidade de navegação para baixo (*drill-down*) ou para cima (*roll-up*), no detalhe das informações, se dá pela adoção de hierarquias nos esquemas multidimensionais, que definem os níveis "navegáveis" que o usuário poderá acessar.

Com essa estrutura, as consultas tendem a requerer maior tempo de processamento, uma vez que, totais precisam ser calculados sempre que um nível de detalhe mais alto é solicitado. Quando esse tipo de consulta se torna suficientemente freqüente, a criação de tabelas agregadas, que contém os dados pré-calculados para alguns níveis, pode ser uma alternativa de solução, apesar de seu custo de armazenamento.

Um problema a ser contornado no modelo dimensional é o gerenciamento de alterações de dados nas tabelas dimensão. Muitas informações, como o estado civil ou o endereço de uma loja alternam-se com o tempo e o *data warehouse* deve estar preparado para controlar essas alterações da maneira mais adequada possível. Em alguns casos, dados podem ser simplesmente sobrescritos pelos novos, mas em geral, uma alteração como essa poderia trazer problemas com as agregações ou consultas baseadas no valor antigo. Nesse caso, outras técnicas podem ser utilizadas como a adição de uma nova linha na tabela dimensão, alterando as novas informações, ou então a criação de uma nova coluna contendo o valor inicial de determinado atributo, ou ainda o uso de artefatos de dados.

São muitas as alternativas para construção de modelos multidimensionais e, um "bom" modelo parece estar condicionado ao problema específico que tratamos. Desse modo, a ausência de receitas facilmente aplicáveis e gerais torna a modelagem multidimensional uma atividade complexa, mas, da qual, em muito dependem os resultados dos *data warehouses* e dos sistemas de suporte a decisão.

GLOSSÁRIO

Ad-hoc	São consultas com acesso casual único e tratamento dos dados segundo parâmetros nunca antes utilizados, geralmente executado de forma iterativa e heurística. Isso tudo nada mais é do que o próprio usuário gerar consultas de acordo com suas necessidades de cruzar as informações de uma forma não vista e com métodos que o levem a descoberta daquilo que procura.
Agregações	Linhas físicas em um banco de dados, na maioria das vezes criadas pela soma de outros registros nos bancos de dados visando melhorar o desempenho de consulta.
Análise	Decomposição de campos operacionais, com o nome ou o endereço em partes elementares padrão.
Área de Apresentação de Dados	Local em que o *data warehouse* está organizado, armazenado e disponível para consulta direta dos usuários, ferramenta de acesso de dados e outras aplicações analíticas.
Atributo	Coluna (campo) em uma tabela de dimensão
Banco de dados multidimensional	Banco de dados em que os dados são apresentados em cubos de dados, em oposição a tabelas em uma plataforma de banco de dados relacional.
Byte	Unidade de medida formada por 8 bits de dados.
Chave composta	Em um banco de dados, uma chave formada por várias colunas.
Chave estrangeira (FK)	Coluna em uma tabela de banco de dados relacional cujos valores são extraídos de valores de uma chave primária localizada em uma outra tabela.
Chave primária (PK)	Coluna em uma tabela de banco de dados que é exclusivamente diferente de cada linha na tabela.
Classificar	Ordenar os dados de acordo com critérios projetados
Cláusula FROM (SQL)	Cláusula SQL que lista as tabelas necessárias para a consulta
Cláusula GROUP BY (SQL)	Cláusula SQL que lista exclusivamente os itens não agregados na lista SELECT, ou seja, tudo o que não seja SUM, COUNT, MIN, MAX, AVG.
Cláusula ORDER BY (SQL)	Cláusula SQL que determina a ordem das linhas do conjunto de respostas.
Coluna	Estrutura de dados que contém um item de dados específicos dentro de uma linha (registro). Equivalente a um campo do banco de dados.
Consulta *(Query)*	Solicitação feita pelo usuário de informações armazenadas em um *data warehouse*
Cubo	Nome de uma estrutura dimensional em uma plataforma de banco de dados de processamento analítico on-line (OLAP) ou multidimensional, originalmente referindo-se ao caso simples de três dimensões: produto, mercado e hora.
Dados atômicos	Os dados mais granulares detalhados capturados por um processo corporativo. Devem ser disponibilizados na área de apresentação dos dados para responder a consultas específicas e imprevisíveis.

Data Staging Area	Área de armazenamento e conjunto de processos que limpam, transforma, combinam, retiram duplicidades, armazenam e preparam dados de origem para serem usados no *data warehouse*
Data warehouse	Aglomeração das áreas de *staging* e de apresentação do *data warehouse* de uma empresa, em que dados operacionais estão estruturados especificamente para consulta e desempenho de análise e facilidade de utilização.
Desnormalização	Aceitação de redundância em uma tabela para que ela permaneça simples e não normalizada. Esse procedimento tem por objetivo melhorar o desempenho e facilitar ainda mais a utilização.
Dimensão	Entidade independente em um modelo dimensional que serve como ponto de entrada ou como mecanismo para aplicar o recurso de "separação e combinação" (slicing and dicing) das medidas aditivas localizadas na tabela de fatos do modelo dimensional.
Drill across **(interligação)**	Ato de solicitar dados com a mesma identificação de duas ou mais tabelas de fatos em um único relatório, quase sempre envolvendo consultas separadas que são intercaladas com uma segunda passagem através da correspondência de cabeçalhos de linhas.
Drill down **(descer na hierarquia)**	Ato de adicionar um cabeçalho de coluna ou substituir um cabeçalho de linha em um relatório para dividir as linhas pelo conjunto de respostas mais minuciosamente.
DSS	Veja *Sistemas de suporte à tomada de decisões*
Escalabilidade	Capacidade de acomodar exigências de crescimento futuras
Esparsa	Tabela de fatos que possui um número relativamente pequeno de todas as combinações possíveis de valores chaves.
Extração, Transformação e Carga (ETL)	Conjunto de processos através dos quais os dados operacionais de origem são preparados para o *data warehouse*
Ferramenta de acesso a dados	Uma ferramenta cliente que consulta, obtém ou manipula dados armazenados em um banco de dados relacional, de preferência um modelo dimensional localizado na área de apresentação de dados
Ferramenta de ETL	Aplicação de software normalmente residente no cliente e no servidor que ajuda nos processos de extração/transformação da carga de dados da produção.
Granularidade	Nível de detalhe capturado no *data warehouse*
Índice	Estrutura de dados associada a uma tabela que esta ordenada logicamente pelos valores de uma chave e é usada para melhorar o desempenho do banco de dados e a velocidade de acesso à consulta.
Metadados	Qualquer dado mantido para sustentar as operações ou a utilização de um *data warehouse*; funciona como uma enciclopédia para o *data warehouse*
Modelagem dimensional	Metodologia que permite modelar logicamente dados para melhorar o desempenho de consultas e prover facilidade de utilização a partir do conjunto de eventos básicos de medição.
MOLAP (OLAP Multidimensional)	Implementações de processamento analítico on-line dedicadas que não dependem de bancos de dados relacionais.
Normalização	Técnica de modelagem lógica que remove redundância de dados separando os dados em muitas entidades distintas, cada uma das quais se torna uma tabela e um SGBD relacional.

OLAP (Processamento analítico on-line)	É um conjunto de princípios vagamente definidos que fornecem uma estrutura dimensional de suporte à tomada de decisões.
OLTP (Processamento de transações on-line)	Descrição original de todas as atividades e sistemas associados à entrada segura de dados em um banco de dados.
Processamento analítico	Utilização de dados para fins de análise visando suportar a tomada de decisões na empresa.
SGBD(Sistema Gerenciador de Banco de Dados)	É uma aplicação de computador para armazenar, recuperar e modificar dados de uma forma altamente estruturada
Sistema de origem	Sistema operacional de registro cuja função é capturar as transações ou outras medidas de desempenho de processos de uma empresa.
Sistema de suporte à tomada de decisões (*DSS, Decision Support System*)	Nome original do processo de *data warehouse*.
Sistema operacional de registro	Sistema operacional para captura de dados sobre as operações e processos de negócio de uma empresa.
Slice and Dice	Capacidade de acessar igualmente um *data warehouse* através de qualquer uma de suas dimensões. É o processo de separação e combinação de dados do warehouse em combinações aparentemente infinitas.
***Snowflake* (Floco de neve)**	Dimensão normalizada em que uma dimensão de tabela simples é decomposta em uma estrutura de árvore com, potencialmente, muitos níveis de alinhamento
SQL (*Structured Query Language*)	Linguagem padrão para acesso a bancos de dados relacionais
Tabela	Coleção de linhas (registros) que possuem colunas associadas (campos).
Tabela de dimensão	Tabela em um modelo dimensional com uma chave primária composta por uma única parte e colunas de atributos descritivos.
Tabela de fatos sem fatos	Tabela de fatos que não possui fatos, mas captura determinados relacionamentos de muitos para muitos entre as chaves de dimensão. É mais normalmente usada para representar eventos ou fornecer informações que não aparecem nas outras tabelas de fatos.
Visão (SQL)	Instrução SQL que cria cópias lógicas de uma maneira que uma consulta completa possa ser usada separadamente em uma instrução SELECT.

REFERÊNCIAS BIBLIOGRÁFICAS

COLOSSI, N.; MALLOY, W.; REINWALD B. **Relational Extensions for OLAP.** IBM Systems Journal, vol 41, nº 4, 2002.

FERNANDES, Lúcia. **Oracle 9i Para Desenvolvedores Curso Completo**. Rio de Janeiro. Axcel Books do Brasil, 2002. 1614 p.

GRAY, Jim; BOSWORTH, Adam; LAYMAN, Andrew; PIRAHESH, Hamid. **Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Technical Report MSR-TR-95-22**. Redmond, WA.

GUPTA, Ashish, MUMICK, Inderpal Singh. **Maintenance of Materialized Views: Problems, Techniques, and Applications**. San Jose, 1995.

INMON, W.H. **Como construir o Data Warehouse.** Rio de Janeiro: Editora Campus, 1997. 388p.

INMON, W.H, WELCH, J. D., GLASSEY, K. L. **Gerenciando Data Warehouse.** São Paulo: Makron Books, 1999. 375p.

KIMBALL, Ralph. **Data Warehouse Toolkit.** São Paulo: Makron Books, 1996-b. 388 p.

KIMBALL, Ralph, REEVES, Laura, ROSS, Margy, THORNTHWAITE, Warren. **The Data Warehouse Lifecycle Toolkit – Expert Methods for Designing, Developing and Deploying Data Warehouses**. New York: John Wiley & Sons, Inc., 1998. 771 p.

KIMBALL, Ralph, ROSS, Margy. The **Data Warehouse Toolkit (Segunda Edição).** Rio de Janeiro: Editora Campus, 2002. 494 p.

MACHADO, F. N. R.; ABREU, M. P. **Projeto de Banco de Dados: uma visão prática.** São Paulo: Érica, 1996.

MEYER, Don, CANNON, Casey. **Building a Better Data Warehouse.** New Jersey: Prentice-Hall PTR, 1998. 227 p.

POE, Vidette, KLAUER, Patricia, BROBST, Stephen. **Building a Data Warehouse for Decision Support.** New Jersey. Prentice-Hall, Inc, 1998. 285 p.

SOARES, Vânia Jesus de Araujo. **Modelagem Incremental no Ambiente de Data Warehouse (tese de mestrado).** Rio de Janeiro, 1998.

VAISMAN, Alejandro A. **Olap, Data Warehousing, and Materialized Views: a Survey**. Buenos Aires, 1998.

BIBLIOGRAFIA COMPLEMENTAR

COUGO, Paulo. **Modelagem Conceitual e Projeto de Banco de Dados**. Rio de Janeiro: Campos, 1997.

FIGUEIREDO, Adriana Maria C.M. **MOLAP x ROLAP: Embate de Tecnologias para Data warehouse**. Developer's Magazine. Rio de Janeiro: Axcel Books do Brasil Editora Ltda. Fevereiro de 1998, nº 18 p.24.

PEDERSEN, T. B., JENSEN, C. S. **Multimensional Database Technology.** Revista Computer, 2001.

PENDSE, N. **What Is OLAP?** *The OLAP Report*, see http://www.olapreport.com/fasmi.htm.

THOMSEN, E. **OLAP Solutions**. John Wiley&Sons, Inc., Hoboken, NJ (1997).

APÊNDICE A - OLAP (Processamento Analítico On-line)

A análise multidimensional para OLAP surgiu após a publicação de *A Programming Language (Ken Iverson,1962).* Com base nessas idéias a IBM desenvolveu e implementou a primeira linguagem com análise multidimensional, no final da década de 60, chamada de APL. A evolução desta linguagem culminou na criação das ferramentas OLAP na década 90.

A aplicação OLAP soluciona o problema de síntese, análise e consolidação de dados, pois é o processamento analítico *online* dos dados. Tem capacidade de visualização das informações a partir de muitas perspectivas diferentes, enquanto mantém uma estrutura de dados adequada e eficiente. A visualização é realizada em dados agregados, e não em dados operacionais porque a aplicação OLAP tem por finalidade apoiar os usuários finais a tomar decisões estratégicas.

À medida que os usuários do *data warehouse* ampliam seu conhecimento das capacidades do processamento DSS (Decision Support System) e que cresce o volume de dados exponencialmente, a necessidade de técnicas mais sofisticadas para facilitar o uso do ambiente do *data warehouse* também aumenta.

Para suprir esta demanda por um ambiente OLAP completo e eficiente, um conjunto de regras baseadas na arquitetura foram definidas por E.F.Codd, servindo de guia para fornecedores de ferramentas de análise e outros produtos.

Visão conceitual multidimensional	Todos os atributos necessários para dar suporte ao processamento DSS devem ser moldados e os dados capturados para que seja possível haver qualquer visão conceitual de dimensões.
Transparência	O acesso a qualquer nível de d*ata warehouse* dever ser transparente ao usuário.
Acessibilidade	O *data warehouse* deve permitir acessibilidade aos dados possibilitando transformá-los em informação
Performance de relatório consistente	O tempo de resposta na geração dos relatórios deve ser constante , independente de ambiente ou outras circunstâncias.
Arquitetura Cliente-Servidor	O *data warehouse* é uma arquitetura e não uma tecnologia, portanto é independente de plataforma.
Dimensionalidade genérica	Qualquer atributo que seja necessário a um usuário-alvo deve estar disponível aquele usuário como uma dimensão. Assim, todas as dimensões e os atributos ou grupos de atributos que elas representam são "genéricos".
Operação dimensional cruzada irrestrita	Os níveis departamentais podem ser construídos para dar suporte a qualquer operação dimensional cruzada. (*drill-across*)
Manipulação de dados intuitiva	O uso de metadados é muito importante pois eles fornecem definições em um contexto de negócio para os usuários, fazendo com que todos eles tenham a mesma compreensão dos dados.
Flexibilidade quanto a relatórios	O *data warehouse* deve permitir ao usuário a maior flexibilidade possível na construção de relatórios.
Dimensão e níveis de agregação ilimitados	Depende de como o nível atômico do *data warehouse* é projetado.
Pesquisa de detalhes (Nível de Linha)	O *data warehouse* arquitetado irá suportar capacidades de pesquisas (drill-down) tanto no nível departamental quanto individual.
Atualização incremental de banco de dados	Atualmente os sistemas de gerenciamento de bancos de dados multidimensionais não suportam atualização incremental de dados, por isso o banco de dados inteiro deve ser recarregado com os dados novos.
Arrays múltiplos	Um *data warehouse* pode e deve suportar um número adequado e os tipos de matrizes a fim de satisfazer as necessidades de informação dos diferentes usuários alvo.
Seleção de subconjuntos	É a seleção de subconjuntos a partir do nível atômico do *data warehouse*, usado para popular os níveis departamentais e individuais.
Suporte a dados locais	A arquitetura do *data warehouse* suporta a localização física dos dados tão perto do usuário final quanto apropriado. A implementação descentralizada da arquitetura do *data warehouse*, especialmente nos níveis departamental e individual, é um de seus recursos mais poderosos.

Quadro 1: Regras OLAP
Fonte: Inmon W. H. (1999)

1 Origem dos dados do OLAP

A origem dos dados do OLAP é o nível de dados estruturado organizacional de um DW que já foi previamente carregado por outros processos aleatórios advindos de outros sistemas da empresa.

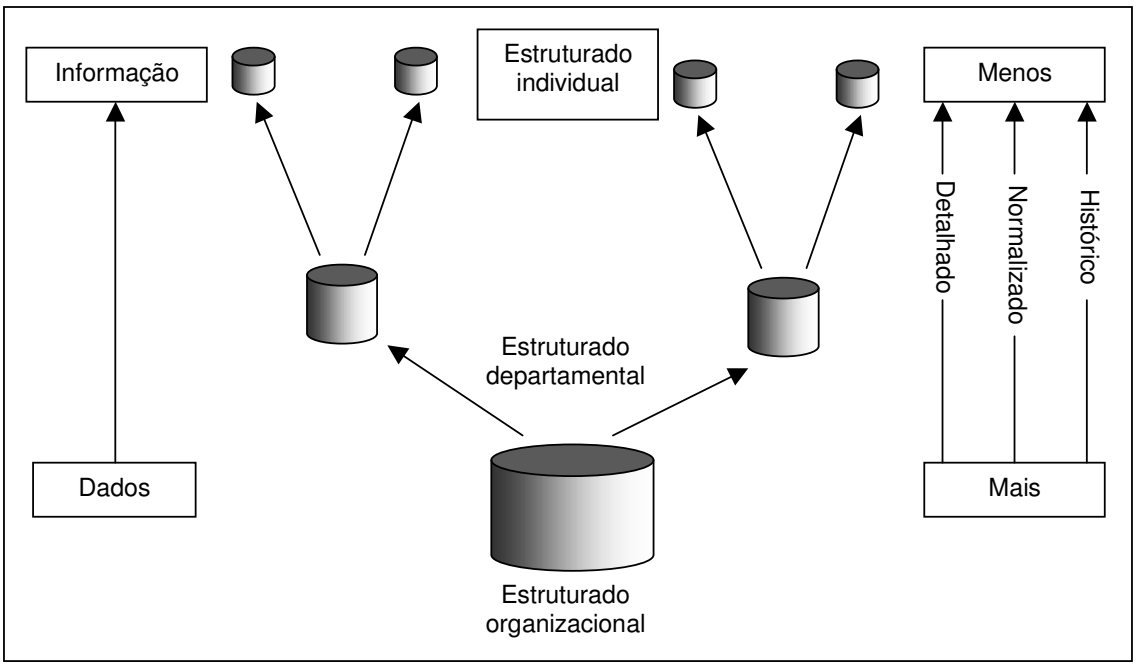


Figura 1: A arquitetura de Data Warehouse.

O OLAP possui um tamanho reduzido em comparação ao nível de dados estruturado organizacional (cerca de três vezes menor). Isso se dá devido ao fato do OLAP atender determinadas áreas de negócio, adequando seu conteúdo apenas às necessidades dos usuários que irão utilizá-lo, e/ou armazenando um período histórico menor dos dados. Ou seja, os dados são interpretados e agregados para atender determinadas áreas de negócios (departamentos) da empresa, segmentando e preparando os dados de maneira particular para cada necessidade, como mostra a **figura 1**. Tal medida é de extrema importância para garantir um tempo de resposta mais rápido nas consultas.

Podemos dizer então que, como o OLAP envolve um volume de dados reduzido em relação ao nível de dados estruturado organizacional, ele é mais flexível.

2 OLAP e a carga de trabalho

Uma das vantagens mais interessantes do OLAP é a redistribuição da carga de trabalho (processamento) gerados pelo acesso a dados realizado no ambiente estruturado organizacional para uma combinação de aquisição de dados do ambiente estruturado organizacional para o ambiente OLAP, e o acesso a dados realizado em ambos. A figura 2 mostra as diferenças antes e depois do ambiente OLAP ter sido criado [INMON, 1999].

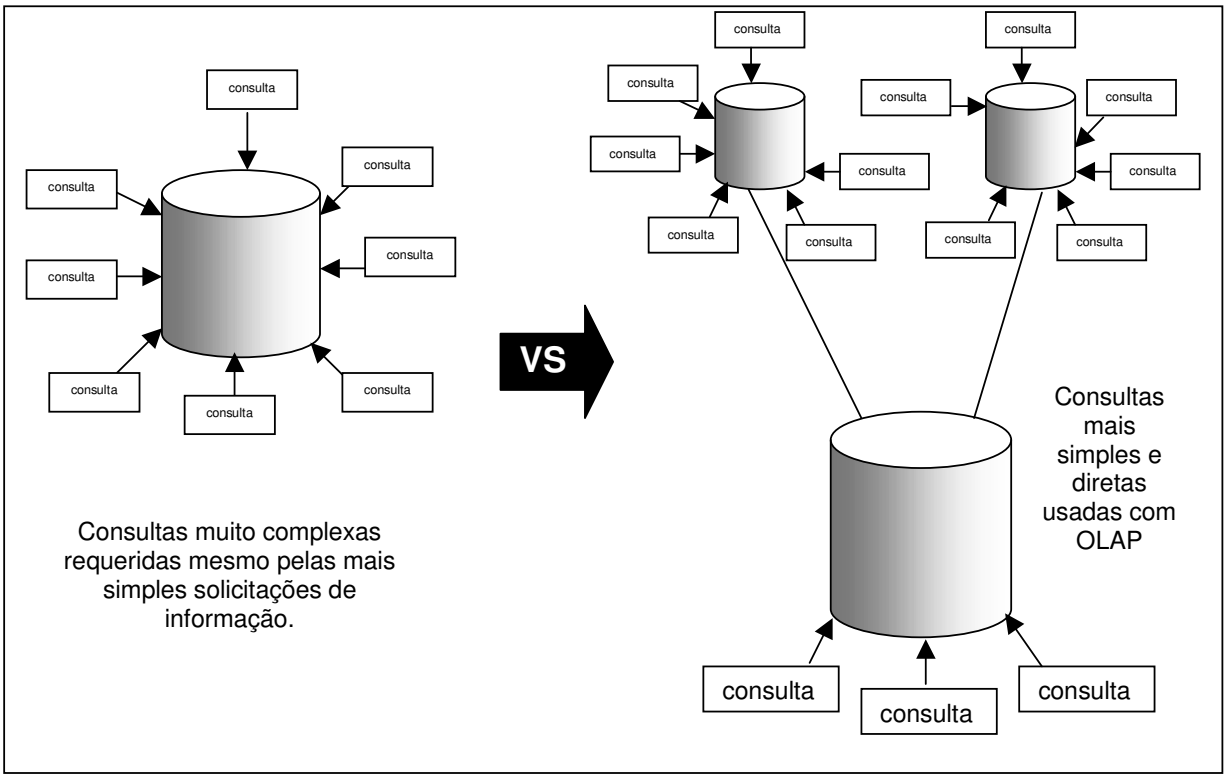


Figura 2: OLAP desvia profundamente a carga de trabalho

Quando não há ambiente OLAP, todas as consultas são executadas no ambiente estruturado organizacional. Executar todas as consultas no ambiente estruturado organizacional pode não ser o problema contanto que não haja muitos dados e que não haja muito processamento ocorrendo nele.

No instante em que houver muitos dados no ambiente estruturado organizacional ou processamento demais no ambiente, uma opção para facilitar o acesso a esses dados é a criação de um ambiente OLAP.

O uso deste ambiente provoca um desvio de processamento de consultas do nível estruturado organizacional do *data warehouse* para o próprio OLAP, proporcionando:

- Economia de processamento;
- Alta flexibilidade;
- Personalização de dados necessários por determinada área de negócio;
- Tirar vantagem de diferentes softwares para satisfazer diferentes necessidades, residindo em uma plataforma OLAP;
- Permitir que partes significativas de dados sejam isoladas;
- Permitir que subconjuntos de dados sejam isolados.

3 Necessidades do ambiente OLAP

Alguns dos fatores que aceleram a necessidade de uma ambiente OLAP para fornecer um acesso a dados mais eficiente na transformação de dados em informações incluem:

- Pré-agregação de dados para obter uma performance melhor.
- Pré-categorização de dados para uma melhor compreensão e usabilidade por parte dos usuários finais.
- Padronização de cálculos, métricas e outros dados derivados para assegurar a precisão por toda a organização.
- Um único ponto de acesso (dados estrutural organizacional) *versus* muitos (dados OLAP).

Apesar das vantagens oferecidas pelo ambiente OLAP, nem sempre é possível atender às necessidades dos usuários somente com este nível de detalhe. Algumas vezes, consultas requerem um nível de detalhe ou tipos de dados necessários tais, que somente no ambiente estruturado organizacional é possível obter os resultados esperados.

4 Metadados no OLAP

Devido às freqüentes alterações nas regras de negócio que servem de base para a aquisição dos dados no ambiente OLAP, torna-se imprescindível a utilização de metadados,

uma vez que, sem eles, seria impossível reconhecer e compreender as alterações feitas aos dados ao longo de diferentes períodos de tempo.

Os componentes dos metadados OLAP são muito similares aos encontrados no nível estruturado organizacional, dentre eles estão:

- Informação descritiva sobre o que está no ambiente OLAP (conteúdo, estrutura, definição etc.);
- A origem dos dados (dados estruturados organizacionais ou dados externos);
- O nome comercial e técnico dos dados;
- Uma descrição do resumo, subconjunto, superconjunto e/ou os processos de desnormalização que descrevem a jornada dos dados do nível estruturado organizacional ao ambiente OLAP;
- Métricas que descrevem quantos dados e de que tipos encontram-se em um ambiente OLAP;
- Informação da programação de atualização, descrevendo quando os dados OLAP foram inseridos;
- Informação sobre modelagem, descrevendo como os dados no ambiente OLAP se relacionam ao modelo de dados corporativo e ao modelo de dados de OLAP.

Há a necessidade de metadados tanto globais quanto locais no ambiente OLAP. Os metadados locais são utilizados para uma determinada área de negócio que cada ambiente

OLAP atende, já os metadados globais servem para reter informações que são aplicadas a diversos ambientes OLAP, mantendo assim uma integração entre eles.

5 Arquiteturas OLAP

A arquitetura em que a aplicação OLAP trabalha deve ser elaborada de acordo com o método de armazenamento de dados. Os principais métodos de armazenamentos são MOLAP e ROLAP. Cada um deles deve ser utilizado quando suas particularidades melhor atenderem às necessidades da área de negócio que fará uso do OLAP.

5.1 MOLAP (Multidimensional OLAP)

MOLAP (*Multidimensional On Line Analytical Processing*) é uma classe de sistemas que permite a execução de análises bastante sofisticadas usando como gerenciador de banco de dados, um banco multidimensional [FIGUEIREDO, 1998].

A tecnologia MOLAP não é somente uma ferramenta de visualização, e sim, uma tecnologia complexa de armazenamento e visualização de dados onde devemos nos preocupar com todo o processo de engenharia de banco de dados, tamanho, estruturação, processo de carga, indexação, otimização, etc.

No MOLAP os dados são armazenados em matrizes proprietárias concebidas especificamente para análises dimensionais e indexados de maneira a prover um ótimo desempenho no acesso a qualquer elemento. Os índices são necessários apenas para o gerenciamento de dados esparsos e cabem na memória principal.

Usando indexação, antecipação da maneira como os dados são acessados e alto grau de agregação dos dados no processamento das agregações, o MOLAP é capaz de apresentar uma melhor performance em relação ao ROLAP no tempo de resposta ao usuário, uma vez que no processo de carga dos dados ele pode pré-calcular diversas informações.

Servidores MOLAP aplicam estratégias de compressão para manipular cubos esparsos (cubos em que a maioria das células não tem valor), fazendo um uso eficiente do *'caching'*. Por outro lado, ele consome muito espaço. Como conseqüência, o MOLAP é geralmente usado em *Data Marts* (pequenos DW envolvendo somente um único setor de uma organização).

5.2 ROLAP (Relacional OLAP)

Sistemas ROLAP fornecem análise multidimensional de dados armazenados em uma base de dados relacional. Atualmente existem duas maneiras de se fazer este trabalho:

- fazer todo o processamento dos dados no servidor da base de dados, neste caso o servidor OLAP gera os comandos SQL em múltiplos passos e as tabelas temporárias necessárias para o devido processamento das consultas;

- executar comandos SQL para recuperar os dados, mas fazer todo o processamento (incluindo junções e agregações) no servidor OLAP.

No ROLAP a sumarização é executada por uma ferramenta externa, os dados são mapeados nos registros nas tabelas do modelo estrela, agregações são pré-calculadas nas tabelas sumário e metadados são armazenados em tabelas relacionais (índices).

No final dos anos 80 e aproximadamente início dos anos 90, um grande número de produtos foi desenvolvido para explorar este método de organização de dados multidimensionais. Em 1986, Ralph Kimball, foi um dos fundadores da *Red Brick Systems* para desenvolver um sistema de bancos de relacional desenhado especificamente para tratar consultas em um modelo estrela. Entretanto, a linguagem SQL como implementada no início dos anos 90 implicou em sérios problemas de performance na implementação de cubos OLAP. Alguns destes problemas foram:

- GROUP BY: Não podem ser usados para retornar os resultados de células com diferentes níveis de agregação. Isto significa que muitas consultas diferentes foram necessárias para obter todos os valores solicitados.

- Consultas altamente agregadas podem referenciar virtualmente todas as linhas da tabela fato de maneira que poucos valores das células são retornados.

- Geralmente não existia memória para armazenar os resultados de grandes cálculos, ou seja, para cada consulta era necessário reiniciar o processamento. Como agregações para vários níveis eram calculados, as linhas da tabela de fatos eram lidas repetidamente.

- SQL suportava apenas algumas funções de agregação (SUM, COUNT, AVG, MIN e MAX), entretanto alguns sistemas de bancos de dados relacionais tinham extensões com capacidades analíticas adicionais.

- Os catálogos de bancos de dados não continham informações sobre os meta dados dos modelos estrela. Como resultado, modelos estrela tinham que ser "descritos" diversas vezes – uma vez para cada aplicação ou ferramenta usada.

- Como os metadados não faziam parte dos catálogos de bancos de dados, informações estruturais de chaves ficavam indisponíveis para os compiladores de consulta.

Em geral, a solução encontrada para resolver os problemas de performance nos sistemas ROLAP foi manter um número como sumário da tabela fato e tabelas dimensão associadas. Em tal sistema, sempre que dados na tabela fato são modificados, a tabela sumário precisa ser ajustada. Apesar desses problemas, sistemas ROLAP foram largamente implementados. Em tempo de consultas, se a tabela sumário não estiver disponível com a agregação desejada, o sistema OLAP deve escolher uma tabela sumário apropriada contendo parcialmente resultados agregados ou então consultar a base da tabela fato [COLOSSI, 2002].

Uma vantagem na adoção de uma solução ROLAP reside na utilização de uma tecnologia estabelecida, de arquitetura aberta e padronizada como é a relacional, beneficiando-se da diversidade de plataformas, escalabilidade e paralelismo de hardware [FIGUEIREDO, 1998].