UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEXANDRE SCHLÖTTGEN

Modelo de Gerenciamento de Versões para Evolução de *Data Warehouses*

Dissertação apresentada como requisito parcial,
para a obtenção do grau de
Mestre em Ciência da Computação

Profª. Dra. Nina Edelweiss
Orientadora

Profº. Dr. Clesio Saraiva dos Santos
Co-orientador

Porto Alegre, abril de 2004.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Schlöttgen, Alexandre

Modelo de Gerenciamento de Versões para Evolução de *Data Warehouses* / Alexandre Schlöttgen. – Porto Alegre: PPGC da UFRGS,2004.

120 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2004. Orientadora: Nina Edelweiss.

1. *Data Warehouse*. 2. Modelagem Multidimensional. 3. OLAP 4. Evolução de esquemas multidimensionais. I. Edelweiss, Nina. II. Clesio S. dos Santos. III Título.

AGRADECIMENTOS

Primeiramente, agradeço ao meus pais, Ivoni e Luiz, pelos fundamentos da minha educação, conselhos e apoio nos momentos difíceis. Agradeço também a minha irmã, Cristina, pelo companheirismo e a minha namorada Carol, que teve muita paciência, me ajudando sempre.

Um agradecimento especial à professora Nina, pela orientação dedicada e competente. Você me ensinou muitas coisas importantes durante o mestrado. Obrigado por todo o apoio e principalmente pela sua amizade.

Aos professores e funcionários do Instituto de Informática agradeço pelo profissionalismo e pelo ótimo relacionamento que tivemos nestes anos. O que torna este instituto um centro de excelência no país é, sem dúvida, as pessoas que nele trabalham.

Aos meus colegas de mestrado, obrigado pelas dicas, trocas de idéias e principalmente pela amizade, foram muito boas as conversas de corredor, o congresso de Gramado, etc.

Aos meus chefes e colegas de trabalho, meu sincero obrigado por permitirem que eu fizesse o mestrado, segurando as pontas enquanto eu estava nas aulas e reuniões do mestrado.

Por fim, gostaria de dedicar este trabalho de mestrado e o título que me é conferido por ele, ao meu avô Reinhold Holz, que me ensinou a dar valor aos estudos e que deve estar muito feliz por mais esta conquista.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
DBA	*Data Base Administrator*
DM	*Data Mart*
DW	*Data Warehouse*
E/R	Entity/Relationship
ID	Identificador Único
MVB	Matriz de Valores Base
OLAP	*On-line Analytic Processing*
OLTP	*On-line Transaction Processing*
OO	Orientado a Objetos
SAD	Sistema de Apoio à Decisão
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Sistemas de tomada de decisão baseados em *Data Warehouse* (DW) estão sendo cada dia mais utilizados por grandes empresas e organizações. O modelo multidimensional de organização dos dados utilizado por estes sistemas, juntamente com as técnicas de processamento analítico on-line (OLAP), permitem análises complexas sobre o histórico dos negócios através de uma simples e intuitiva interface de consulta.

Apesar dos DWs armazenarem dados históricos por natureza, as estruturas de organização e classificação destes dados, chamadas de dimensões, não possuem a rigor uma representação temporal, refletindo somente a estrutura corrente. Para um sistema destinado à análise de dados, a falta do histórico das dimensões impossibilita consultas sobre o ambiente real de contextualização dos dados passados. Além disso, as alterações dos esquemas multidimensionais precisam ser assistidas e gerenciadas por um modelo de evolução, de forma a garantir a consistência e integridade do modelo multidimensional sem a perda de informações relevantes.

Neste trabalho são apresentadas dezessete operações de alteração de esquema e sete operações de alteração de instâncias para modelos multidimensionais de DW. Um modelo de versões, baseado na associação de intervalos de validade aos esquemas e instâncias, é proposto para o gerenciamento dessas operações. Todo o histórico de definições e de dados do DW é mantido por esse modelo, permitindo análises completas dos dados passados e da evolução do DW.

Além de suportar consultas históricas sobre as definições e as instâncias do DW, o modelo também permite a manutenção de mais de um esquema ativo simultaneamente. Isto é, dois ou mais esquemas podem continuar a ter seus dados atualizados periodicamente, permitindo assim que as aplicações possam consultar dados recentes utilizando diferentes versões de esquema.

Palavras-chave: Data Warehouse, modelagem multidimensional, OLAP, evolução de esquemas multidimensionais.

Version Management Model for Data Warehouse Evolution

ABSTRACT

The use of decision-making systems based on Data Warehouse (DW) by big companies and organizations is growing rapidly. The multidimensional model for organizing data used by these systems, together with the on-line analytical processing techniques (OLAP), allows complex analysis of the business history through a simple and intuitive query interface.

Even though it is inherent for the DWs to store historical data, the structures for organization and classification of these data, called dimensions, do not actually have a temporal representation, reflecting only the current structure. Considering a system aiming data analysis, querying the real contextualization environment of the past data is not possible due to the absence of the dimensions history. Besides, the changes on multidimensional schemas need to be assisted and managed by an evolution model, in order to guarantee the consistency and integrity of the multidimensional model with no loss of relevant information.

This work presents seventeen schema changing operations and seven changing operations for instances of DW multidimensional models. A version model based on the association between validity intervals and schemas or instances is proposed to manage these operations. This model maintains the whole definition and data history of the DW, allowing complete analysis of past data and of the DW evolution.

Besides supporting historical queries on the DW definitions and instances, the model also allows maintaining more than one active schema simultaneously. This means that two or more schemas can keep updating their data periodically, allowing this way the applications to query recent data using different versions of the system.

Keywords: Data Warehouse, multidimensional modeling, OLAP, multidimensional schemes evolution.

1 INTRODUÇÃO

Um esquema de banco de dados (BD) costuma sofrer muitas alterações ao longo do seu ciclo de vida. Em um BD relacional, estudos apontam um aumento médio de 139% no número de relacionamentos e de 274% no número dos atributos em relação à versão inicial do esquema (Blaschka, 2000). O controle consistente e inteligente das mudanças estruturais, das instâncias de dados e das aplicações que se utilizam destes dados é de fundamental importância para qualquer sistema de informação.

Para *Data Warehouses* (DWs), grandes bancos de dados baseados em assuntos, integrados, não-voláteis, que armazenam informações históricas sobre a evolução dos negócios (valores de indicadores variáveis em relação ao tempo) de uma empresa para utilização em sistemas de apoio à decisão (Inmon, 1997), o entendimento e o gerenciamento de suas modificações é crucial para a análise correta das consultas realizadas. É também muito importante possuir conhecimento sobre as alterações ocorridas nas instâncias (dados da extensão do BD), principalmente mudanças dos dados relativos às dimensões. Estas mudanças muitas vezes, ocasionam interpretações incorretas dos resultados das consultas relativas a um período de tempo no qual ocorreram alterações na constituição das dimensões (Body et al., 2002). Para minimizar os problemas da alteração do esquema de dados é proposto, neste trabalho, um modelo de versionamento dos esquemas multidimensionais que permite a consulta aos dados em qualquer versão do esquema. Entre as principais vantagens de um modelo específico para o armazenamento e manipulação de versões de esquema de DW destacam-se:

- controle da consistência da estrutura dimensional - DWs possuem regras específicas de integridade inerentes ao modelo multidimensional que podem ser automaticamente verificadas na criação da nova versão do esquema;
- nenhuma perda de dados - as exclusões de entidades, atributos ou relacionamentos ocorrem somente na nova versão do esquema. As instâncias continuam existindo inalteradas na versão anterior;
- atualização futura das aplicações do DW - não há necessidade de alterar as configurações, códigos fontes ou executar compilações dos aplicativos, pois eles permanecem referenciando a versão anterior do esquema. Isto permite uma alteração mais suave e gradual das aplicações para o novo esquema;
- armazenamento do histórico de evolução do esquema - permitindo a análise e o entendimento do desenvolvimento e da evolução do DW.

Para controlar as alterações no esquema é proposto um modelo de versões para o DW, de forma que a cada novo esquema gerado (conjunto de alterações nos dados da intenção) seja gerada uma nova versão do DW. Para diferenciar as versões foi

adicionado um elemento temporal aos dados e metadados, indicando o dia em que a versão foi criada. Para o controle dos dados da extensão de uma versão são utilizados dois elementos temporais: início e fim de validade do dado, caracterizando o período em que os dados são válidos. Entre as vantagens da utilização deste modelo destacam-se:

- possibilidade de consulta das alterações sofridas na constituição dos membros de uma dimensão - podemos verificar todas as mudanças de uma certa dimensão e utilizar estas informações para um melhor entendimento dos resultados das consultas;
- análise dos dados com a configuração das dimensões de um instante do tempo - consiste em pesquisar um período ($T_{início}$, T_{fim}) sob a constituição dos membros das dimensões de um certo instante T. Os DWs convencionais somente permitem consultas sob a configuração atual das dimensões;
- juntando o versionamento de esquemas e o armazenamento temporal das alterações dos dados do DW temos um ótimo controle sobre as evoluções do DW permitindo consultas sofisticadas baseadas na realidade de um instante de tempo.

Para explicar as propriedades e regras do modelo de versões, bem como o controle das alterações nos esquemas e suas instâncias este trabalho foi dividido em oito capítulos orientados aos seguintes assuntos:

- capítulo 2 – apresenta os principais conceitos de DW necessários para o entendimento do modelo multidimensional. Neste capítulo é introduzido um modelo de dados para a representação de DWs, mas sem a preocupação com o armazenamento e controle das alterações dos esquema e das instâncias. Também é apresentado um estudo de caso de modelagem de um pequeno DW que é referenciado pelos outros capítulos;
- capítulo 3 – discute o comportamento evolutivo de DWs e as implicações das alterações dos esquemas e dos dados, mostrando a necessidade de um controle seguro de adaptação das instâncias numa mudança do esquema. Neste capítulo são apresentados os principais trabalhos da área de evolução de esquemas e dados de modelos multidimensionais;
- capítulo 4 – apresenta um conjunto de vinte e quatro operações atômicas de alteração de esquema e dados que são gerenciadas pelo modelo de versões. Também são discutidas algumas funções complexas de alteração de dados formadas pela composição das funções atômicas;
- capítulo 5 – explica o armazenamento e o controle das versões de esquema. Mostra como deve ser feita a criação de novos esquemas, a adaptação das instâncias e o acesso dos aplicativos às versões. É o capítulo central deste trabalho;
- capítulo 6 – adiciona ao modelo de versões de esquema o armazenamento do histórico de valores de suas instâncias, permitindo assim consulta aos valores passados das instâncias de um esquema;
- capítulo 7 – apresenta os mapeamentos necessários para a utilização do modelo de versões para o controle da evolução de outros modelos multidimensionais;
- capítulo 8 – resume as principais contribuições e conclusões deste trabalho, além de indicar futuros trabalhos para o aprimoramento do modelo.

2 CONCEITOS BÁSICOS

Este capítulo tem como objetivo apresentar os conceitos básicos sobre *Data Warehouse* necessários para o entendimento dos outros capítulos. Além de explicar o que é um DW, são introduzidos os seguintes conceitos:

- *Data Mart*;
- Consultas OLAP;
- Arquiteturas de DW;
- Modelo de dados para DW (fatos e dimensões);
- Hipercubo de dados.

É apresentado também um novo modelo de dados para a definição de DWs, definido neste trabalho, que servirá de base para a compreensão e a construção de modelos mais complexos com suporte ao versionamento de esquemas e dados (capítulos cinco e seis). Para facilitar a compreensão desse modelo foi criado um exemplo fictício de DW, que também será referenciado nos outros capítulos para exemplificar novos conceitos e definições.

2.1 Data Warehouse

Os sistemas de banco de dados evoluíram muito nas últimas décadas agregando muitas funcionalidades ao seu objetivo principal, que é manter e disponibilizar dados para as aplicações computacionais. Paralelo a isto, ocorreu a especialização dos SGBDs (Sistemas de Gerenciamento de Banco de Dados) para o tratamento diferenciado do armazenamento e acesso às informações levando em conta não somente a natureza e relacionamento dos dados, mas também as necessidades das aplicações. Neste contexto, surgiram os Sistemas de Apoio à Decisão (SAD) com seus SGBDs e ferramentas específicas para a manipulação de informações analíticas. Dentre estas ferramentas destaca-se a tecnologia de *Data Warehousing* que é considerada a evolução natural dos Ambientes de Apoio à Decisão (Machado, 2000).

Data Warehouse, cuja tradução literal é Armazém de Dados (Machado, 2000), pode ser definido como um banco de dados destinado a sistemas de apoio à decisão, cujos dados foram armazenados em estruturas lógicas dimensionais, possibilitando o seu processamento analítico por ferramentas especiais. O conceito de DW objetiva a definição de uma base de dados preparada em vários níveis de granularidade e obtida a partir de outros sistemas computacionais da organização (*legacy systems*). A idéia é extrair dados analíticos dos sistemas de produção, transformá-los e armazená-los em vários graus de relacionamento e sumarização, de forma a facilitar e agilizar os processos de tomada de decisão. (Barbieri, 2001)

A expressão *Data Warehouse* é utilizada tanto como referência a todo o ambiente de tomada de decisão (extração, integração e carga dos dados, SGBD, consultas), como referência única ao SGBD. Ou seja, dependendo do contexto, pode indicar somente o repositório de dados e não o sistema como um todo (Pereira, 1999).

2.2 Arquitetura de DW

A arquitetura de um DW pode denotar tanto os elementos funcionais (extração, transformação, carga, interface de consulta, apresentação dos resultados das consultas, etc), quanto o tipo de armazenamento, distribuição física e acesso aos dados (Pereira, 1999). Neste trabalho utilizaremos o termo arquitetura no sentido de armazenamento e acesso aos dados.

Existem genericamente dois tipos de arquiteturas: a centralizada e a distribuída (Inmon, 1997). Na centralizada, existe um único ambiente e repositório de dados responsável por todos os dados gerenciais da organização, enquanto que na distribuída existem vários repositórios e ambientes independentes espalhados pelos vários departamentos de uma empresa, sendo cada um destes responsável pela aquisição, carga e distribuição de seus dados.

A arquitetura distribuída, por sua vez, possui muitas variantes, dentre as quais se destacam a arquitetura local e a arquitetura global ou integrada (Inmon, 1997; Machado 2000; Barbieri, 2001). Na arquitetura local, cada departamento, filial ou área da organização possui o seu DW independente, não havendo coordenação de dados ou estrutura de dados de um D*ata Warehouse* local para outro. Já na arquitetura global cada área é responsável por seus dados, mas a estrutura dos dados é definida em comum acordo pelas diversas áreas. Ou seja, existe uma preocupação com a possibilidade de integração dos dados, possibilitando consultas distribuídas sobre dados de mais de um DW.

A arquitetura básica utilizada para o desenvolvimento deste trabalho é a de um DW centralizado, onde todos os dados são armazenados e gerenciados por um único SGBD. Esta escolha se deve ao fato da arquitetura centralizada ser a alternativa ideal do ponto de vista de manutenção, pois todos os dados estão armazenados em um único local, evitando assim problemas de sincronização e integração dos dados, comuns na arquitetura distribuída.

Um DW de arquitetura centralizada, apesar do nome, não precisa possuir necessariamente somente um único SGBD para acesso aos dados, podendo ser feitas várias cópias redundantes de partes de seus dados para SGBDs secundários, distribuindo assim o acesso aos dados. O que não pode ocorrer na arquitetura centralizada é a existência de dados diferentes entre o repositório central e os repositórios secundários.

Estes SGBDs secundários são chamados na literatura de *Data Marts* (DM) e servem como repositórios intermediários entre o DW e o usuário final, permitindo um aumento de performance devido ao menor volume e à localidade dos dados (o DW pode estar no CPD da matriz da empresa e os DMs distribuídos nas redes locais das filiais). Além disso, eles aumentam a segurança de acesso aos dados, pois os usuários finais consultam somente os dados dos DMs para os quais possuem direito de acesso e não todo o DW. Podemos dizer que os DMs são vistos como clientes ou aplicações pelo SGBD do DW, que seria um servidor de estruturas multidimensionais. A figura 2.1 exemplifica este tipo de arquitetura.

Já no caso da arquitetura distribuída, cada DW de uma área é também chamado de *Data Mart*, de forma que podemos dizer que neste caso um DM é um pequeno DW.

Alguns autores, como Inmon (1997), definem que o DW deve armazenar somente os dados mais detalhados sob uma forma relacional e não redundante enquanto que os DMs devem ser carregados com os dados do DW sumarizados em uma estrutura multidimensional de rápido acesso, com poucos dados e muita redundância.

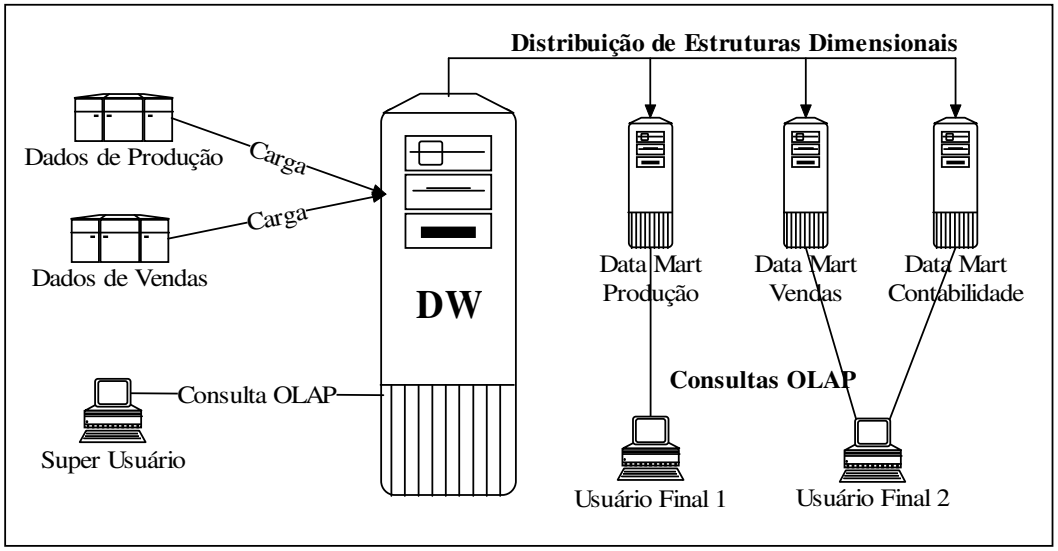

Figura 2.1: Exemplo de arquitetura centralizada

2.3 Modelo de Dados para *Data Warehouse*

Um DW armazena basicamente dois tipos de dados: numéricos e descritivos (ou classificatórios), sendo os dados numéricos chamados de fatos e os descritivos de dimensões. As dimensões possuem os dados estáticos que classificam e descrevem os fatos,que são os valores dinâmicos do DW (Kimball, 1996).

Os fatos guardam um histórico de valores relacionando-os com as dimensões participantes do fato. Eles são ditos dinâmicos, pois crescem ao longo do tempo, uma vez que novas medidas são adicionadas a cada nova carga do DW (Cabibbo; Torloni, 1998). As dimensões não costumam ter modificações significativas na constituição de seus membros e nos seus dados descritivos, tal como ocorre com os fatos.

Os fatos são as entidades centrais do DW e as dimensões constituem as estruturas de pesquisa e classificação dos fatos (Abelló; Samos; Saltor, 2001). O modelo de relacionamento de um fato com suas dimensões é chamado de modelo dimensional (Kimball, 1996). Outro nome dado a este tipo de modelo é esquema estrela (s*tar schema*), pois sua representação gráfica lembra o formato de uma estrela, com a tabela fato no centro e as dimensões nas pontas (figura 2.2). Um DW é constituído por vários esquemas estrelas, pois ele possui inúmeros fatos com várias dimensões compartilhadas.

O esquema estrela, por sua vez, deu origem a diversas variantes, principalmente nos modelos lógicos e físicos, adaptando-se as diferentes necessidades e produtos comerciais. Entre estes, os mais utilizados nos BDs relacionais são os modelos: *StarER*,

que possui os níveis de dimensões desnormalizados e *SnowFlake* (floco de neve), que possui os níveis de dimensões normalizados, formando uma hierarquia de níveis explícita e não redundante (Gopalkrishnan et al., 1999).

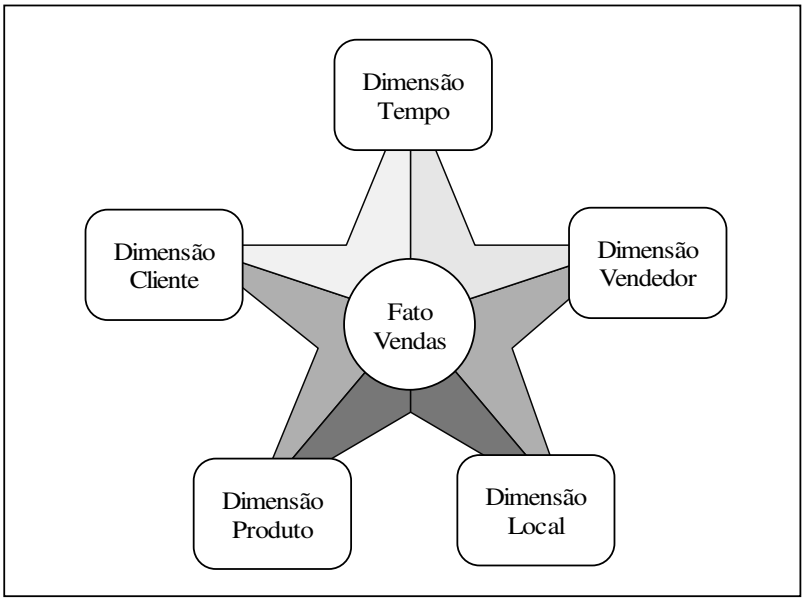

Figura 2.2: Modelo Estrela

2.3.1 Propriedades das Dimensões

As dimensões são ortogonais (independentes entre si) e são compostas por membros organizados em níveis hierárquicos para que seja possível executar as funções OLAP de *Drill-Down e Roll-Up* que aumentam ou diminuem o nível de detalhamento dos dados de uma consulta (Barbieri, 2001; Morzy; Krembel, 2003). Para a dimensão Tempo, por exemplo, podemos ter os seguintes membros: Datas (01/08/2002, 05/10/2003, 08/10/2003, 07/11/2003, 05/12/2003), Meses (10/2002, 10/2003, 11/2003, 12/2003) e Anos (2002, 2003) como é mostrado da figura 2.3. Estes membros são organizados em 3 níveis hierárquicos: data, mês e ano, sendo que os valores (métricas) associados a 05/10/2003 e 07/10/2003 irão compor a métrica do mês 10/2003, que por sua vez irá compor a métrica do ano de 2003 juntamente com os valores dos meses 11/2003 e 12/2003.

A divisão hierárquica de uma dimensão é chamada de hierarquia de classificação e pode possuir diferentes caminhos hierárquicos (Abelló; Samos; Saltor, 2001). A maioria dos autores costuma representar a hierarquia de classificação através de uma grafo acíclico dirigido, onde os nodos representam os níveis e os arcos apontam os caminhos possíveis de sumarização. (Sapia et al., 1999; Blaschka, 2000; Nguyen; Tjoa, Wagner, 2000; Trujillo; Palomar; Gomez, 2001).

A figura 2.4 mostra uma hierarquia de classificação para a dimensão Tempo com três caminhos alternativos:

- Data => Mês => Bimestre => Semestre => Ano;
- Data => Mês => Trimestre => Semestre => Ano;

- Data => Semana => Estação.

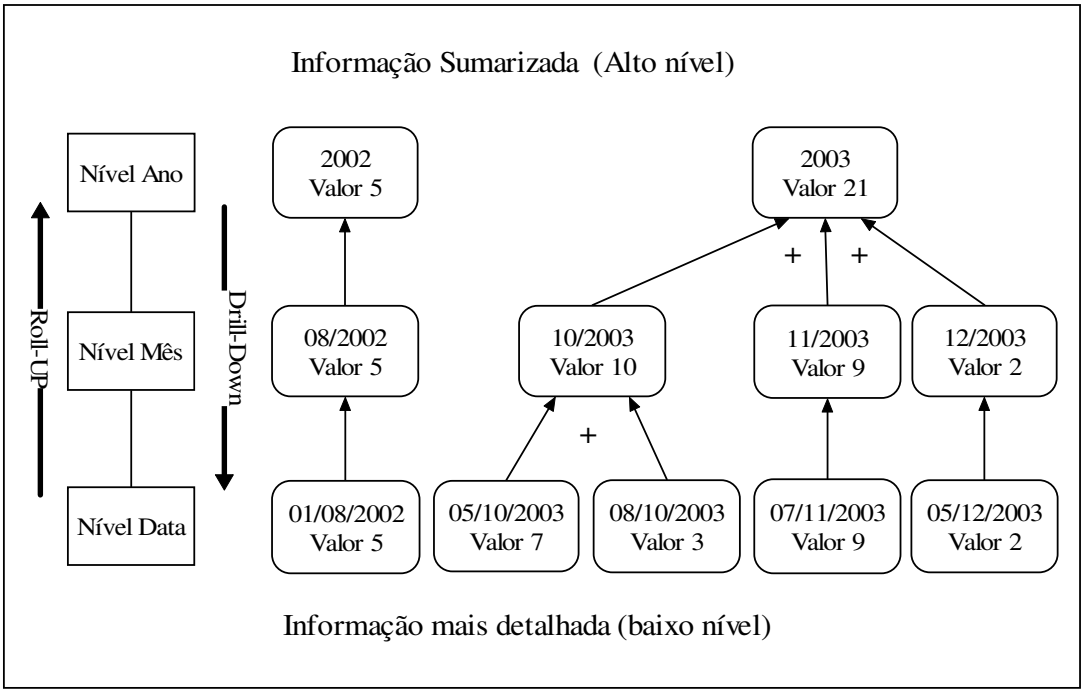

Figura 2.3: Exemplo de hierarquia para a dimensão tempo

Cada membro de dimensão possui quatro informações indispensáveis:

- um identificador único (ID) utilizado nos relacionamentos internos do DW (geralmente um número binário ou hexadecimal);
- um nome para identificação pelo usuário;
- a indicação do nível hierárquico ao qual pertencem;
- uma lista de IDs seus membros superiores (ou ancestrais), isto é, os membros pais que utilizam ele (filho ou descendente) para sumarizar dados.

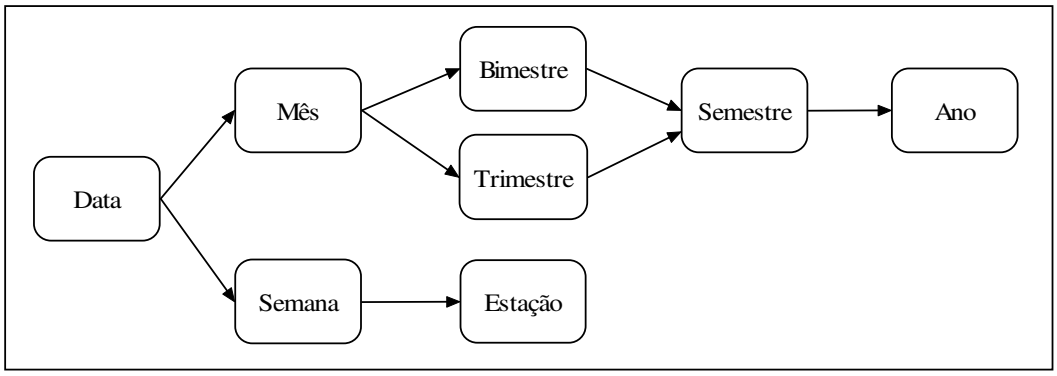

Figura 2.4: Caminhos hierárquicos para a dimensão tempo

Na figura 2.3, o membro de nome 10/2003, pertence ao nível Mês e possui como membros superiores somente o membro chamado 2003. Já no caso da hierarquia da figura 3.3, os membros pertencentes ao nível Data precisam informar o ID de seus dois pais: Mês e Semana. O mesmo ocorre para o nível Mês (Bimestre e Trimestre). A lista de níveis superiores é nula para os últimos níveis da árvore hierárquica (no caso das figura, os níveis Ano e Estação).

Além disso, os membros de dimensão podem possuir atributos descritivos para caracterizá-los melhor (Hüsemann et al., 2000). Estes atributos podem servir de parâmetros em operações de seleção de membros ou como informação adicional nos resultados das consulta. Os atributos descritivos são definidos nos níveis de dimensão. Para o nível Data, por exemplo, poderíamos ter os atributos: Flag de Feriado, Flag de Fim de Semana , Dia da Semana, etc.

2.3.2 Propriedades dos Fatos

Os fatos são constituídos por um conjunto de elementos formados por todas as combinações possíveis de membros de dimensões distintas (Golfarelli; Maio; Rizzi, 1998). Cada um destes elementos possui uma lista de métricas associadas. Em outras palavras, um fato relaciona um conjunto de dimensões e fornece um conjunto de medidas (métricas) válidas para o ponto de encontro (interseção) dessas dimensões (Abelló; Samos; Saltor, 2001). Um fato não precisa ter um ID, pois ele pode ser representado pela lista de IDs dos membros das dimensões com os quais se relaciona.

Cada linha da tabela 2.1 indica um elemento do Fato Vendas, que é composto por cinco membros de dimensões distintas (Tempo, Local, Produto, Vendedor, Cliente) e duas métricas (Valor da Venda e Número de Prestações).

Uma informação muito importante referente aos fatos é como eles devem ser agrupados e calculados (sumarizados) nos diversos níveis das dimensões. As métricas dos fatos podem utilizar funções diferentes para as totalizações nos níveis hierárquicos superiores das dimensões ou até não permitir totalizações. Na tabela 2.1, por exemplo, podemos utilizar uma função de soma para compor os totais das métricas Valor da Venda e Número de Prestações. Mas qual o valor da informação do número total de prestações efetuadas na cidade de São Paulo? Talvez seja mais interessante utilizar uma função de média geral ou média ponderada (em relação ao valor da venda), para se ter uma idéia do número de prestações das compras efetuadas.

Tabela 2.1: Exemplo da composição do fato Vendas

Dimensões					**Métricas**	
Tempo	**Local**	**Produto**	**Vendedor**	**Cliente**	**Valor da Venda**	**Nro de Prestações**
10/2003	Piauí	Mesa	João Silva	Maria Lima	R$ 130,00	3
10/2003	Piauí	Mesa	João Silva	Lucas Pinto	R$ 120,00	2
10/2003	Piauí	Sofá	João Silva	Maria Lima	R$ 250,00	4
10/2003	São Paulo	Mesa	Ana Souza	Fábio Silva	R$ 150,00	3
10/2003	São Paulo	Mesa	Ana Souza	Lúcia Mendes	R$ 150,00	3
10/2003	São Paulo	Mesa	José Pedro	Ana Maria	R$ 200,00	4
10/2003	São Paulo	Sofá	José Pedro	Mário Fontes	R$ 500,00	1

A tabela 2.2 apresenta os mesmos fatos da tabela 2.1 sumarizados em alguns níveis mais altos, com a métrica Valor de Venda utilizando a função soma e a métrica Número de Prestações utilizando a função média geral. Todas as dimensões possuem o nível Todos que caracteriza o nível mais alto de qualquer dimensão e que geralmente é suprimido das representações das árvores hierárquicas. O nível Todos significa desconsideração da dimensão em questão para a consulta realizada, pois não existe classificações distintas neste nível que é formado por somente um membro.

Tabela 2.2: Fato vendas sumarizado em níveis mais altos

Dimensões					**Métricas**	
Tempo	**Local**	**Produto**	**Vendedor**	**Cliente**	**Valor da Venda (soma)**	**Nro de Prestações (média)**
2003	PA	Todos	Todos	Todos	R$ 500,00	3
2003	SP	Todos	Todos	Todos	R$ 1.000,00	3
2003	PA	Mesa	Todos	Todos	R$ 250,00	2,5
2003	SP	Mesa	Todos	Todos	R$ 500,00	3,3
2003	PA	Sofá	Todos	Todos	R$ 250,00	4
2003	SP	Sofá	Todos	Todos	R$ 500,00	2

2.4 Hipercubos de Dados

A forma mais utilizada para a compreensão e visualização dos dados de um DW é imaginá-lo como um conjunto de cubos, onde cada cubo representa uma fato do negócio. O cubo representa um conjunto de métricas que compartilham o mesmo conjunto de dimensões (Nguyen; Tjoa; Wagner, 2000).

Um cubo é formado por um conjunto ordenado de células, onde cada célula é localizada pela intersecção de suas três dimensões (altura, largura e profundidade). A célula por sua vez, armazena os valores das métricas para a respectiva localização da célula.

Obviamente, a maioria dos fatos de um DW possue mais de três dimensões, não podendo ser corretamente representados na imagem de um cubo (elemento 3D), de forma que alguns autores utilizam a expressão "hipercubo". Neste trabalho será utilizada a expressão "cubo".

As ferramentas utilizadas para as consultas OLAP utilizam-se desta representação tridimensional dos dados para definir suas principais operações de consulta. Estas operações são baseadas na manipulação e visualização de um cubo físico (real) por uma pessoa, como por exemplo:

- rotações de 90º do objeto (mudança da posição dos eixos x, y, z);
- visualização plana de um dos lados (esconder uma das dimensões);
- corte de pedaços (seleção de partes do cubo);
- focalização de detalhes através da aproximação do objeto (ver dados em nível maior de detalhes).

A maioria dos operadores das consultas OLAP tem como base estas operações simples e intuitivas, só que adaptadas para a manipulação de dados muldimensionais (n dimensões) de forma que o usuário não precise de muito conhecimento para formalizar suas consultas.

2.5 Consultas OLAP

As consultas analíticas feitas em um DW ou DM têm como objetivo verificar o comportamento de métricas do negócio (dados numéricos, medidas) ao longo do tempo. Estas consultas são normalmente chamadas de consultas OLAP *(On-line Analytic Processing*), tendo como principais características o cálculo de valores contidos em uma enorme quantidade de registros e a apresentação dos resultados em formatos de mais alto nível, como tabelas ou gráficos.

As consultas OLAP utilizam-se de parâmetros para selecionar membros de algumas dimensões e retornam os valores encontrados na intersecção destes membros. O resultado geralmente é mostrado em forma de tabelas ou gráficos. As ferramentas OLAP (aplicativos utilizados para a montagem, execução e visualização das consultas) possuem um reduzido número de operadores dimensionais que agrupados, permitem realizar consultas complexas sem que o usuário final precise ter alto conhecimento da estrutura de armazenamento e da linguagem de consulta. Aliás, a grande maioria dos produtos comerciais possuem interfaces gráficas para a execução de consultas *ad-hoc*.

As consultas precisam que os dados estejam organizados e armazenados em uma forma multidimensional, para que as consultas realizadas sejam executadas no menor tempo de resposta possível. No modelo multidimensional, as estruturas físicas de armazenamento, a construção das consultas e a visualização dos resultados são muito parecidas.

2.6 Modelo para Data Warehouse

Para apresentar e formalizar as idéias dos próximos capítulos, foi definido neste trabalho um modelo conceitual genérico. Este modelo, além de definir detalhadamente as propriedades e relacionamentos dos elementos do DW, permitirá um melhor entendimento das mudanças necessárias para o controle das alterações e armazenamento do histórico dos dados.

Esse modelo de dados foi criado com base no estudo de outros modelos (Golfarelli; Maio; Rizzi, 1998; Cabibbo; Torlone, 1998; Sapia et. al, 1999; Kimball, 1996, Mangisengi; Tjoa; Wagner, 1999; Tryfona; Busborg; Christiansen, 1999; Agrawal; Gupta; Sarawagi; 1997, Franconi; Kamble; 2003, etc), reunindo os conceitos e características destes modelos de uma forma simples, didática e sem perda de expressividade.

A seguir são apresentadas as definições deste modelo e logo depois é mostrado um exemplo de DW (estudo de caso) utilizando estes formalismos.

2.6.1 Definição de Cubo

Um cubo é um *Star Schema* que descreve um fato. Ele possui um conjunto de métricas associadas a um mesmo conjunto de hierarquias de dimensões (HD). Ele é representado por uma tupla com quatro elementos:

$CUBO_{id}$ = <Id, Nome, F, CC>
F = {M_1, M_2, ..., M_n}
CC = {C_1, C_2, ..., C_n}

onde: Id = identificador único do cubo;
Nome = nome do cubo;
F = conjunto de métricas de um negócio com pelo menos uma métrica (fato);
CC = conjunto com os caminhos de classificação disponíveis para este cubo.
M_n = métrica (ID)
C_n = caminho de classificação (ID)

Os caminhos de classificação do cubo devem gerar árvores hierárquicas consistentes e com níveis dimensionais iguais ou superiores aos níveis base das métricas.

2.6.2 Definição de Métrica

Uma métrica (M) representa um indicador numérico. Ela define o tipo e domínio dos valores possíveis (inteiro positivo, decimal, real, moeda, etc), os níveis base (NB) das dimensões e utiliza uma função de agregação (soma, média) para calcular os valores dos níveis dimensionais superiores aos níveis de dimensão base. Uma métrica é definida por:

M_{id} = <Id, Nome, Tipo_Domínio, f_Agreg, NB >
NB = {N_1, N_2, ..., N_n}

onde: Id = identificador único da métrica;
Nome = nome da métrica;
Tipo_Domínio = tipo e domínio da métrica;
f_Agreg = função de agregação utilizada para composição dos valores dos níveis superiores das hierarquias de dimensões;
NB = lista dos níveis base de diferentes dimensões;
N_n = nível de dimensão (Id).

Exemplos de f_Agreg:

- SOMA - soma dos valores dos membros filhos;
- MÉDIA - soma dos valores dos membros filhos dividido pela sua quantidade;
- MAIOR - maior valor encontrado para um membro filho;
- MENOR - menor valor encontrado para um membro filho.

2.6.3 Definição de Dimensão

Uma dimensão (D) é representada por uma tupla de dois elementos:

D_{id} = <Id, Nome>

onde: Id = identificador único da dimensão;
Nome = nome da dimensão.

2.6.4 Definição de Nível de Dimensão

Uma dimensão é formada por um conjunto de níveis (N) que são representados por tuplas de três elementos:

N_{id} = <Id, Nome, D_{id}>

onde: Id = identificador único do nível de dimensão;
Nome = nome do nível de dimensão;
D_{id} = identificador da dimensão ao qual o nível faz parte.

Observação: todas as dimensões possuem o nível "Todos", que possui um único membro representado pelo símbolo "⊥". Esse nível é sempre o mais alto da árvore hierárquica e sua definição está implícita na criação de uma dimensão.

2.6.5 Definição de Atributo de Nível de Dimensão

Os níveis de dimensão podem possuir atributos descritivos (A) para melhor caracterizar os membros de dimensão (MB).

A_{id} = <Id, Nome, N_{id}, Tipo_domínio >

onde: Id = identificador único do atributo de nível de dimensão;
Nome = nome do atributo de dimensão;
N_{id} = nível do atributo de dimensão;
Tipo_Domínio = tipo e domínio do atributo de dimensão.

2.6.6 Definição de Caminho de Classificação

Os níveis de uma dimensão possuem relacionamentos de pai-filho para indicar a hierarquia de classificação da dimensão. Cada relacionamento de pai-filho entre dois níveis (caminho de classificação) é definido por:

C_{id} = <Id, NF_{id}, NP_{id}>

onde: Id = identificador único do caminho de classificação;
NP_{id} = identificador do nível pai (superior);
NF_{id} = identificador do nível filho (inferior).

2.6.7 Definição de Hierarquia de Níveis

O conjunto de todos os caminhos de classificação formam a hierarquia de classificação da dimensão (HD) que é representada por:

HD = {C_1, C_2, ..., C_n}

Sendo n, o número total de caminhos da dimensão.

2.6.8 Definição de Membro de Dimensão

Um membro de dimensão (MB) é uma instância de um nível de dimensão, sendo representado da seguinte forma:

$\mathbf{MD_{id} = <Id, Nome, N_{id}, MDP , MAD>}$

onde: Id = identificador único do Membro de Dimensão;
Nome = nome do Membro de Dimensão;
N_{id} = identificador do Nível de Dimensão ao qual o membro pertence;
MDP = conjunto de Membros Pais (Superiores imediatos);
MAD = conjunto ordenado de valores atribuidos aos atributos do nível.

2.6.9 Definição de Valor Base da Métrica

Um valor base da métrica representa o resultado associado à interseção dos membros bases de cada dimensão participante da métrica. A união de todos os valores formam a matriz multidimensional de valores base (MVB) que apresenta todas as interseções possíveis de membros de dimensões. A partir dos valores de MVB e da função de agregação utilizada é possível calcular os valores da métrica em todos os níveis de dimensões do conjunto de hierarquias de dimensões da métrica em questão.

$\mathbf{VB = <M_{id}, MD_{id\ x}\ MD_{id\ x\ ..} MD_{id}, VAL >}$

onde: M_{id} = identificador único da métrica;
MD_{id} = identificador único de membro de dimensão;
VAL = valor da métrica para a intersecção dos membros de dimensão indicados.

Todos os membros pertencem a dimensões diferentes e participam do nível básico definido para a métrica. O número de membros deve ser igual ao número de dimensões participantes da métrica.

2.6.10 Definição de *Data Mart* e *Data Warehouse*

Um *Data Mart* (DM) é um subconjunto dos cubos do *Data Warehouse* (DW). Isto é, o *Data Warehouse* possui todos os cubos necessários para cada um dos *Data Marts* da empresa. Além disso, um mesmo cubo pode ser utilizado por mais de um DM..

$\mathbf{DM = \{CUBO_1, CUBO_2, ..., CUBO_n\}}$
$\mathbf{DW = \{CUBO_1, CUBO_2, ..., CUBO_m\}}$

O número de hipercubos do DM (n) sempre é menor ou igual ao número de hipercubos totais do DW (m) ($n \leq m$). Isto implica em que:

$\mathbf{DW \subseteq DM \subseteq CUBO}$

2.7 Exemplo de DW (Estudo de Caso)

Para deixar mais claro o modelo utilizado, é apresentado um estudo de caso que será muito utilizado nos exemplos dos capítulos subseqüentes.

O DW escolhido para o estudo de caso é bem simples e retrata uma empresa que vende telefones celulares. Ele é composto por dois fatos: Vendas e Estoque e três dimensões: Local, Tempo e Produto. A figura 2.5 apresenta uma representação gráfica de alto nível do esquema do DW sugerido.

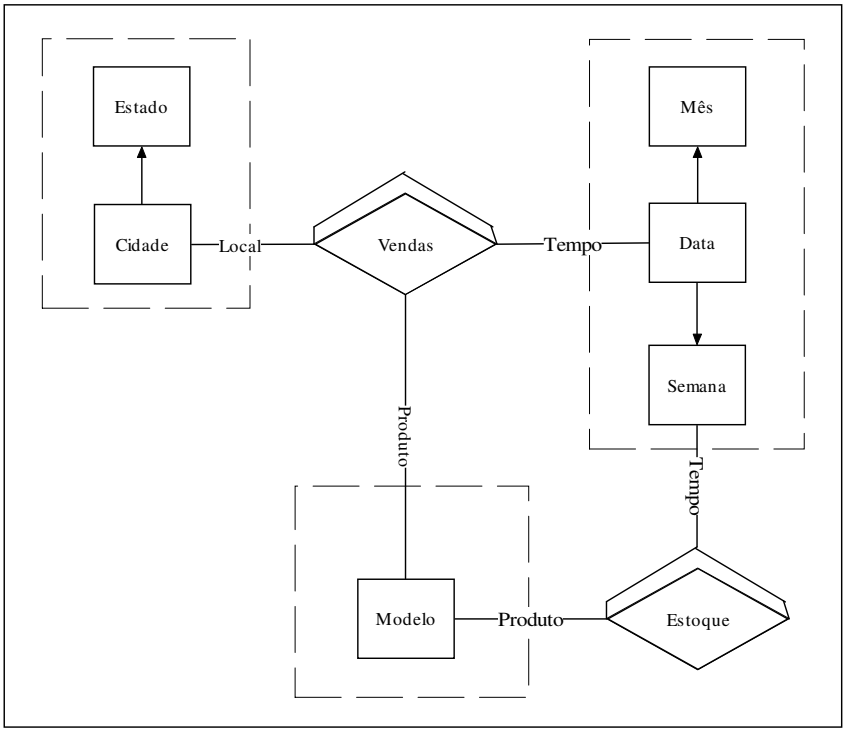

Figura 2.5: Esquema de alto nível para o exemplo de DW

O fato Vendas é composto por três dimensões: Local, Tempo e Produto. O fato Estoque é composto pelas dimensões Tempo e Produto. Além disso, a dimensão Tempo possui duas hierarquias diferentes, uma com os níveis Data, Mês e Semana para o fato Vendas e outra com somente o nível Semana para o fato Estoque.

Além disso, o fato Vendas possui duas métricas, quantidade e valor da venda, e o fato Estoque uma única métrica, quantidade de produtos estocados no depósito central da companhia.

As dimensões possuem os seguintes membros organizados em seus respectivos níveis dimensionais (ver figuras 2.6 e 2.7):

- Membros de Produto - {N5120, M3320};
- Membros de Local - {Canoas, Viamão, São Paulo, Campinas, RS, SP};

- Membros de Tempo - {26/01/2003, 28/01/2003, 01/02/2003, 03/02/2003, 5º/2003, 6º/2003, JAN/2003, FEV/2003}.

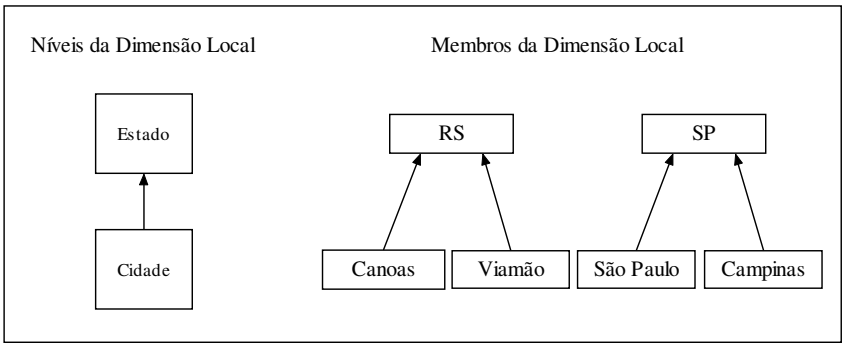

Figura 2.6: Membros da dimensão Local

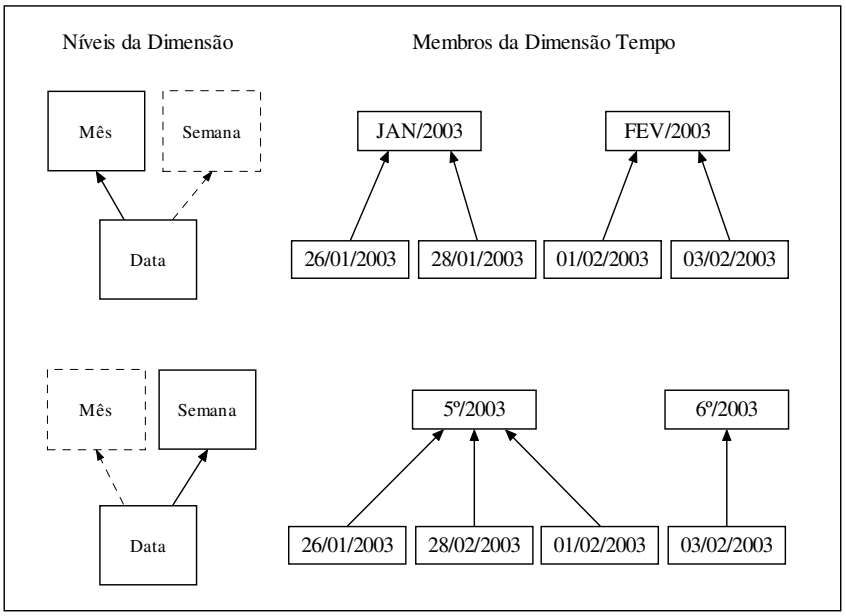

Figura 2.7: Membros da dimensão tempo

Os valores das métricas associadas aos membros bases (de menor granularidade) das dimensões são mostrados nas tabelas 2.3, 2.4 e 2.5, para as métricas quantidade de produtos venda, valor correspondente e quantidade de produtos no estoque, respectivamente.

Para melhorar a análise das vendas foram adicionados ao nível Data da dimensão Tempo dois atributos, Feriado e Dia da Semana, para permitir consultas mais elaboradas.

Tabela 2.3: Métrica quantidade de produtos vendidos do fato Vendas

Quantidade de Produtos Vendidos								
Data / Cidade	26/01/2003		28/01/2003		01/02/2003		03/02/2003	
	N5120	M3320	N5120	M3320	N5120	M3320	N5120	M3320
Canoas	4	2	1	5	2	0	1	9
Viamão	2	1	2	1	1	1	0	1
São Paulo	3	2	1	0	2	0	2	2
Campinas	4	1	2	0	1	0	0	2

Tabela 2.4: Métrica valor (preço) do fato Vendas

Valor das Vendas								
Data / Cidade	26/01/2003		28/01/2003		01/02/2003		03/02/2003	
	N5120	M3320	N5120	M3320	N5120	M3320	N5120	M3320
Canoas	$ 400	$ 400	$ 100	$ 1000	$ 200	-	$ 100	$ 1800
Viamão	$ 200	$ 200	$ 200	$ 200	$ 100	$ 200	-	$ 200
São Paulo	$ 300	$ 400	$ 100	-	$ 200	$ 200	$ 200	$ 400
Campinas	$ 400	$ 200	$ 200	-	$ 100	-	-	$ 400

Tabela 2.5: Métrica quantidade de produtos estocados do fato Estoque

Quantidade de Produtos no Estoque		
Data / Produto	5ª Semana de 2003	6ª Semana de 2003
N5120	10	4
M3320	20	14

2.8 Modelagem do Exemplo de DW através do Modelo Definido

Os dados (instâncias) e metadados (esquema) deste exemplo de DW são modelados da forma apresentada a seguir, segundo o modelo definido na seção 2.6 :

Definição do *Data Warehouse*:

$\mathbf{DW = \{CUBO_1, CUBO_2, ..., CUBO_m\}}$

$DW = \{CUBO_{vend}, CUBO_{est}\}$

Definição dos Cubos:

$\mathbf{CUBO_{id} = <Id, Nome, F, CC>}$

$CUBO_{vend} = <"vend", "Vendas", \{M_{\mathbf{nro\_vend}}, M_{\mathbf{valor\_vend}}\}, \{C_{cid->uf}, C_{uf->todos}, C_{dt->mes}, C_{mes->todos}, C_{pr->todos}\}>$

$CUBO_{est} = <"est", "Estoque", \{M_{\mathbf{nro\_est}}\}, \{C_{sem->todos}, C_{pr->todos}\}>$

Obs.: os identificadores únicos (Ids) geralmente são representado por números mas, para facilitar o entendimento, serão utilizados aqui nomes curtos na definição deste DW.

Definição das Dimensões:

$\mathbf{D_{id} = <Id, Nome>}$
D_{loc} = <"Loc", "Local" >
D_{temp} = <"Temp", "Tempo" >
D_{prod} = <"Prod", "Produto">

Definição dos Níveis das Dimensões:

$\mathbf{N_{id} = <Id, Nome, D_{id} >}$
N_{cid} = <"cid", "Cidade", D_{loc} >
N_{uf} = <"uf", "Estado", D_{loc} >
N_{dt} = <"dt", "Data", D_{temp} >
N_{sem} = <"sem", "Semana" , D_{temp} >
N_{mes} = <"mes", "Mês" , D_{temp} >
N_{pr} = <"pr", "Modelo do Produto", D_{pr} >

Definição dos Atributos dos Níveis de dimensão:

$\mathbf{A_{id}}$ **= <Id, Nome,** $\mathbf{N_{id}}$**, Tipo_domínio >**
$A_{feriado}$ = <"feriado", "É feriado", N_{dt}, Lógico(Sim/Não) >
$A_{diasemana}$ = <"diasemana", "Dia da Semana", N_{dt},
texto fixo(Domingo, Segunda, Terça, Quarta,
Quinta, Sexta, Sábado) >

Definição dos Caminhos de classificação:

$\mathbf{C_{id}}$ **= <Id,** $\mathbf{NF_{id}}$**,** $\mathbf{NP_{id}}$ **>**
$C_{cid->uf}$ = <"cid->uf", N_{cid}, N_{uf} >
$C_{dt->mes}$ = <"dt->mes", N_{dt}, N_{mes} >
$C_{dt->sem}$ = <"dt->sem", N_{dt}, N_{sem} >
$C_{mes->todos}$ = <"mes->todos", N_{dt}, ⊥ >
$C_{sem->todos}$ = <"sem->todos", N_{dt}, ⊥ >
$C_{uf->todos}$ = <"uf->todos", N_{dt}, ⊥ >
$C_{pr->todos}$ = <"pr->todos", N_{dt}, ⊥ >

Definição das Hierarquias das Dimensões:

Hierarquia da dimensão Tempo = {$C_{dt->mes}$, $C_{mes->todos}$, $C_{sem->todos}$}
Hierarquia da dimensão Local = {$C_{cid->uf}$,$C_{uf->todos}$}
Hierarquia da dimensão Produto = {$C_{pr->todos}$}

Definição dos Membros das Dimensões:

$\mathbf{MD_{id}}$ **= < Id , Nome ,** $\mathbf{N_{id}}$ **,MDP , MAD>**
MD_{cid1} = < "cid1" , "Canoas" , N_{cid}, {MD_{RS}},∅ >
MD_{cid2} = < "cid2" , "Viamão" , N_{cid}, {MD_{RS}},∅ >
MD_{cid3} = < "cid3" , "São Paulo", N_{cid}, {MD_{SP}},∅ >
MD_{cid4} = < "cid4" , "Campinas" , N_{cid}, {MD_{SP}},∅ >
MD_{prod1} = < "prod1", "N5120" , N_{prod1}, {⊥} ,∅ >
MD_{prod2} = < "prod2", "M3320" , N_{prod2}, {⊥}, ∅ >

$MD_{26/01/2003}$ = < "26/01/2003", "26/01/2003", N_{dt}, {MD_{5sem}, $MD_{jan2003}$}, {"não", "Domingo"} >
$MD_{28/01/2003}$ = < "28/01/2003", "28/01/2003", N_{dt}, {MD_{5sem}, $MD_{jan2003}$}, {"não", "Terça"} >
$MD_{01/02/2003}$ = < "01/02/2003", "01/02/2003", N_{dt}, {MD_{5sem}, $MD_{fev2003}$}, {"sim", "Sábado"} >
$MD_{03/02/2003}$ = < "03/02/2003", "03/02/2003", N_{dt}, {MD_{6sem}, $MD_{fev2003}$}, {"não", "Segunda"} >
MD_{RS} = < "RS", "RS", N_{uf} , {⊥}, ∅ >
MD_{SP} = < "SP", "SP", N_{uf} , {⊥}, ∅ >
MD_{5sem} = < "5sem" , "5ªsem/2003" , N_{sem}, {⊥} , ∅ >
MD_{6sem} = < "6sem" , "6ªsem/2003" , N_{sem}, {⊥} , ∅ >
$MD_{jan2003}$ = < "jan2003", "JAN/2003" , N_{mes} , {⊥}, ∅ >
$MD_{fev2003}$ = < "fev2003, "FEV2003" , N_{mes}, {⊥}, ∅ >

Definição das Métricas e Valores Base:

M_{id} = <Id, Nome, Tipo_Domínio, f_Agreg, NB>
M_{nro_vend} = <"nro_vend", "Nº de itens vendidos", Inteiro Positivo, SOMA, {"N_{dt}", "N_{cid}", "N_{pr}"}>

VB = <M_{id}, MD_{id} x MD_{id} x ..MD_{id}, VAL >

$VB_{i,j,m}$ = <M_{nro_vend}, MD_i x MD_j x MD_m, Val >

onde: i = identificador único de um membro do nível de dimensão N_{dt} (data);
j = identificador único de um membro do nível de dimensão N_{cid} (cidade);
m = identificador único de um membro do nível de dimensão N_{pr} (modelo do produto);
Val = valor do tipo inteiro positivo.

Exemplos de VBs da métrica "Nº de itens vendidos":

$VB_{26/01/2003, cid1, prod1}$ = <M_{nro_vend}, $MD_{26/01/2003}$ x MD_{cid1} x MD_{prod1}, 4 >
$VB_{26/01/2003, cid1, prod2}$ = <M_{nro_vend}, $MD_{26/01/2003}$ x MD_{cid1} x MD_{prod2}, 2 >
$VB_{26/01/2003, cid2, prod1}$ = <M_{nro_vend}, $MD_{26/01/2003}$ x MD_{cid2} x MD_{prod1}, 2 >
$VB_{26/01/2003, cid2, prod2}$ = <M_{nro_vend}, $MD_{26/01/2003}$ x MD_{cid2} x MD_{prod2}, 1 >
$VB_{28/01/2003, cid1, prod1}$ = <M_{nro_vend}, $MD_{28/01/2003}$ x MD_{cid1} x MD_{prod1}, 1 >

O conjunto de todos os valores base da métrica "Nº de itens vendidos" formam a MVB da tabela 2.6 que é uma Matriz tridimensional (Tempo, Local e Produto) com o valor da métrica associado a cada interseção dos três membros de dimensões nos seus níveis mais baixos. Esta matriz possui 32 elementos (**4** dias **x 4** cidades **x 2** produtos).

Tabela 2.6: Matriz tridimensional da métrica “Nº de itens vendidos”

Dimensão Tempo(T)	Dimensão Local(L)	Dimensão Produto(P)	Valor f(TxLxP)
26/01/2003	Canoas	N5120	4
26/01/2003	Canoas	M3320	2
26/01/2003	Viamão	N5120	2
26/01/2003	Viamão	M3320	1
26/01/2003	São Paulo	N5120	3
26/01/2003	São Paulo	M3320	2
26/01/2003	Campinas	N5120	4
26/01/2003	Campinas	M3320	1
28/01/2003	Canoas	N5120	1
28/01/2003	Canoas	M3320	5
28/01/2003	Viamão	N5120	2
28/01/2003	Viamão	M3320	1
28/01/2003	São Paulo	N5120	1
28/01/2003	São Paulo	M3320	0
28/01/2003	Campinas	N5120	2
28/01/2003	Campinas	M3320	0
01/02/2003	Canoas	N5120	2
01/02/2003	Canoas	M3320	0
01/02/2003	Viamão	N5120	1
01/02/2003	Viamão	M3320	1
01/02/2003	São Paulo	N5120	2
01/02/2003	São Paulo	M3320	1
01/02/2003	Campinas	N5120	1
01/02/2003	Campinas	M3320	0
03/02/2003	Canoas	N5120	1
03/02/2003	Canoas	M3320	9
03/02/2003	Viamão	N5120	0
03/02/2003	Viamão	M3320	1
03/02/2003	São Paulo	N5120	2
03/02/2003	São Paulo	M3320	2
03/02/2003	Campinas	N5120	0
03/02/2003	Campinas	M3320	2

M_{valor_vend} = <”valor_vend”, “Valor da Venda”,
Decimal Positivo, SOMA,{“N_{dt}”, “N_{cid}”, “N_{pr}”}>

$VB_{i,j,m}$ = <M_{nro_vend}, MD_i x MD_j x MD_m, Val >

onde: i = identificador único de um membro do nível de dimensão N_{dt} (data);
j = identificador único de um membro do nível de dimensão N_{cid} (cidade);
m = identificador único de um membro do nível de dimensão N_{pr} (modelo do produto);
Val = valor do tipo decimal positivo.

O conjunto de todos os valores base da métrica "Valor da Venda" forma uma MVB tridimensional (Tempo, Local e Produto) com o valor da métrica associado a cada interseção dos três membros de dimensões nos seus níveis mais baixos. Esta matriz possui 32 elementos (**4** dias **x 4** cidades **x 2** produtos).

M_{nro_est} = <"nro_est", "N° de itens no estoque",
Inteiro Positivo, MÉDIA, {N_{sem}, N_{pr} }>

$VB_{i,j}$ = <M_{nro_vend}, MD_i x MD_j, Val >

onde: i = identificador único de um membro do nível de dimensão N_{sem} (semana);
j = identificador único de um membro do nível de dimensão N_{pr} (modelo do produto);
Val = valor do tipo inteiro positivo.

O conjunto de todos os valores base da métrica "Nº de itens no estoque" formam uma MVB bidimensional (Tempo e Produto) com o valor da métrica associado a cada interseção dos dois membros de dimensões nos seus níveis mais baixos. Esta matriz possui 4 elementos (**2** semanas **x 2** produtos) e é representada pela tabela 2.7.

Tabela 2.7: Matriz para a métrica "Nº de itens no estoque"

Dimensão Tempo(T)	**Dimensão Produto(P)**	**Valor f(TxP)**
5ª Semana de 2003	N5120	10
5ª Semana de 2003	M3320	04
6ª Semana de 2003	N5120	20
6ª Semana de 2003	M3320	14

2.9 Considerações Finais

Neste capítulo foram apresentados os principais conceitos necessários para o desenvolvimento das idéias dos próximos capítulos, incluindo um novo modelo conceitual.

O modelo proposto, além de definir as métricas, dimensões e seus relacionamentos, permite a composição (definição) dos cubos que são, na abordagem do modelo, visões de partes do DW. Os cubos, por sua vez, são disponibilizados para os diferentes *Data Marts* da empresa, conforme a definição de arquitetura centralizada ou global (Inmon, 1999; Machado, 2000; Barbieri, 2001).

As definições citadas e exemplificadas através do estudo de caso servirão de referência para o entendimento das operações de evolução e para a construção de um modelo mais complexo, com controle de versões de esquema e dados. Esses assuntos serão tratados nos próximos capítulos.

3 EVOLUÇÃO DE DATA WAREHOUSES

Este capítulo analisa melhor o comportamento evolutivo dos dados e definições de um DW, citando suas principais características, motivos, conseqüências e problemas. Também são apresentados alguns trabalhos importantes na área de evolução de DWs, de forma a situar o leitor sobre os estudos e soluções de outros autores.

3.1 Comportamento Evolutivo de um DW

Assim como qualquer sistema de banco de dados, DWs sofrem diversas mudanças nos seus dados (instâncias) e metadados (definições do esquema). Entretanto, existem algumas peculiaridades no comportamento evolutivo de um DW, como por exemplo, o baixo número de operações de exclusão e alteração dos dados, insignificantes se comparados ao volume das operações de inserção de novos dados. Os usuários finais (administradores, gerentes e analistas de negócio), inclusive, não tem permissão para alterar os dados.

O que geralmente ocorre em um DW são cargas incrementais e periódicas de novos valores para as métricas relacionadas aos fatos ocorridos nos intervalos entre as atualizações dos dados. Também ocorrem inclusões de novos membros de dimensão, mas em menor escala, mas raramente é preciso alterar ou excluir algum dado do DW.

As alterações dos dados, quando ocorrem, costumam ser referentes às mudanças nos membros das dimensões:

- mudança do nome do membro;
- mudança dos atributos descritivos do membro;
- mudança dos membros pais ou membros filhos;
- mudança do nível do membro.

Alterações nos valores das métricas também podem ocorrer na prática devido a erros na conversão e integração dos dados ou devido a carga de dados errados vindos dos sistemas fontes. Estes tipos de erros são muito difíceis de serem detectados depois que os dados já estão consolidados no DW, de forma a serem muito raras as alterações corretivas nos valores das métricas.

Se por um lado as exclusões e alterações dos dados consolidados do DW ocorrem de forma lenta, por outro, as alterações no esquema de dados do DW costumam ser bem freqüentes devido a uma série de fatores, dentre os quais citamos:

- redução do espaço de armazenamento livre - DWs costumam crescer muito rapidamente, necessitando de alterações na granularidade mínima dos fatos;
- aumento de performance - diminuição do número de dimensões, alteração da granularidade, mudança das hierarquias, etc;

- mudança dos sistemas fontes - alteração ou falta de dados para composição das dimensões ou das granularidades mínimas;
- inclusão de novos assuntos ou requisitos - novas estruturas e dados para análises mais variadas e complexas (adição de novos fatos, métricas, dimensões, níveis de dimensão, atributos).

Além disso, as mudanças no esquema de um DW costumam ser complexas e custosas, pois pequenas alterações se refletem em muitas operações de criação, exclusão e alteração de um grande volume de instâncias. Isso sem mencionar os vários problemas operacionais e semânticos que uma mudança do esquema pode gerar, tais como:

- prejudicar a disponibilidade do DW, pois muito tempo é gasto no ajuste das instâncias para o novo esquema, devido ao alto grau de dependência entre os dados (principalmente o relacionamento das métricas com as dimensões) e o elevado volume de registros e índices reprocessados;
- necessitar de ajuste dos aplicativos, com alteração e compilação dos programas, reprocessamento dos cubos dos *Data Marts* que possuem seus cubos formados a partir dos dados do DW;
- causar perdas ou inconsistências nos dados, ocasionadas por erros na estratégia de migração dos dados para o novo esquema;

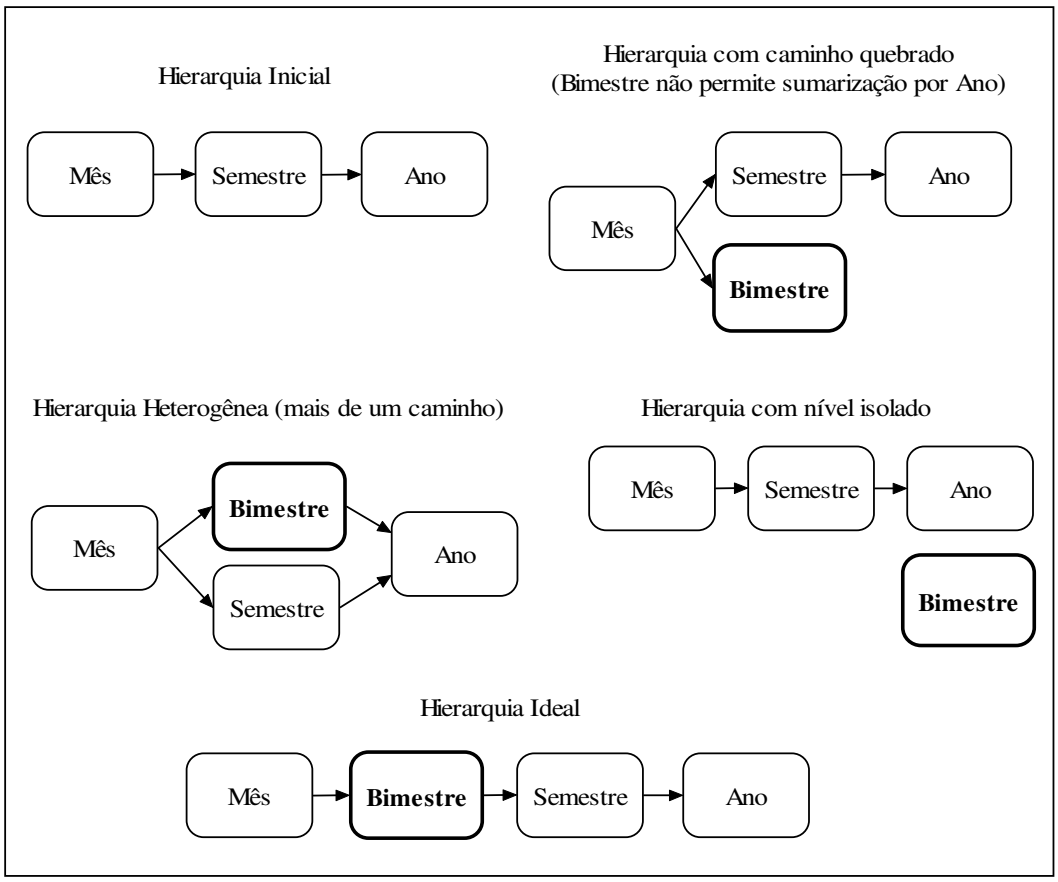

Figura 3.1: Erros na inclusão de um nível de dimensão

- gerar estruturas desconexas ou sem sentido semântico, como por exemplo: níveis de dimensões isolados, caminhos de classificação quebrados, hierarquias heterogêneas, funções de agregação que não fazem sentido (soma de porcentagens), etc. A figura 3.1 ilustra alguns erros que podem ocorrer na inclusão de um novo nível na dimensão tempo.

3.2 Migração dos Dados para Nova Versão do Esquema

A parte mais difícil e factível de erros na mudança do esquema lógico é, sem dúvida, a transformação dos dados armazenados do esquema antigo para o novo esquema. Os SGBDs e linguagens de BD geralmente não possuem suporte adequado para a adaptação das instâncias de um DW, sendo necessária a execução de *scripts* de manutenção, pelo administrador do DW, antes e depois da alteração das definições. Para ilustrar melhor o problema de mudança do esquema são apresentados dois exemplos de alteração que necessitam da geração de *scripts* de adaptação das instâncias:

- um nível de dimensão precisa ter um atributo desmembrado em dois ou mais atributos com tipos de dados (domínios) diferentes. Como exemplo um campo texto de endereço que precisa ser estruturado em um campo numérico para o número, um campo texto para o nome da rua, um campo numérico para o CEP, etc . Para realizar esta simples modificação estrutural é preciso excluir o campo de endereço e criar os novos campos: número, rua, CEP, etc. Mas, para evitar a perda dos dados de endereço é preciso executar um *script* antes da alteração do esquema, para guardar os dados em um arquivo e depois um *script* para o preenchimento correto dos novos campos de endereço a partir deste arquivo;
- a granularidade de uma métrica precisa ser alterada para um nível mais alto da hierarquia de dimensões. Como exemplo, a necessidade de armazenar somente valores mensais em vez de valores diários. Para realizar esta alteração é preciso mudar o nível base da métrica que define a granularidade mínima da dimensão tempo (alteração do esquema) e ajustar todos os elementos da matriz de valores básicos da métrica (alteração das instâncias). Para tanto é preciso executar um *script* para guardar todos os valores da matriz de valores em um arquivo, executar a alteração da granularidade da métrica (mudança da definição do nível base para a dimensão tempo) e executar um outro *script* para gerar a nova matriz de valores a partir dos dados do arquivo auxiliar.

Com estes dois exemplos já fica evidente que o administrador do DW possui um papel importantíssimo na evolução do esquema do DW, pois sem a correta criação dos *scripts* de adaptação das instâncias, muitos dados seriam perdidos ou ficariam inconsistentes depois da mudança do esquema.

Outra característica interessante, apresentada nestes dois exemplos, é que algumas mudanças no esquema não precisariam, a rigor, da intervenção do administrador. A modificação da granularidade para níveis mais altos, por exemplo, poderia automaticamente disparar um *script* genérico para a criação da nova matriz de valores base da métrica. Já o desmembramento do endereço necessita de um algoritmo particular, baseado no conhecimento do administrador em relação à formação dos valores do atributo endereço.

Vale relembrar que pequenas alterações nos esquemas do DW costumam afetar uma grande quantidade de instâncias, de forma que o tempo de atualização total do esquema e dos dados pode tornar o DW indisponível por um longo intervalo de tempo.

Um grande problema que costuma ocorrer em um DW, é a obtenção de valores do passado, quando é diminuída a granularidade de um métrica. O DW precisa ser carregado com os valores do novo nível de detalhe que servirão de base para a formação dos valores dos agregados, mas nem sempre estes dados existem para o passado. Nestes casos, geralmente são aplicados dois tipos de soluções:

- utilização de uma função de conversão de valores, baseada no conhecimento do negócio ou nos dados do presente. Exemplo: uma vez mudada a granularidade mínima da dimensão tempo de mês para dia para a métrica número de produtos fabricados, e visto a inexistência de dados passados ao nível de dia, utiliza-se uma função que divide igualmente o valor total das vendas no mês (que está armazenada no DW) para cada dia do mês que não for sábado, domingo ou feriado;
- versionamento da métrica ou de todo o DW. Os valores da métrica são guardados em dois repositórios com esquemas diferentes e válidos para intervalos de tempo diferentes. Para se consultar uma métrica num intervalo de tempo que abranja duas ou mais versões é preciso gerar uma consulta em cada repositório e juntar as respostas manualmente. Esta solução costuma ser executada quando ocorrem grandes modificações de uma única vez no esquema do DW.

3.3 Histórico da Evolução (Versões dos Esquemas e Dados)

O principal objetivo de um DW é facilitar o entendimento da evolução de um certo negócio, ou seja, ele é construído para ajudar o analista a compreender as relações entre as variáveis do negócio (dimensões e seus atributos) na composição dos resultados do negócio (métricas), a fim de descobrir problemas com antecedência e tomar as medidas certas para melhorar os resultados futuros.

A metodologia de análise proporcionada pelo DW é baseada na idéia do acompanhamento de poucas variáveis (dimensões) por vez, através da fixação das outras variáveis (dimensões), de modo que o analista possa descobrir quais são as dimensões e membros de dimensão que são relevantes para a composição dos resultados obtidos e por que eles são relevantes.

Esta forma de análise é perfeita quando não existem mudanças no esquema, nem nos membros das dimensões durante o intervalo de tempo pesquisado. Todavia, mudanças ocorrem, e a melhor forma de minimizar os efeitos das mudanças nas consultas dos usuários do DW é possibilitar a consulta das modificações realizadas, através do armazenamento do histórico de modificação dos esquemas e dos dados .

Para deixar bem claro o problema apresentado, tome como exemplo um DW de uma empresa que possua cinco filiais, cada uma responsável por uma região do país. O DW de vendas possui uma métrica para análise das vendas em cada cidade, sendo cada filial responsável por um conjunto distinto de cidades. Em um determinado dia, uma das filiais é fechada e as cidades desta filial passam a fazer parte de outras filiais. Agora imagine um usuário pesquisando o faturamento mensal por filiais após a exclusão da filial "Nordeste". Todas as filiais que receberam cidades da filial "Nordeste" estarão mostrando valores errados (maiores) para o tempo anterior a exclusão da filial

"Nordeste", pois não há indicação no DW de quando as cidades mudaram de filial, para este DW "Nordeste" nunca existiu e as cidades nunca mudaram de filial.

Para resolver este problema, o mais correto seria desvincular o relacionamento estático de pai-filho de filial e cidade, criando duas dimensões distintas: uma para cidade e outra para filial, pois na teoria, todas as dimensões de um DW devem ser ortogonais e estáticas. Na prática isto geraria uma explosão do número de dimensões, prejudicando a manutenção do DW e as consultas dos usuários finais, de forma que a criação de novas dimensões devem ser muito bem analisadas, levando-se em conta o que realmente o analista precisa ter.

De qualquer modo, é muito interessante armazenar o histórico dos dados e modificações do esquema, permitindo que seja consultada e entendida todas as mudanças ocorridas no DW.

3.4 Implementações e Modelos de DW

Inicialmente *Data Warehouses* eram implementados em banco de dados relacionais genéricos, sendo as regras semânticas da modelagem dimensional controladas pelas aplicações. Mas, devido ao crescimento do mercado de DW, todos os fornecedores de SGBDs passaram a incorporar novos recursos e extensões aos seus produtos para suportarem os modelos de dados dos DWs. Hoje existem muitas soluções disponíveis para a implementação dos DWs. (Pereira, 1999; Machado, 2000; Barbieri 2001)

Atualmente a maioria destas soluções são implementadas como uma camada adicional sobre de dois tipos básicos de banco de dados: relacionais e orientados a objetos. Os BDs relacionais de um modo geral possuem um suporte muito pobre à evolução do esquema, principalmente no que se refere a adaptação das instâncias. Estes BDs somente se preocupam com a integridade do modelo relacional, deixando a parte de correção das instâncias sob responsabilidade do DBA (*Data Base Administrator*) (Blaschka, 2000).

Os BDs orientados a objetos, por sua vez, são por definição muito mais ricos semanticamente, permitindo a redefinição de classes e a adaptação automática de suas instâncias segundo critérios de consistência bem definidos. Os BDs orientados à objetos são geralmente implementados como uma camada adicional no topo de um BD relacional, sendo então chamados de objeto-relacionais. (Blaschka, 2000)

A partir da década de 90, muitos modelos de DW foram propostos para a modelagem dimensional, tais como:

- Modelo Estrela de Kimball (1996,1998);
- NMDM (*Nested Multidimensional Data Model*) de Lehner (1998);
- MD de Cabibbo e Torloni (1998);
- DFM (*Dimensional Fact Model*) de Golafarelli, Maio e Rizzi (1998);
- MER (*Multidimensional Entity Relationship*) de Sapia et al. (1999);
- *StarER* de Tryfona, Bushorg e Christiansen (1999);
- *Object-Relation View* de Gopalkrishnan et al. (1999);
- GOLD de Trujillo, Palomar e Gómez (2000);
- Modelo multidimensional OO de Nguyen, Tjoa e Wagner (2000);

- Metodologia para modelagem de DW de Hüsemann et al. (2000);
- Metodologia de Moody e Kortink (2000);
- Modelo de dados GMD de Franconi e Kamble (2003).

Cada um destes modelos possui uma forma diferente de armazenar e recuperar as informações sob uma visão multidimensional, mas poucos deles se preocupam efetivamente com o armazenamento e o controle do histórico dos dados e dos esquemas do DW.

3.5 Modelos com Controle de Evolução dos Dados

Devido à importância da manutenção dos históricos e da correta adaptação das instâncias na alteração das dimensões, vários trabalhos foram propostos nesta área. A seguir é apresentado um resumo de alguns trabalhos significativos na área de evolução de dados de DW.

3.5.1 Modelos de Kimball para Evolução dos Dados

Kimball (1996) foi um dos primeiros autores a tratar o problema da alteração dos dados das dimensões em relação ao tempo, mais precisamente a alteração dos atributos descritivos dos membros. Ele sugeriu três tipos de soluções para manter a consistência do DW :

1) substituir os valores antigos dos registros dos membros e, portanto, perder a capacidade de rastrear o histórico passado;
2) dividir a chave única dos membros em duas partes, uma para armazenar o identificador único e outra para armazenar a versão do membro;
3) utilizar três campos para cada atributo do membro, um para armazenar o valor atual, outro para o valor original e um terceiro para guardar a data da última modificação.

A primeira alternativa, chamada de "*Update Model*" (Abelló; Martin, 2003; Body et al. 2003), se preocupa somente com a correta manutenção da última versão dos membros das dimensões, garantindo somente a consistência dos dados na versão atual das dimensões. A segunda alternativa trata cada versão diferente de um mesmo membro como membros distintos e independentes. Na terceira alternativa, as evoluções são mantidas dentro dos membros, mas somente a primeira e a última versão dos dados de um membro são mantidas.

Em (Kimball et al., 1998) são sugeridas três diferentes estruturas de armazenamento destinadas ao entendimento das evoluções dos dados do DW:

- *Transaction Model* – baseado na manutenção de *logs* de transações do DW;
- *Delivery Status Model* – utilização de uma tupla (registro) para representar cada versão de dados de cada membro de dimensão do DW;
- *Inventory Snapshot Model* – utilização de uma tupla para cada membro de dimensão, contendo as várias versões do membro associadas a instantes de tempo (*timestamps*).

3.5.2 O *Business Data Warehouse* de Devlin

Devlin (1997) distingue dois tipos de dados de acordo com o seu comportamento temporal dentro dos BDs:

- dados transientes, que são destruídos no caso de alterações ou exclusões;
- dados periódicos, que são sempre adicionados e nunca excluídos.

Segundo ele, todos os dados do DW deveriam ser periódicos e não somente os valores das métricas. Como a maioria dos DW armazenam os dados das dimensões como dados transientes, ele utiliza a expressão de *Business Data Warehouse* (BDW) para indicar que um DW armazena todas as relações temporais entre seus dados. O BDW proposto por Devlin armazena três *timestamps* (instantes de tempo) para cada registro do DW: dois para guardar o período de validade do registro (*start time* e *end time*) e um terceiro (i*nitial Time*) para indicar o instante de início de existência de um objeto de negócio (registro). Este terceiro instante serve para identificar a reutilização de uma chave operacional, uma vez que os BDs operacionais permitem reaproveitar valores de chaves. Os *timestamps* são incluídos ao nível de registro na proposta de Devlin para melhorar o desempenho, sendo criados novos registros toda vez que um ou mais campos são modificados.

3.5.3 Modelo de Consistência Temporal para DW de Bruckner et al.

Brukner, List, Schiefer e Tjoa (2002) destacam a necessidade de caracterização temporal dos dados do DW. Eles defendem o armazenamento de três aspectos temporais dos dados:

- o tempo de validade, que indica o período de validade do dado no mundo real;
- o tempo de revelação, que indica o momento em que os dados foram extraídos dos sistemas fontes;
- o tempo de carga, que indica o momento em que os dados foram integrados ao DW.

Um modelo conceitual de consistência temporal baseado nas relações temporais desses três aspectos é utilizado para enriquecer a qualidade dos dados, permitindo a representação histórica e a análise das mudanças dos dados.

3.5.4 O *Dimensional Update Model* de Hurtado, Mendelzon e Vaisman

Hurtado, Mendelzon e Vaisman (1999, 2000, 2002) realizaram diversos estudos sobre o comportamento temporal dos dados das dimensões, principalmente sobre as mudanças nas hierarquias dos membros. Eles definiram cinco operadores para alteração das definições das dimensões e dois operadores para alteração dos membros das dimensões (Hurtado; Mendelzon; Vaisman, 1999):

- *generalize* – adição de um novo nível superior;
- *specialize* – adição de um novo nível inferior;
- *relate level* – adição de um nova relação de pai-filho entre dois níveis;
- *unrelate level* – exclusão de uma relação de pai-filho entre dois níveis;
- *delete level* – exclusão de um nível;
- *add instance* – inclusão de um novo membro de dimensão;
- *delete instance* – exclusão de um membro de dimensão.

Além desses operadores, chamados de primitivos, foram definidos quatro operadores complexos para a execução de alterações nos membros de dimensão:

- *Reclassify* (reclassificação) – mudança dos membros pais (superiores) de um membro de dimensão;
- *Split* (divisão) – divisão de um membros em dois novos membro;
- *Merge* (união) – união de dois membros em um novo membro;
- *Update* (alteração) – alteração dos atributos descritivos dos membros.

Para cada um destes operadores complexos são apresentadas regras pré-condicionais de integridade e algoritmos que indicam o tipo, número e ordem de execução dos operadores primitivos para a correta realização da operação complexa. Em resumo, é apresentado um modelo conceitual para o controle das alterações das dimensões (*Dimensional Update Model*) que assegura a consistência semântica dos dados e das estruturas multidimensionais durante as alterações.

3.5.5 O Modelo Temporal de DW de Eder e Koncilia

Eder e Koncilia (2001, 2002 e 2003) definem um T*emporal Data Warehouse* que armazena todas as versões da estrutura hierárquica das dimensões, permitindo consultas sobre qualquer uma das versões das dimensões. Neste modelo, os níveis das dimensões não são fixos, nem modelados previamente, sendo deduzidos das hierarquias dos membros. Isto significa que a criação ou exclusão de novos níveis não é tratado como uma mudança de esquema e sim como uma alteração das relações de pai-filho entre os membros. Todos os membros, atributos e relacionamentos de pai-filho possuem datas de início e fim de validade.

Para realizar a consulta dos dados sob a visão de uma determina versão, são utilizadas matrizes de conversão que guardam as funções de mapeamento de cada membro de uma versão para cada membro de sua versão sucessora. Com a utilização destas matrizes, mais as matrizes de valores das métricas de cada versão, são feitas sucessivas operações de multiplicação matricial para o cálculo dos valores correspondentes a uma determinada versão temporal do DW.

Em resumo, é provido um modelo matemático para o controle das operações de alteração das constituição das dimensões (inclusão, exclusão, união, divisão ou mudança de nível dos membros) via operações matriciais. Neste modelo, não existe redundância das métricas, pois elas são adaptadas a esquemas e hierarquias passadas ou futuras durante a execução da consulta. Por outro lado, para cada versão das dimensões são criadas enormes matrizes esparsas formadas por N x M elementos, sendo N o número de membros da versão atual e M o número de membros da versão sucessora. A manipulação destas matrizes pode causar sérios problemas de performance numa implementação prática deste modelo.

3.5.6 O Modelo Temporal de DW de Body et al.

Body et al. (2002) propõem um modelo conceitual para permitir a comparação de dados de diferentes versões das dimensões. O modelo é construído sobre esquemas multidimensionais temporais, onde cada alteração nos membros das dimensões e seus relacionamentos são devidamente armazenados e associados a intervalos de validade. As versões das dimensões são inferidas pelas interseções de intervalos de validade das versões dos membros e seus relacionamentos. Uma tabela fato multiversionada é utilizada para guardar as métricas associadas as diversas versões das dimensões.

Cada versão da composição dos membros das dimensões (intervalo de tempo em que não houve modificações nos membros) forma um modo temporal de apresentação. O modelo permite a visualização dos dados do DW em qualquer um dos modos temporais de apresentação, através do armazenamento de funções transitivas de mapeamento entre duas versões consecutivas. Também é suportado um modo de apresentação especial denominado de "tempo consistente" que considera as hierarquias e atributos dos membros válidos no momento da validade da métrica. No modo de tempo consistente não existe mapeamento entre versões, sendo mostrado todas as versões de um membro como sendo membros diferentes e independentes.

As evoluções dos membros são compostas por cinco operações simples:

- criação de um membro;
- exclusão de um membro;
- transformação de um membro (mudança de um atributo ou de seu nome);
- união de n membros dentro de um membro;
- divisão de um membro em n membros;
- reclassificação do membro (mudança do nível).

O modelo é implementado sob uma arquitetura dividida em três partes:

- um DW temporal que guarda o histórico dos membros e as funções de mapeamento entre versões dos membros sem nenhuma redundância;
- um DW multiversionado que armazena os modos de apresentação (versões das dimensões, e o tempo consistente) como uma dimensão a parte e utiliza uma grande tabela fato multiversionada que contém os valores das métricas duplicadas para cada versão, gerando um alto nível de redundância;
- e um cubo OLAP construído a partir do DW multiversionado, contendo os dados agregados e permitindo a visualização dos dados em qualquer um dos modos de apresentação, de acordo com a escolha do usuário.

Para auxiliar o usuário na escolha da melhor versão de análise, são atribuídos fatores de confiabilidade para cada valor de métrica (dado fonte, valor exato, valor aproximado, relação desconhecida) que podem ser calculados, gerando um valor global de qualidade de cada versão (confiabilidade dos dados) e um *Ranking* das melhores versões para a pesquisa realizada.

3.5.7 O Bitemporal DW de Abelló e Martín

Abelló e Martín (2003) aplicaram conceitos de bancos de dados temporais para definirem um DW bitemporal. Neste DW, os valores dos atributos são armazenados juntamente com seus respectivos tempos de transação e validade. O tempo de transação indica o tempo em que o dado foi inserido no BD e o tempo de validade o tempo em que o dado retrata a realidade (Jensen; DYRESON, 1998). Segundo Abelló, o gerenciamento dos aspectos temporais dos dados é imprescindível para qualquer sistema de tomada de decisão. Um modelo temporal orientado a objetos é proposto para controlar os tempos de transação e validade dos objetos, segundo regras de integridade temporal. Os objetos do modelo possuem um *lifespan* (tempo de vida) que é formado por um conjunto contínuo de intervalos, indicando as diversas classificações e valores de um objeto.

A representação dos objetos é baseada no *Inventory Snapshot Model* de (Kimball et al., 1998) utilizando-se uma tupla formada por n elementos, sendo cada elemento uma 3-upla contendo uma valor para o atributo, um intervalo de tempo de validade e um intervalo de tempo de transação para um determinado valor. Para a atribuição das datas de início e fim dos intervalos e definição da semântica temporal dos dados são estudadas quatro tipos diferentes de fontes de aquisição de dados para população do DW (Martin; Abelló, 2003):

- instantâneos periódicos;
- arquivos de *log* que armazenam as transações do BD ou de arquivos diferenciais (*delta files*);
- utilização de gatilhos em fontes cooperativas que gravem as alterações em tabelas auxiliares (*delta tables*);
- banco de dados bitemporais.

Para os autores, o DW não deve ser implementado em um SGBD multidimensional, sendo as estruturas multidimensionais utilizadas somente em *Data Marts* carregados com uma única versão temporal dos dados do DW bitemporal. O DW bitemporal serve, portanto, como um fonte integrada de dados para a formação de DMs, nos quais são feitas as consultas OLAP.

Como em qualquer DW, foi preciso tomar alguns cuidados na hora da implementação do modelo, para não prejudicar demais a performance do DW. Foram criadas duas tabelas diferentes para armazenar os valores dos atributos, uma para armazenar somente os valores correntes e outra para armazenar todo o histórico. Na tabela de valores correntes, cada linha representa um objeto do DW, contendo uma coluna para o OID, uma coluna para atributo do objeto e duas colunas para representarem as datas de início de transação e de início de validade. A tabela de históricos por sua vez, representa um valor de atributo por linha, contendo uma coluna para o OID, outra para a identificação do atributo e quatro para as datas de início e fim de transação, e início e fim de validade.

Utilizando estas duas tabelas, os autores conseguem diminuir o número de junções necessárias para consultas aos valores correntes do DW, melhorando o tempo de resposta e ao mesmo tempo otimizar o gasto em espaço de armazenamento, diminuindo a redundância no armazenamento dos valores históricos.

Abelló e Martín tratam somente as evoluções nos valores dos atributos dos membros das dimensões, não sendo explicado como deve ser o armazenamento dos tempos de validade e de transação dos membros, dos relacionamentos de pai-filho entre membros e dos valores das métricas.

3.6 Modelos com Controle da Evolução dos Esquemas

A maioria dos modelos de evolução de DW tratam apenas as modificações nas instâncias do DW, havendo um número menor de trabalhos na área de evolução de esquemas de DW. A seguir são apresentados alguns trabalhos significativos na área de evolução de esquemas de DW.

3.6.1 Modelo para evolução de esquema de Blaschka et al.

Blaschka (2000) apresenta um modelo para o controle da evolução de esquemas multidimensionais baseado no ME/R (*Multidimensional Entity Relationship*) de (Sapia

et al., 1999). Ele faz um levantamento completo das operações de evolução do ME/R (quatorze operações possíveis), das relações de dependência entre as operações e das restrições de integridade de cada operação, criando uma álgebra para evolução de esquemas multidimensionais. São também apresentados algoritmos para o ordenamento das operações de alteração do esquema e para a correta migração automática das instâncias pré-existentes do DW para o novo esquema, garantindo a integridade semântica dos dados.

Além do modelo conceitual, são apresentados mapeamentos das operações evolutivas do modelo multidimensional para uma implementação relacional do ME/R. Todo um ambiente para a manipulação das alterações foi implementado, contendo visualização gráfica do esquemas ME/R (Hahn; Sapia; Blaschka, 2000) e migração automáticas das instâncias.

3.6.2 Modelo e Linguagem de Consulta Temporal de Vaisman

Em (Vaisman, 2001; Vaisman; Mendelzon, 2002) é proposto um modelo multidimensional temporal para o gerenciamento das várias versões dos dados e esquemas das dimensões, focando principalmente das relações hierárquicas entre os membros. O modelo de Vaisman utiliza os mesmos conceitos do *Dimensional Update Model* (Hurtado, Mendelzon e Vaisman 1999, 2000, 2002) para o tratamento das alterações agregando a possibilidade de consultas temporais sobre os dados através de uma linguagem própria, denominada de TOLAP.

Neste modelo são atribuídos intervalos de validade para os membros das dimensões e seus relacionamentos de pai-filho. Esses intervalos retratam o tempo em que os membros existiram no mundo real. As tabelas fato por natureza já possuem seu tempo de validade representado pela dimensão tempo, entretanto é adicionado mais um atributo temporal de tempo de transação para cada tabela fato, indicando qual a versão de dimensões é válida para os registros da tabela. Este atributo é armazenado automaticamente pelo BD toda vez que ocorre alguma mudança no esquema das dimensões, indicando portanto o momento da criação de uma nova versão do esquema do DW.

Uma linguagem específica de consulta foi construída para este modelo, combinando as características de linguagens temporais (TSQL2, SQL/TP) e o padrão das consultas OLAP. Esta linguagem, chamada de *Temporal OLAP* ou simplesmente TOLAP, possui tradução de todos os seus elementos e construções para o SQL (Vaisman; Mendelzon, 2002). Contudo, pequenos trechos de código de consultas em TOLAP costumam gerar grande quantidade de código não otimizado em SQL.

Para a implementação do modelo temporal de DW são sugeridos dois tipos de estruturas, uma otimizada para esquemas de dimensões fixos (onde não há mudanças de esquema, somente de dados) e outra para esquemas variáveis. A principal diferença entre os tipos, é que no modelo de esquema fixo cada relação de pai-filho entre membros é mapeada em um registro, enquanto que no esquema variável todas as relações são mapeadas em um único registro.

O modelo apresentado permite, através da linguagem TOLAP, a realização de consultas sobre o histórico dos dados, bem como a visualização no padrão OLAP dos dados válidos de uma versão do esquema, ou de todos os dados passados na visão da última versão do esquema.

3.7 Comparação dos Modelos de Evolução de DW

Apesar de cada modelo ou metodologia de controle das evoluções do DW terem suas peculiaridades, eles podem ser divididos em dois grandes grupos:

- modelos para controle das alterações;
- e modelos para consultas temporais.

Os modelos para controle das alterações, chamados na literatura de "*Update Models*" têm como principal objetivo prover um mecanismo eficiente e seguro de manutenção das instâncias do DW nas modificações dos dados ou dos esquemas. Isto é, sua principal preocupação é garantir que as alterações não afetem a consistência semântica dos dados. Os trabalhos de Blaschka (2000), Kimball (1996) e Hurtado (1999, 2000, 2002), são exemplos de modelos para controle de alteração.

Os DWs temporais, por outro lado, têm como objetivo permitir consultas sobre diferentes versões da organização dos dados (estrutura hierárquica das dimensões e suas relações com as métricas) e entendimento das modificações realizadas ao longo do tempo. Os trabalhos de Abelló e Martín (2003), Vaisman et al. (2001, 2002), Body et al. (2002, 2003) e Eder e Koncilia (2001, 2002 e 2003) são bons exemplos deste tipo de modelo.

Os modelos temporais, de uma forma geral, não suportam operações de evolução do esquema ou suportam somente algumas operações. O modelo de Vaisman explica bem como são realizadas as alterações nos membros das dimensões e permite algumas operações de evolução de esquema sobre as dimensões (criação e exclusão de novos níveis e caminhos de classificação). Entretanto, não trata de mudanças nas definições das métricas, nem explica em detalhe como são executadas as adaptações das instâncias nas operações de alteração das definições das dimensões.

O modelo de controle de alteração de Blaschka (2000) é um dos poucos modelos que trata todas as operações de evolução do esquema (dimensões e métricas). Todavia, ele é destinado somente ao processo de mudança de esquema, não armazenando o histórico dos dados e alterações.

O desempenho das consultas realizadas nos modelos temporais também é um ponto importante de ser analisado. Apesar de ser difícil fazer uma comparação exata da performance de cada modelo, é possível fazer algumas deduções com base em características como:

- número de junções necessárias para um consulta;
- existência de dados pré-calculados;
- natureza das funções de conversão entre versões.

O modelo de Eder e Koncilia, sempre armazena os dados de acordo com sua estrutura no momento da carga dos dados, de forma que para qualquer consulta OLAP (seja ela sobre uma versão do passado ou sobre a versão atual) é preciso realizar uma série de cálculos matriciais em tempo de execução para a conversão das métricas para a versão hierárquica dos membros indicada pela consulta.

O modelo de Vaisman, por outro lado, não precisa realizar funções de conversão entre as versões das dimensões em tempo de execução, pois os valores das métricas para cada versão são previamente armazenados (calculados no momento da evolução dos dados e não nas consultas). Entretanto, existe uma perda de performance das consultas devido as muitas junções necessárias para a determinação das versões das dimensões

utilizadas no intervalo de tempo consultado (tarefa feita em tempo de execução da consulta).

O modelo de Abelló e Martín (2003) armazena os dados correntes em tabelas separadas e otimizadas para consultas sobre a versão atual dos dados do DW, que são as consultas mais freqüentes.

O modelo de Body et al. (2002), armazena todos os dados pré-calculados (versões das dimensões e valores das métricas) em um enorme cubo muldimensional, onde umas das dimensões representa a versão das dimensões. O desempenho é otimizado ao máximo, porém o espaço de armazenamento necessário para guardar este cubo se torna inviável com o aumento do número de versões diferentes das dimensões.

Na tabela 3.1 é apresentado um quadro comparativo dos principais modelos, incluindo a proposta deste trabalho. Os itens avaliados são os seguintes:

1. armazenamento do histórico de hierarquias dos membros de dimensão, guardando os intervalos de validade dos membros e dos relacionamentos de pai-filho entre membros;
2. armazenamento do histórico de valores dos atributos dos membros, guardando os intervalos de validade de todos os valores dos atributos dos membros;
3. consultas das métricas em qualquer versão dos membros. Isto é, consultas aos valores das métricas utilizando uma hierarquia de dimensões fixa. Isto implica em guardar o histórico das operações de divisão, união e mudança de nível dos membros, bem como funções de conversão das métricas entre membros de versões de hierarquias diferentes;
4. consulta das métricas na versão de tempo consistente. Ou seja, apresentar os valores de cada métrica de acordo com a hierarquia de dimensões válida para a validade da métrica sem utilizar, portanto, uma hierarquia de dimensões fixa;
5. alto desempenho nas consultas sobre qualquer versão dos dados, conseguido através da replicação ou pré-cálculo dos dados;
6. alteração do esquema das dimensões, isto é, permitir alterações nas definições dos níveis, caminhos de classificação e atributos dos níveis, além da inclusão ou exclusão de dimensões;
7. alteração do esquema das métricas, isto é, permitir alterações nos domínios das métricas e nos níveis dimensionais base, além da inclusão ou exclusão de métricas;
8. armazenamento do histórico de evolução do esquema;
9. alto desempenho sobre consultas a qualquer versão do esquema. Neste caso, significa possuir uma separação dos dados por versão do esquema, que pode ser feita via utilização de tabelas dimensão e fato diferenciadas para cada esquema ou utilização de repositórios independentes para cada esquema;
10. espaço de armazenamento reduzido, que significa basicamente o grau de replicação dos dados;
11. possibilidade de múltiplas versões do esquema ativas simultaneamente. Significa que as cargas de novos dados são feitas sobre mais de uma versão do esquema, possibilitando a consulta de dados atuais sob qualquer uma das versões dos esquemas ativos;
12. interface ou linguagem de consulta para dados históricos. Significa que o modelo permite consultas complexas sobre o histórico de mudanças dos dados ou esquemas através de construções simples e de alto nível.

Tabela 3.1: Comparação dos Modelos de Evolução

Características dos Modelos	**1**	**2**	**3**	**4**	**5**	**6**
1. Histórico das hierarquias dos membros de dimensão	S	S	S	S		**S**
2. Histórico dos valores dos atributos dos membros		S	S	S		**S**
3. Consulta das métricas em qualquer versão dos membros	S	S	S		N/A	
4. Consulta das métricas na versão de tempo consistente			S	S	N/A	**S**
5. Alto desempenho para consultas sobre qualquer versão dos dados		S			N/A	
6. Alteração do esquema das dimensões			P	S	S	**S**
7. Alteração do esquema das métricas				P	S	**S**
8. Histórico das alterações de esquema			P	P	N/A	**S**
9. Alto desempenho para consultas sobre qualquer versão do esquema	N/A	N/A		P	N/A	**S**
10. Espaço de armazenamento reduzido (pouca redundância de dados)	P		S	S	N/A	**P**
11. Mais de uma versão do esquema ativa simultaneamente	N/A	N/A			N/A	**S**
12. Interface ou linguagem de consulta de fácil manipulação dos dados históricos		S		S	N/A	

Legenda: 1. Eder e Koncilia 2. Body et al. 3. Abelló e Martín
4. Vaisman et al. 5. Blaschka et al. 6. Proposta deste trabalho
" " = não suporta S = suporta P = suporta parcialmente
N/A = não avaliado pois não armazena histórico

Para cada item avaliado, são atribuídos quatro valores possíveis:

- espaço em branco – indica que o item não é suportado pelo modelo;
- S – indica que o item é suportado;
- P – indica que o item é suportado parcialmente;
- N/A - como alguns modelos não armazenam o histórico dos dados ou não tratam alterações no esquema nenhum item relativo às versões de dados ou esquemas pode ser analisado.

Para os itens suportados parcialmente é feito um breve comentário a seguir, explicando por que o item não é satisfeito plenamente:

- Eder e Koncilia no modelo que propõem, apesar de não replicarem explicitamente os dados para cada versão hierárquica dos membros, utilizam extensas matrizes para representar as operações de união e divisão dos membros;
- o modelo de Abelló e Martín somente descreve operações de inclusão ou exclusão de atributos das dimensões, não sendo tratada a inclusão de novos níveis e dimensões ou mudanças na estrutura das métricas;
- o modelo de Vaisman et al. permite a alteração dos níveis base das métricas, mas não trata mudança nos domínios das métricas ou inclusão/exclusão de dimensões participantes de uma métrica;
- em relação ao custo de armazenamento, o modelo proposto neste trabalho mantém todas as versões dos dados de um mesmo esquema armazenadas de forma não redundante, mas possui replicação dos dados do passado entre as sucessivas versões do esquema.

3.8 Considerações Finais

Neste capítulo foram apresentados os pontos chaves da evolução do esquema e dos dados de um DW, destacando-se:

- tipo e freqüência das mudanças;
- causas da evolução;
- conseqüências da mudança do esquema nas instâncias pré-existentes;
- problemas que podem ser ocasionados na evolução do esquema;
- histórico dos dados e das modificações.

Também foram apresentados e comparados trabalhos significativos na área de evolução de dados e esquemas de DW. Muitas das idéias e dos conceitos destes trabalhos foram utilizados para construir a proposta deste trabalho.

4 OPERAÇÕES DE EVOLUÇÃO DE DATA WAREHOUSES

Este capítulo apresenta em detalhes as operações de evolução de um DW que foram divididas em dois grupos: operações de mudança do esquema e operações de mudança das instâncias (dados).

Como existe uma infinidade de operações de evolução possíveis, foi definido um conjunto reduzido de operações atômicas, tendo como base o modelo conceitual apresentado no capítulo dois. A composição ordenada destas operações atômicas permite a formação de funções complexas, cobrindo todos os tipos de modificações possíveis ao modelo de DW.

A operações atômicas de mudança de esquema foram baseadas no trabalho de Blaschka (2000) que apresentada uma descrição formal das operações de evolução de esquema para o modelo MER, bem como seus efeitos nas instâncias do DW.

Para a melhor compreensão das operações, foi criada uma linguagem de programação de alto nível, com uma sintaxe bem simples. Para a definição das operações são utilizadas a seguintes notações:

- palavras em maiúsculo - comandos fixos ;
- ; - término de uma operação;
- [..] - opcional;
- (..) - lista de itens separados por vírgula;
- | - escolha de um termo ou outro.

Para simplificar os comandos da linguagem de manipulação dos dados do DW, foi inserida uma regra de unicidade global nos nomes dos elementos, assim como é feito para o identificador único. Desta forma, não é permitida a criação de campos de atributos de dimensão ou de membros de dimensão com o mesmo nome, mesmo que eles pertençam a dimensões diferentes. Esta restrição pode ser retirada facilmente com a adição de cláusulas mais específicas na linguagem que detalhem melhor a identificação dos elementos. Todavia, isto não é necessário para este estudo, sendo útil somente em uma implementação real.

Para facilitar o entendimento de cada tipo de operação, são mostrados exemplos que ilustram como são montadas as operações e quais os elementos do modelo formal que são modificados.

4.1 Operações de Evolução do Esquema

São as operações que modificam as definições do *Data Warehouse*, incluindo, alterando ou excluindo métricas, dimensões e atributos. Estas operações são complexas e custosas, pois pequenas mudanças costumam alterar grandes quantidades de registros dos dados da extensão.

Ao final da execução de um conjunto de operações de evolução de esquema é sempre interessante realizar uma validação semântica do novo esquema gerado. Ou seja, apesar das operações não permitirem alterações inconsistentes devido às suas regras de integridade, é possível criar um esquema com erros semânticos. Isto é, um esquema que não faz sentido no mundo real.

4.1.1 Operações Atômicas de Evolução de Esquema

A seguir são definidas as operações atômicas de alteração de esquema, necessárias para o modelo de DW apresentado.

4.1.1.1 Criação de um Cubo

Descrição: Criação de um cubo para tratar novos fatos.

Sintaxe:
```
CREATE CUBO Nome_Cubo
            WITH METRICS (Nome_Metrica)
            PATHES(Membro_filho TO Membro_pai);
```

```
onde:   Nome_cubo       = nome do cubo;
        (Nome_Metrica)  = lista de métricas;
        (Membro_filho TO Membro_pai) = listas de caminhos
        de classificação das dimensões.
```

Restrições de Integridade:

1) unicidade do nome: não é permitida a criação do cubo caso já exista alguma outra entidade no DW com o mesmo nome;
2) existência das métricas informadas;
3) consistência das árvores hierárquicas formadas com os caminhos informados. Os caminhos devem formar árvores conexas e não podem possuir níveis inferiores à granularidade mínima das métricas (níveis base das métricas).

Exemplo: Criação do cubo Vendas com as métricas N° de itens vendidos e Valor das Vendas.

```
CREATE CUBO "Vendas"
    WITH METRICS "N° de itens vendidos",
                 "Valor da Venda"
    PATHES "Cidade" TO "Estado",
           "Estado" TO "⊥",
           "Data" TO "Mês",
           "Mês" TO "⊥",
           "Modelo do Produto" TO "⊥" ;
```

$CUBO_{vend}$ = <"vend", "Vendas", {M_{nro_vend}, M_{valor_vend}},
{$C_{cid->uf}$, $C_{uf->todos}$, $C_{dt->mes}$, $C_{mes->todos}$, $C_{pr->todos}$}>

4.1.1.2 Alteração de um Cubo

Descrição: Alteração do nome ou inclusão/exclusão de alguma métrica ou caminho de classificação do cubo.

Sintaxe:
```
UPDATE CUBO Nome_Cubo
    [TO Nome_Cubo]
    [INSERT/REMOVE METRICS (Nome_Metrica)]
    [INSERT/REMOVE PATHES (Membro_filho TO
    Membro_pai)] ;
```

onde: Nome_cubo = nome do cubo;
(Nome_Metrica) = lista de métricas;
(Membro_filho TO Membro_pai) = listas de caminhos de classificação das dimensões.

Restrições de Integridade:

1) unicidade do nome: não é permitida a alteração do nome caso já exista alguma outra entidade no DW com o mesmo nome;
2) existência das métricas informadas;
3) as novas métrica ou novos caminhos incluídos devem ser consistentes. Não pode ser inserido um caminho que não for comum a todas as métricas.

Exemplo: Mudança do nome do cubo Vendas para Total de Vendas, remoção da métrica N° de itens vendidos e remoção do nível Data do cubo.

```
UPDATE CUBO "Vendas" TO "Total de Vendas"
REMOVE METRICS "N° de itens vendidos"
REMOVE PATHES "Data" TO "Mês" ;
```

Antes: $CUBO_{vend}$ = <"vend", "Vendas", {M_{nro_vend}, M_{valor_vend}},
{$C_{cid->uf}$, $C_{uf->todos}$, $C_{dt->mes}$, $C_{mes->todos}$, $C_{pr->todos}$}>
Depois: $CUBO_{vend}$ = <"vend", "Total de Vendas",
{M_{valor_vend}}, {$C_{cid->uf}$, $C_{uf->todos}$, $C_{mes->todos}$, $C_{pr->todos}$}>

4.1.1.3 Exclusão de um Cubo

Descrição: Exclusão de um cubo.

Sintaxe:
```
DELETE CUBO Nome_Cubo ;
```

onde: Nome_cubo = nome do cubo.

Restrições de Integridade:
1) existência do cubo.

Exemplo: Exclusão do cubo Vendas.

```
DELETE CUBO "Vendas" ;
```

4.1.1.4 Criação de uma Métrica

Descrição: Criação de uma nova métrica, definindo domínio de valores, função de agregação e conjunto de hierarquias de dimensões.

Sintaxe:
```
CREATE METRIC Nome_metrica
        AS Tipo_domínio
        WITH FUNCTION f_agreg
        BASE LEVELS (Nome_nivel);
```

```
onde:   Nome_Metrica  = nome da métrica;
        Tipo_domínio  = nome do domínio;
        f_agreg       = nome da função de agregação;
        (Nome_nivel)  = lista de níveis base de
                        diferentes dimensões
```

Restrições de Integridade:

1) unicidade do nome: não é permitida a criação caso já exista alguma outra entidade no DW com o mesmo nome;
2) existência do tipo/domínio informado;
3) existência da função de agregação informada;
4) existência dos níveis base informados, sendo cada nível de uma dimensão diferente.

Observação: os domínios e funções de agregação são fortemente dependentes da forma de implementação dos SGBDs e das ferramentas OLAP, motivo pelo qual não serão melhor detalhadas neste estudo.

Exemplo: Criação da métrica Valor da Venda.

```
CREATE METRIC "Valor da Venda"
     AS "Decimal Positivo"
     WITH FUNCTION "SOMA"
     BASE LEVELS "Data", "Cidade", "Modelo do produto";
```

M_{valor_vend} = <"valor_vend", "Valor da Venda", Decimal Positivo, SOMA, {"N_{dt}", "N_{cid}", "N_{pr}"}, MVB_{mb2} >

4.1.1.5 Alteração de um Métrica

Descrição: Alteração do nome ou do domínio ou da função de agregação ou dos níveis base da métrica.

Sintaxe:
```
UPDATE METRIC Nome_metrica
        [TO Nome_metrica]
        [AS  Tipo_domínio]
        [WITH FUNCTION f_agreg]
```

```
            [BASE LEVELS (Nome_nivel)] ;

onde:   Nome_Metrica     = nome da métrica ;
        Tipo_domínio     = nome do domínio;
        f_agreg          = nome da função de agregação;
        (Nome_nivel)     = lista de níveis base de
                           diferentes dimensões.
```

Restrições de Integridade:

1) unicidade do nome: não é permitida a alteração do nome caso já exista alguma outra entidade no DW com o mesmo nome;
2) existência do tipo/domínio informado;
3) existência da função de agregação informada;
4) existência dos níveis base informados, sendo cada nível de uma dimensão diferente.

Observações:

- a modificação do domínio da métrica afeta diretamente a matriz de valores, podendo torná-la inconsistente, necessitando portanto de uma função para conversão dos valores entre os dois domínios durante ou após a execução da operação de alteração da métrica;
- a mudança de um nível de granularidade invalida todas as entradas na matriz de valores. Caso for feita uma substituição por um nível superior, basta criar uma nova matriz de valores com os valores agregados da matriz anterior. Já no caso de se utilizar uma granularidade menor (nível mais detalhado), será preciso fazer uma carga completa dos novos valores.

Exemplos:

1) Alteração do nome, domínio e função da métrica Valor da Venda.

```
UPDATE METRIC "Valor da Venda"
       TO "Valor da Venda dos Produtos"
       AS "Real"
       WITH FUNCTION "MÉDIA" ;

Antes: M_valor_vend = <"valor_vend", "Valor da Venda",
       Decimal Positivo, SOMA, {"N_dt", "N_cid", "N_pr"}>

Depois: M_valor_vend = <"valor_vend", "Valor da Venda dos
        Produtos", Real,MÉDIA, {"N_dt", "N_cid", "N_pr"}>
```

2) Alteração da granularidade mínima da métrica Valor da Venda de data para mês.

```
UPDATE METRIC "Valor da Venda"
       BASE LEVELS "Mês", "Cidade",
                   "Modelo do produto";

Antes: M_valor_vend = <"valor_vend", "Valor da Venda",
                     Decimal Positivo, SOMA,
                     {"N_dt", "N_cid", "N_pr"},MVB_mb2 >
```

Depois: M_{valor_vend} = <"valor_vend", "Valor da Venda",
Decimal Positivo, SOMA,
{"N_{mes}", "N_{cid}", "N_{pr}"},MVB_{mb2} >

4.1.1.6 Exclusão de uma Métrica

Descrição: Exclusão de uma métrica.

Sintaxe: `DELETE METRIC Nome_metrica ;`

onde: Nome_metrica = nome da métrica.

Restrições de Integridade:
1) existência da métrica informada;
2) inexistência de cubos utilizando a métrica informada.

Exemplo: Exclusão da métrica Valor da Venda.

```
DELETE METRIC "Valor da Venda" ;
```

4.1.1.7 Criação de uma Dimensão

Descrição: Criação de uma nova dimensão.

Sintaxe: `CREATE DIMENSION Nome_dim ;`

onde: Nome_dim = nome da dimensão.

Restrições de Integridade:
1) unicidade do nome: não é permitida a criação caso já exista alguma outra entidade no DW com o mesmo nome.

Exemplo: Criação da dimensão Tempo.

```
CREATE DIMENSION "Tempo" ;
```

4.1.1.8 Alteração de uma Dimensão

Descrição: Alteração do nome de uma dimensão.

Sintaxe: UPDATE DIMENSION Nome_dim TO Nome_dim ;

onde: Nome_dim = nome da dimensão.

Restrições de Integridade:
1) unicidade do nome: não é permitida a alteração caso já exista alguma outra entidade no DW com o mesmo nome.

Exemplo: Alteração do nome da dimensão Produto.

```
UPDATE DIMENSION "Produto" TO "Modelo do Telefone" ;
```

4.1.1.9 Exclusão de uma Dimensão

Descrição: Exclusão de uma dimensão.

Sintaxe: `DELETE DIMENSION Nome_dim;`

```
onde:   Nome_dim = nome da dimensão.
```

Restrições de Integridade:

1) não é permitido excluir uma dimensão caso existir algum nível de dimensão para esta dimensão.

Exemplo: Exclusão da dimensão Local.

```
DELETE DIMENSION "Local" ;
```

4.1.1.10 Criação de um nível de dimensão

Descrição: Criação de um novo nível de dimensão.

Sintaxe:
```
CREATE LEVEL Nome_nivel
        TO DIMENSION Nome_dim ;
```

```
onde:   Nome_nivel  = nome do nível;
        Nome_dim    = nome da dimensão do nível.
```

Restrições de Integridade:

1) unicidade do nome: não é permitida a criação caso já exista alguma outra entidade no DW com o mesmo nome;
2) existência da dimensão informada.

Exemplo: Criação do nível Estado para a dimensão Local .

```
CREATE LEVEL "Estado" TO DIMENSION "Local" ;
```

4.1.1.11 Alteração do nome de um nível de dimensão

Descrição: Alteração do nome de um nível de dimensão.

Sintaxe: `UPDATE LEVEL Nome_nivel TO Nome_nivel ;`

```
onde:   Nome_nivel   = nome do nível.
```

Restrições de Integridade:

1) unicidade do nome: não é permitida a alteração caso já exista alguma outra entidade no DW com o mesmo nome.

Exemplo: Alteração do nome do nível Estado para Unidade Federativa.

```
UPDATE LEVEL "Estado" TO "Unidade Federativa" ;
```

4.1.1.12 Exclusão de um nível de dimensão

Descrição: Exclusão de um nível de dimensão.

Sintaxe: `DELETE LEVEL Nome_nivel ;`

```
onde:   Nome_nivel  = nome do nível.
```

Restrições de Integridade:

1) inexistência de atributos para o nível de dimensão;
2) inexistência de caminhos de classificação para o nível de dimensão;
3) inexistência de métricas que utilizem este nível como nível base;
4) inexistência de membros para este nível de dimensão.

Exemplo: Exclusão do nível Estado .

```
DELETE LEVEL "Estado" ;
```

4.1.1.13 Criação de um atributo de dimensão

Descrição: Criação de um novo atributo descritivo para um nível de dimensão.

Sintaxe:
```
CREATE ATRIBUTE Nome_atributo
        TO LEVEL Nome_nivel
        AS Tipo_dominio ;
```

```
onde:   Nome_atributo   = nome do atributo;
        Nome_nivel      = nome do nível de dimensão;
        Tipo_domínio    = nome de um domínio válido.
```

Restrições de Integridade:

1) unicidade do nome: não é permitida a criação caso já exista alguma outra entidade no DW com o mesmo nome;
2) existência do nível de dimensão informado;
3) existência do tipo/domínio informado.

Exemplo: Criação do atributo É feriado para o nível Data da dimensão Tempo.

```
CREATE ATRIBUTE "É feriado"
       TO LEVEL "Data"
       AS "Lógico(Sim/Não)";
```

4.1.1.14 Alteração de um atributo de dimensão

Descrição: Alteração do nome ou do domínio de um atributo descritivo.

Sintaxe: **UPDATE ATRIBUTE Nome_atributo**
TO Nome_atributo
AS Tipo_dominio;

onde: Nome_atributo = nome do atributo;
Tipo_domínio = nome do domínio.

Restrições de Integridade:

1) unicidade do nome: não é permitida a alteração do nome caso já exista alguma outra entidade no DW com o mesmo nome;
2) existência do tipo/domínio informado.

Observações

A modificação do domínio do atributo afeta diretamente os membros existentes do nível de dimensão, sendo necessário executar uma função de conversão para os dois domínios.

Exemplo: Alteração do nome e domínio do atributo É feriado do nível Data .

```
UPDATE ATRIBUTE "É feriado"
       TO "Tipo Dia"
       AS "caracter";
```

4.1.1.15 Exclusão de um atributo de nível de dimensão

Descrição: Exclusão de um atributo de nível de dimensão.

Sintaxe: **DELETE ATRIBUTE Nome_atributo;**

onde: Nome_atributo = nome do atributo.

Restrições de Integridade:

1) Existência do atributo informado.

Exemplo: Exclusão do atributo "É feriado" .

```
DELETE ATRIBUTE "É feriado" ;
```

4.1.1.16 Criação de um caminho de classificação

Descrição: Consiste em criar uma novo caminho de classificação, definindo os relacionamentos de pai-filho existentes entre os níveis de uma dimensão .

Sintaxe: **CREATE DIMENSION PATH Nome_nivel_pai**
TO Nome_nivel ;

onde: Nome_nivel_pai = nome do nível pai;
Nome_nivel = nome do nível filho.

Restrições de Integridade:

1) existência dos dois níveis informados;
2) inexistência de "loops" na árvore hierárquica da dimensão com a inclusão do novo caminho.

Exemplo: Criação de um caminho de Data para Mês para a dimensão Tempo;

```
CREATE DIMENSION PATH "Mês" TO "DATA" ;
```

4.1.1.17 Exclusão de um caminho de classificação

Descrição: Exclusão de um caminho de classificação.

Sintaxe:
```
DELETE DIMENSION PATH Nome_nivel_pai
        TO Nome_nivel;

onde:   Nome_nivel_pai  = nome do nível pai;
        Nome_nivel      = nome do nível filho .
```

Restrições de Integridade:

1) existência do caminho informado;
2) inexistência de cubos que utilizem este caminho.

Exemplo: Exclusão do caminho de Data para Mês da dimensão Tempo;

```
DELETE DIMENSION PATH "Mês" TO "Data" ;
```

4.1.2 Operações Complexas de Evolução de Esquema

Através da combinação ordenada das dezessete operações atômicas de evolução de esquema definidas é possível construir um *script* de operações, possibilitando a criação de um novos esquemas, seja a partir de um esquema pré-existente, ou de um esquema nulo.

Quando é feita a criação do novo esquema a partir de um esquema inicial com dados populados é imprescindível a execução de operações de evolução das instâncias, de modo que os dados populados possam ser transformados e aproveitados na nova versão de esquema criada. Essas operações são apresentadas a seguir.

4.2 Operações de Evolução das Instâncias

Em situações normais, as instâncias ou os dados de um DW geralmente não são alteradas nem excluídas, ocorrendo somente um aumento incremental e periódico de novos membros de dimensões e novos valores na matriz multidimensional de valores base.

Todavia, em alguns casos é preciso alterar ou excluir os valores das métricas e membros de dimensões. Com este objetivo foi definido um conjunto de sete operadores para tratamento das instâncias.

4.2.1 Operações Atômicas de Evolução das Instâncias

Lista de operações atômicas que incluem, excluem ou alteram as instâncias de um esquema de DW, que são definidas a seguir.

4.2.1.1 Criação de um Membro de Dimensão

Descrição: Criação de um novo membro em um nível de dimensão, com atribuição opcional de valores para seus atributos descritivos.

Sintaxe:
```
CREATE MEMBER Nome_Membro
    TO LEVEL Nome_nível
    [WITH ATTRIBUTES (Nome_campo = Valor)] ;
```

```
onde:   Nome_membro  = nome do membro;
        Nome_Nível   = nome do nível de dimensão;
        Nome_campo   = nome do atributo;
        Valor        = valor do atributo.
```

Restrições de Integridade:
1) unicidade do nome, não sendo permitida a criação do membro caso já exista alguma outra entidade do DW com o mesmo nome;
2) existência do nível de dimensão informado;
3) existência dos atributos de dimensão informados;
4) valores dos atributos informados dentro do domínio do atributo.

Observações

Os membros são criados sem conexão com a árvore hierárquica dos membros da dimensão, ou seja, não possuem membros pais associados. Existe uma operação específica para a determinação dos membros pais, que é demonstrada mais adiante.

A única excessão a esta regra é a inclusão de um membro em um nível que não possua níveis superiores por definição. Neste caso, é criado automaticamente sua ligação com o membro distinto Todos, presente em todas as árvores hierárquicas e representado pelo símbolo "⊥".

Exemplos:

1) Criação do membro RS para o nível Estado da dimensão Local:

```
CREATE MEMBER "RS"
    TO LEVEL "Estado" ;
```
Gerando: MD_{RS} = < "RS", "RS", N_{uf} , {⊥}, ∅ >

Obs.: como o nível Estado não possui nível superior, já é incluído o membro distinto "⊥" no conjunto de níveis pais do novo membro RS.

1) Criação do membro Canoas para o nível Cidade da dimensão Local:
```
CREATE MEMBER "Canoas"
    TO LEVEL "Cidade" ;
```
Gerando: MD_{cid1} = < "cid1" ,"Canoas", $N_{cid,}$ ∅, ∅ >

2) Criação do membro 03/02/2003 para o nível Data da dimensão Tempo, com os atributos descritivos É feriado = "não" e Dia da Semana = "Segunda":

```
CREATE MEMBER "03/02/2003"
    TO LEVEL "Data"
    WITH ATTRIBUTES "É feriado" = "Não",
                    "Dia da Semana" = "Segunda" ;
```

Gerando: $MD_{03/02/2003}$ = <"03/02/2003", "03/02/2003", N_{dt}, ∅, {"não", "Segunda"}>

4.2.1.2 Alteração de um Membro de Dimensão

Descrição: Modificação do nome, nível ou atributos de um membro de dimensão.

Sintaxe:
```
UPDATE MEMBER Nome_Membro
        [TO Nome_membro]
        [TO LEVEL Nome_nível]
        [WITH ATTRIBUTES (Nome_campo = Valor)];
```

onde:
- Nome_membro = nome do membro;
- Nome_Nível = nome de um nível de dimensão;
- Nome_campo = nome do atributo;
- Valor = valor do atributo.

Restrições de Integridade:
1) existência do membro com o nome informado;
2) existência do nível de dimensão informado;
3) existência dos atributos de dimensão informados no nível de dimensão do membro;
4) valores dos atributos informados dentro dos domínios dos atributos;
5) no caso de troca de nível o membro não pode possuir nenhum nível pai e nenhum nível filho associado.

Observações

A alteração do nível do membro implica na mudança dos tipos de atributos do membro, de forma que todos os valores antigos dos atributos são perdidos na troca de nível.

Exemplos:

1) Modificação do nome do membro RS.

```
UPDATE MEMBER "RS"
  TO "Rio Grande do Sul" ;
```

Antes: MD_{RS} = <"RS","RS", N_{uf} , {⊥}, ∅ >

Depois: MD_{RS} = <"RS","Rio Grande do Sul", N_{uf} , {⊥}, ∅ >

2) Modificação do nível do membro Viamão de Cidade para Estado .

```
UPDATE MEMBER "Viamão"
  TO LEVEL "Estado";
```

Antes: MD_{cid2} = <"Cid2","Viamão", N_{cid} , ∅, ∅ >
Depois: MD_{cid2} = <"Cid2","Viamão", N_{uf} , {⊥}, ∅ >

4.2.1.3 Exclusão de um Membro de Dimensão

Descrição: Exclusão de um membro de dimensão.

Sintaxe: **`DELETE MEMBER Nome_Membro ;`**

onde: Nome_membro = nome do membro.

Restrições de Integridade:
1) existência do membro com o nome informado;
2) o membro excluído não pode possuir nenhum membro pai e nenhum membro filho;
3) o membro excluído não pode fazer parte de nenhuma matriz de valores base (MVB).

Exemplos:

1) Exclusão do membro RS.
```
DELETE MEMBER "RS" ;
```

2) Exclusão do membro Viamão.
```
DELETE MEMBER "Viamão" ;
```

4.2.1.4 Inclusão de Relacionamento de Pai-Filho entre Membros

Descrição: Inclusão da associação de pai-filho entre dois membros de dimensão.

Sintaxe:
```
INSERT FATHER Nome_Membro_Pai
        TO Nome_Membro_Filho;
```

onde: Nome_Membro_Pai = nome do membro Pai;
Nome_Membro_Filho = nome do membro Filho.

Restrições de Integridade:
1) existência dos membros informados;
2) existência de caminho de classificação entre os níveis informados.

Exemplo:

Inclusão da associação do membro pai RS com o membro filho Viamão.

```
INSERT FATHER "RS" TO "Viamão" ;
```
Antes: MD_{cid2} = <"Cid2","Viamão", N_{cid} , ∅, ∅ >
Depois: MD_{cid2} = <"Cid2","Viamão", N_{cid} , {MD_{RS}}, ∅ >

4.2.1.5 Exclusão de Relacionamentos Pai-Filho entre Membros

Descrição: Exclusão da associação de pai-filho entre dois membros de dimensão.

Sintaxe: **REMOVE FATHER Nome_Membro_Pai**
TO Nome_Membro_Filho;

onde: Nome_Membro_Pai = nome do membro Pai;
Nome_Membro_Filho = nome do membro Filho.

Restrição de Integridade:
1) existência da relação de pai-filho entre os membros informados.

Exemplo:

Exclusão da associação do membro pai JAN/2003 com o membro filho 28/01/2003 .

REMOVE FATHER "JAN/2003" TO "28/01/2003" ;
Antes: $MD_{28/01/2003}$ = <"28/01/2003", "28/01/2003", N_{dt},
{MD_{5sem}, $MD_{jan2003}$}, {"não", "Terça"}>
Depois: $MD_{28/01/2003}$ = <"28/01/2003", "28/01/2003", N_{dt},
{MD_{5sem}}, {"não", "Terça"}>

4.2.1.6 Inclusão de Valores para as Métricas

Descrição: Inclusão de valores para um ou mais elemento(s) da matriz de valores base de uma métrica.

Sintaxe: **INSERT VALUE valor IN METRIC nome_metrica**
[TO (nome_membro)] ;

onde: valor = valor a ser atribuido ao elemento;
nome_metrica = nome da métrica;
(nome_membro)= lista de membros de dimensões distintas.

Restrições de Integridade:
1) existência da métrica informada;
2) valor informado de acordo com o domínio definido para a métrica;
3) a lista de membros deve conter membros de dimensões distintas e pertencentes aos níveis base definidos para a métrica.

Observações:

O comando INSERT VALUE é utilizado tanto para inicializar valores para as métricas, quanto para modificar valores preexistentes. Ele permite atribuir valores a um ou mais elementos da MVB, dependendo do número de membros de dimensão informados. Caso não for informado nenhum membro, todos os elementos da matriz são alterados com o valor informado.

Exemplos:

1) Inclusão do valor 0 para todos os elementos da métrica valor da venda.

```
INSERT VALUE 0 IN METRIC "valor da venda" ;
```

2) Inclusão do valor 300,00 para todos os elementos da métrica valor da venda, que se relacionem com o membro Canoas e o membro N3320.

```
INSERT VALUE 300,00 IN METRIC "valor da venda"
      TO "Canoas","N3320" ;
```

4.2.1.7 Exclusão de Valores para as Métricas

Descrição: Excluisão de valores de um ou mais elemento(s) da matriz de valores de uma métrica.

Sintaxe:
```
REMOVE VALUE IN METRIC nome_metrica
            [TO (nome_membro)] ;

onde:   nome_metrica = nome da métrica;
        (nome_membro)= lista de membros de dimensões
                       distintas.
```

Observações:

O comando REMOVE VALUE, altera os valores dos elementos de MVB que se relacionam com os membros de dimensão informados para o valor indefinido. As implementações de DW tratam o valor indefinido geralmente de duas formas: como sendo a inexistência de elemento (registro) para uma determinada interseção de membros, ou atribuindo um valor nulo para a função de agregação utilizada pela métrica. O valor nulo é por definição um valor que não interfere no resultado de uma função, como por exemplo o zero para a função de adição e o um para a função de multiplicação.

Restrições de Integridade:

1) existência da métrica informada;
2) a lista de membros deve conter membros de dimensões distintas e pertencentes aos níveis base definidos para a métrica.

Exemplos:

1) Exclusão de todos os elementos da métrica N° de itens no estoque associados ao membro N5120 da dimensão Produto .

```
REMOVE VALUE IN METRIC "N° de itens no estoque"
       TO "N3320" ;
```

2) Exclusão de todos os elementos da métrica N° de itens no estoque associados ao membro N5120 e ao membro 5ª Semana de 2003.

```
REMOVE VALUE IN METRIC "N° de itens no estoque"
        TO "N5120"," 5ª Semana de 2003" ;
```

4.2.2 Operações Complexas de Evolução das Instâncias

Através das sete operações atômicas de evolução de instâncias é possível definir operações complexas, como por exemplo as operações de alteração dos membros de dimensão apresentadas por (Eder; Koncilia; Morzy, 2001; Body et al., 2001):

- *SPLIT* – divisão de um membro em dois ou mais novos membros;
- *MERGE* – união de dois ou mais membros para formar um único novo membro;
- *UPDATE* – mudança dos atributos descritivos de um membro;
- *MOVE* – mudança da posição hierárquica. Isto é, o membro muda de nível de dimensão.

A figura 4.1 apresenta uma representação gráfica para o melhor entendimento dessas operações.

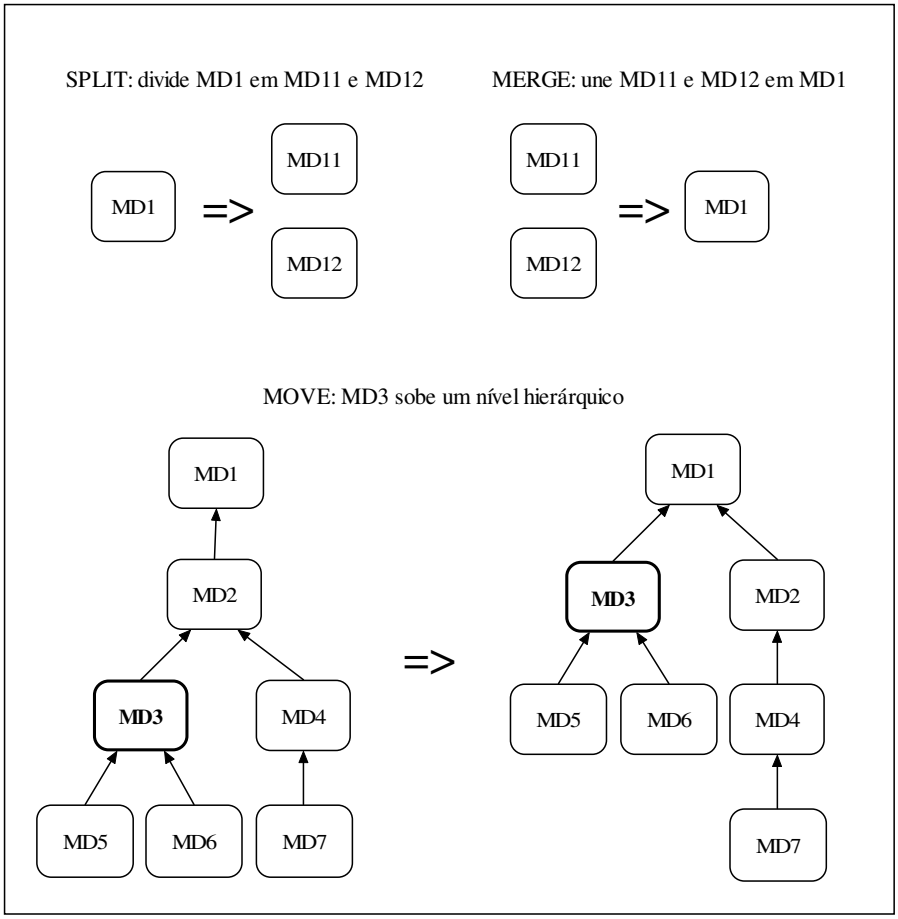

Figura 4.1: Exemplo das operações *SPLIT, MERGE* e *MOVE*

Vale observar que a mudança do nível hierárquico de MD3, no exemplo, causou a mudança do nível hierárquico de MD5 e MD6, deixando em aberto quais os membros que formarão o nível mais baixo da hierarquia de MD5 e MD6. O importante neste exemplo, é mostrar que uma mudança de nível de um membro geralmente causa mudanças em outros membros superiores e inferiores e também nos valores dos fatos, pois o valor das métricas associadas a MD2 precisam ser recalculadas devido a perda de seu filho MD3.

Para o entendimento de como as operações atômicas devem ser agrupadas para compor as operações *SPLIT*, *MERGE* e *MOVE* é mostrado detalhadamente, a seguir, a formação destas operações complexas.

4.2.2.1 Operação *SPLIT*

Objetivo: divisão de um membro em dois ou mais novos membros.

Ordem das operações:

1) operações de CREATE MEMBER para os novos membros;
2) operações de INSERT FATHER para os novos membros, utilizando os mesmos membros pais do membro original;
3) operações de INSERT VALUE para todos os elementos de todas as matrizes de valores que possuem participação do membro original;
4) operações de REMOVE FATHER para retirar todos os membros pais do membro original;
5) operações de REMOVE VALUE para todos os elementos de todas as métricas que possuem participação do membro original;
6) operação de DELETE MEMBER para excluir o membro original.

Exemplo: Divisão da cidade Campinas em duas novas cidades: Campinas1 e Campinas2.

```
1- CREATE MEMBER "Campinas1" ;
2- CREATE MEMBER "Campinas2" ;
3- INSERT FATHER "SP" TO "Campinas1";
4- INSERT FATHER "SP" TO "Campinas2";
5- INSERT VALUE 1 IN METRIC "N° itens vendidos"
        TO "Campinas1","26/01/2003", "N5120" ;
6- INSERT VALUE 0 IN METRIC "N° itens vendidos"
    TO "Campinas1","26/01/2003", "M3320" ;
7- INSERT VALUE 3 IN METRIC "N° itens vendidos"
        TO "Campinas2","26/01/2003", "N5120" ;
8- INSERT VALUE 1 IN METRIC "N° itens vendidos"
    TO "Campinas2","28/01/2003", "M3320" ;
9- INSERT VALUE 0 IN METRIC "N° itens vendidos"
    TO "Campinas1","28/01/2003", "N5120" ;
10-INSERT VALUE 0 IN METRIC "N° itens vendidos"
    TO "Campinas1","28/01/2003", "M3320" ;
11- INSERT VALUE 2 IN METRIC "N° itens vendidos"
    TO "Campinas2","28/01/2003", "N5120" ;
```

... (repetir para todos os elementos de todas as métricas que o membro Campinas participa)

```
30- REMOVE FATHER "SP" TO "Campinas" ;
31-REMOVE VALUE IN METRIC "N° itens no estoque"
        TO "Campinas";
32-REMOVE VALUE IN METRIC "N° de itens vendidos"
        TO "Campinas";
33-REMOVE VALUE IN METRIC "Valor da Venda"
        TO "Campinas";
```

Observações

A obtenção dos valores antigos (do passado) para as métricas dos membros originados de um "SPLIT" pode ser feita de duas formas gerais: através da apuração dos dados do passado ou, na falta desses, através de fórmulas aproximadas que utilizam os valores do membro original para deduzir o valor correspondente para os novos membros (Eder; Koncilia; Morzy, 2001).

4.2.2.2 Operação *MERGE*

Objetivo: união de dois ou mais membros em um único novo membro.

Ordem das operações:

1) operações de CREATE MEMBER para o novo membro;
2) operações de INSERT FATHER, utilizando os mesmos membros pais dos membros originais;
3) operações de INSERT VALUE para todos os elementos de todas as matrizes de valores que possuem participações dos membros originais;
4) operações de REMOVE FATHER para retirar todos os membros pais dos membros originais;
5) operações de REMOVE VALUE para todos os elementos de todas as métricas que possuem participações dos membros originais;
6) operação de DELETE MEMBER para excluir os membros originais.

Exemplo: União dos Produtos N5120 e M3320 em um único produto NM20.

```
1- CREATE MEMBER "NM20" ;
2- INSERT VALUE 6 IN METRIC "N° itens vendidos"
        TO "Canoas","26/01/2003", "NM20" ;
3- INSERT VALUE 3 IN METRIC "N° itens vendidos"
        TO "Viamão","26/01/2003", "M3320" ;
4- INSERT VALUE 3 IN METRIC "N° itens vendidos"
    TO "Campinas2","26/01/2003", "N5120" ;
```

... (repetir para todos os elementos de todas as métricas que os membros N5120 e M3320 participam)

```
22-REMOVE VALUE IN METRIC "N° itens no estoque"
        TO "N5120";
23-REMOVE VALUE IN METRIC "N° de itens vendidos"
    TO "N5120";
24-REMOVE VALUE IN METRIC "Valor da Venda"
        TO "N5120";
```

```
25-REMOVE VALUE IN METRIC "N° itens no estoque"
        TO "M3320";
26-REMOVE VALUE IN METRIC "N° de itens vendidos"
        TO "M3320";
27-REMOVE VALUE IN METRIC "Valor da Venda"
        TO "M3320";
```

4.2.2.3 Operação *MOVE*

Objetivo: mudança do nível de um membro.

Ordem das operações:

1) operações de REMOVE FATHER no membro original para retirar todos os membros pais do membro original;
2) operações de REMOVE FATHER nos membros filhos para retirar todos os membros filhos do membro original;
3) operação de UPDATE MEMBER para mudança do nível ;
4) operações de INSERT FATHER para informar os novos níveis pais do membro;
5) operações de INSERT FATHER para informar os novos níveis pais dos filhos antigos.

Exemplo: Mudança do nível do Membro Campinas de Cidade para Estado.

```
1- REMOVE FATHER "SP" TO "Campinas" ;
2- UPDATE MEMBER "Campinas"
        TO LEVEL "Estado" ;
```

4.2 Dependência entre as operações evolutivas

Foram apresentadas sete operações de alteração dos dados e dezessete operações de alteração do esquema que, combinadas, permitem o gerenciamento completo das mudanças de um DW. Para uma visualização global destas operações foi construída a tabela 4.1 que mostra todas as operações. A tabela também mostra a ordem natural de criação do elementos do DW e as operações prévias necessárias para a correta conclusão de uma determina operação. As duas últimas colunas (Ordem de Criação e Passos Anteriores Necessários) retratam a dependência dos elementos e operações, e são utilizadas para o correto seqüenciamento das operações de evolução.

Para exemplificar a leitura da tabela, tomamos a operação 13 – Incluir Caminho de Classificação. Ela possui ordem de criação igual a cinco, que significa que todas as operações de criação inferiores a cinco devem ser executadas antes dela eque ela deve ser executada antes de todas as operações de criação maiores que cinco.

A operação 13 também só pode ser executada depois que os níveis (operação 4) e as dimensões tenham sido criadas (operação 1).

A ordem de alteração e exclusão, por sua vez é inversa à ordem de criação, ou seja, para se excluir uma dimensão, por exemplo, é preciso excluir os valores das métricas para seus membros, seguida de seus membros, caminhos de classificação, atributos dos níveis, métricas que utilizam seus níveis, níveis de dimensão e só então a dimensão propriamente dita.

Muitos modelos e implementações de DW (Blaschka, 2000; Eder; Koncilia, 2001) tratam a exclusão de elementos complexos (uma dimensão com membros, por exemplo) através de uma única operação de exclusão. Esta por sua vez, dispara automaticamente a exclusão de todos os outros elementos dependentes da estrutura excluída.

Tabela 4.1: Lista das operações evolutivas

N°	Operação de Alteração	Tipo de Alteração	Ordem de Criação	Passos Anteriores Necessários
1	Incluir Dimensão	Esquema	1°	-
2	Alterar Dimensão	Esquema		1
3	Excluir Ddimensão	Esquema		1
4	Incluir Nível de Dimensão	Esquema	2°	1
5	Alterar Nível de Dimensão	Esquema		1
6	Excluir Nível de Dimensão	Esquema		1
7	Incluir Métrica	Esquema	3°	1, 4
8	Alterar Métrica	Esquema		1, 4, 7
9	Excluir Métrica	Esquema		1, 4, 7
10	Incluir Atributo de Nível	Esquema	4°	1, 4
11	Alterar Atributo de Nível	Esquema		1, 4, 10
12	Excluir Atributo de Nível	Esquema		1, 4, 10
13	Incluir Caminho de Classificação	Esquema	5°	1, 4
14	Excluir Caminho de Classificação	Esquema		1, 4, 13
15	Incluir Cubo	Esquema	6°	1, 4, 7, 13
16	Alterar Cubo	Esquema		1, 4, 7, 13, 15
17	Excluir Cubo	Esquema		1, 4, 7, 13, 15
18	Incluir Membro de Dimensão	Instância	7°	1, 4, 10
19	Alterar Membro de Dimensão	Instância		1, 4, 10, 18
20	Excluir Membro de Dimensão	Instância		1, 4, 10, 18
21	Incluir Pai para Membro	Instância	8°	1, 4, 10, 18
22	Excluir Pai de Membro	Instância		1, 4, 10, 18, 22
23	Inserir valor para Métricas	Instância	9°	1, 4, 7
24	Excluir valor para Métricas	Instância		1, 4, 7, 23

4.3 Considerações Finais

Neste capítulo foi apresentado detalhadamente um conjunto de vinte e quatro operações para alteração dos esquemas e instâncias de um DW, bem como as dependências e regras de integridade entre estas operações, objetivando um entendimento global dos efeitos de cada operação atômica no DW.

Os próximos capítulos irão apresentar as modificações necessárias no modelo conceitual apresentado no capítulo dois para permitir que as operações de evolução gerem históricos dos dados e esquemas ao invés de simplesmente modificar a versão atual. Explicando inclusive como deve ser feito o gerenciamento da execução destas operações.

5 SUPORTE À EVOLUÇÃO DE ESQUEMAS

Neste capítulo é apresentado um novo modelo para o gerenciamento da evolução dos esquemas baseado no armazenamento e controle de versões dos esquemas do DW. Entre os objetivos do modelo destacam-se a possibilidade de consulta de dados em esquemas anteriores à versão atual e a identificação das mudanças ocorridas em cada esquema.

5.1 Gerenciamento das Versões de Esquema

Para permitir o controle e o armazenamento do histórico das mudanças do esquema é preciso adicionar uma nova estrutura aos metadados do DW, para representar a versão do esquema. Por isso foi criada a entidade "Versão de Esquema" que possui um identificador único, utilizando internamente para referenciar os diferentes esquemas, além de dois rótulos (atributos) temporais que indicam o intervalo de vigência da referida versão. Além disso, uma versão possui um atributo para indicar seu estado atual, que pode assumir os seguintes valores: em trabalho, atual, ativa ou congelada.

No modelo proposto as versões seguem obrigatoriamente uma seqüência de criação linear, possibilitando somente a derivação de novas versões a partir da última versão criada, que é chamada de versão atual. Quando uma nova versão de esquema é derivada, ela inicia com o estado "em trabalho", não possui nenhuma instância e pode ter suas definições alteradas através da execução das operações de alteração de esquema apresentadas no capítulo três. Neste estado, a versão ainda não possui nenhum valor associado a seus rótulos temporais de vigência.

Cada operação de modificação do esquema realizada em uma versão "em trabalho" é guardada em um *script* chamado de "lista de alterações de esquema". Quando o novo esquema estiver pronto, o administrador do DW executa a operação de ativação da versão, que verifica a consistência do esquema e realiza a adaptação 0automática das instâncias existentes na versão antecessora para a nova versão. Esta operação verifica se a seqüência de comandos da lista de alterações não fere nenhuma regra de integridade do DW, constituindo uma versão de esquema válida. Caso nenhum erro for encontrado, é gerado automaticamente um novo *script* chamado de "lista de alterações de esquema e dados" que verifica as instâncias existentes na versão atual e gera comandos de alteração de dados para adaptação das instâncias atuais para o novo esquema. Resumindo, a operação de ativação inicializa a nova versão com dados da versão atual. Juntamente com a ativação, a data da atualização é atribuída à data de vigência da nova versão, a versão anterior troca do estado "atual" para o estado "ativa" e o estado da nova versão passa de "em trabalho" para "atual". Todas estas operações constituem uma única transação, não havendo portanto a ocorrência de mais de uma versão no estado "atual" ao mesmo tempo. Depois da ativação, a versão permite a

execução de operações de alteração das instâncias e a derivação de novas versões a partir dela. A figura 5.1 mostra o processo de criação e ativação de uma nova versão.

Nas versões em trabalho também é disponibilizada a operação de "verificação de consistência" que simula uma operação de ativação, com a geração da lista de alterações do esquema e dados, mas que não efetiva a mudança de estado da versão, nem inicializa ela com dados. Esta operação tem como objetivo somente relatar ao administrador do DW as mudanças que serão feitas nas instâncias para adaptá-las ao novo esquema. Desta forma, o administrador pode ter uma visão melhor das conseqüências das modificações realizadas no esquema do DW, antes delas serem efetivamente realizadas.

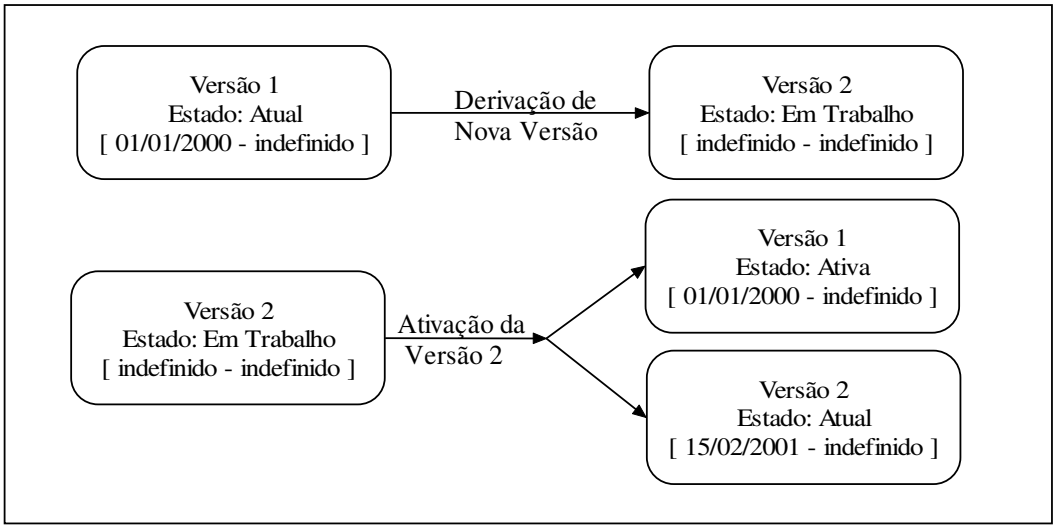

Figura 5.1: Criação e ativação de nova versão do esquema

Como pode ser visto na figura 5.1, tanto a versão 1 quanto a versão 2 do esquema possuem a data final de vigência indefinida. Isto significa que as duas versões são válidas para o tempo presente. O período de vigência de uma versão indica o intervalo de tempo em que a versão permanece disponível para atualizações. A data de início de vigência sempre é única (diferente) entre as versões e indica o dia em que a versão foi ativada, tornando-se a versão corrente do DW. Esta data não tem nenhuma relação com os dados presentes nesta versão, uma vez que os dados são copiados e adaptados da versão imediatamente anterior. Em outras palavras, podemos ter dados que foram inseridos no DW antes ou depois da data de início de vigência da versão.

A data final de vigência de uma versão, por outro lado, indica o momento em que a versão deixou de ser atualizada, se tornando uma versão congelada (imutável). Logo, uma versão com data final de vigência definida nunca possuirá dados inseridos ou modificados após esta data. A ação de definir a data final de vigência é chamada de "congelamento" e muda o estado da versão de "ativa" para "congelada".

Uma versão possui dois períodos importantes ao longo de sua existência, o período em que foi a versão atual e o período em que este ativa para alterações nas instâncias. O período em que uma versão foi a versão "atual" é definido pela data de início de vigência da versão e pela data de início de vigência de sua versão sucessora. E o período em que uma versão esteve "ativa" é definido pelas datas de início e fim de vigência da versão.

Além das operações de derivação, ativação e congelamento, existe ainda a operação de exclusão física das versões em trabalho, que consiste no descarte de uma versão em construção. A exclusão física somente pode ser feita nas versões sem data inicial de vigência, ou seja, somente nas versões em trabalho.

Vale ainda dizer que todas as versões são criadas a partir do esquema atual, sendo feita uma cópia exata deste no momento da derivação. A única exceção à regra é a primeira versão do DW que não possui versão antecessora e que deve ser criada vazia, através de um comando especial de inicialização do DW. A figura 5.2 mostra o diagrama de estados que uma versão de esquema pode apresentar, bem como as operações já descritas acima.

Observe que a operação de promoção automática é disparada pela ativação da versão sucessora, disparando portanto duas operações de transição de estado: a versão anterior passa de "atual" para "ativa" e a versão sucessora de "em trabalho" para "atual", garantido assim a existência de somente uma versão atual por ponto no tempo.

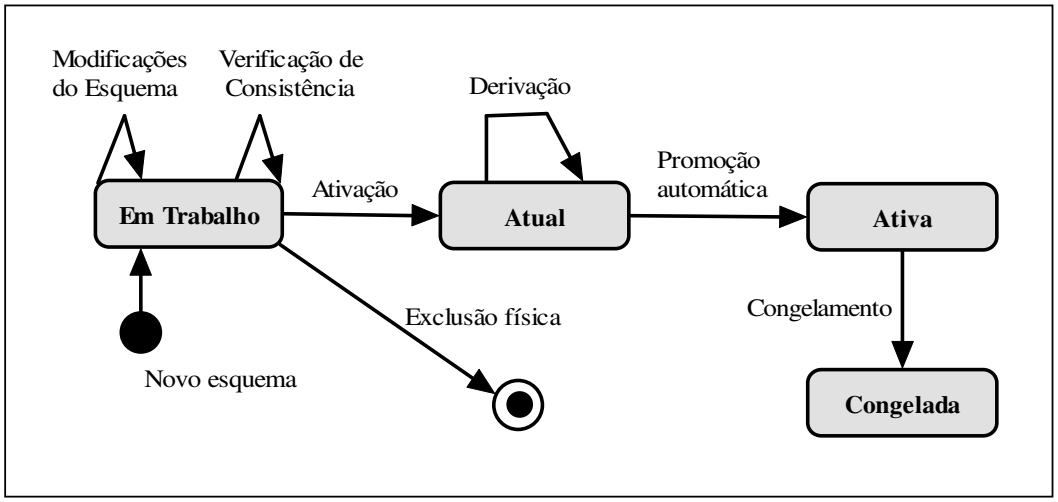

Figura 5.2: Diagrama de estados de uma versão

Para um melhor entendimento das características e restrições impostas por cada estado é apresentada a tabela 5.1. Note que podemos ter mais de uma versão de esquema em trabalho por vez, possibilitando a criação de alternativas diferentes para a versão sucessora da atual. Todavia, quando uma das versões em trabalho for ativada, todas as outras versões em trabalho serão automaticamente excluídas.

5.2 Novas Definições para o Modelo de DW Proposto

Para armazenar e controlar as diferentes versões de esquema, é preciso acrescentar novas estruturas e atributos ao modelo apresentado no capítulo 2, de forma que seja possível referenciar as diferentes versões, bem como consultar as mudanças ocorridas nas definições do DW ao longo do tempo.

Tabela 5.1 - Tabela comparativa dos estados de um esquema

Estados & Características	Em Trabalho	Atual	Ativa	Congelada
Permite Alterações das definições	**Sim**	Não	Não	Não
Permite Derivações de novos esquemas	Não	**Sim**	Não	Não
Permite Alterações dos dados	Não	**Sim**	**Sim**	Não
Permite consulta aos dados	Não	**Sim**	**Sim**	**Sim**
Permite mais de uma versão neste estado	**Sim**	Não	**Sim**	**Sim**

5.2.1 Definição de Versão de Esquema

A primeira estrutura a ser acrescentada é a entidade versão, que possui um atributo para indicar o seu estado, dois rótulos temporais: início e fim de vigência e um identificador único gerado automaticamente pelo DW. Para o usuário final, a identificação da versão será feita pela data inicial de vigência, que é única entre as versões.

$VERSAO_{id}$ = < Id, DataIni, DataFim, Estado >

```
Onde:    Id      = Idenficador único da versão;
         DataIni = Data inicial de vigência;
         DataFim = Data final de vigência;
         Estado  = {Em trabalho, Atual, Ativa, Congelada}.
```

Para relacionar as definições dos esquemas às suas respectivas versões, são adicionados dois rótulos temporais (data início e data fim) em todos os atributos e entidades que podem variar de uma versão para outra.

A maioria dos autores, como Blaschka (2000), Eder e Koncilia (2001) e Günzel (2000), não tratam a mudança dos nomes das entidades (cubos, dimensões, métricas, níveis e atributos de dimensões) como sendo operações de mudança do esquema, ou tratam estas modificações como sendo a exclusão de um elemento e inclusão de outro diferente. Entretanto, esta abordagem pode gerar confusão na análise das diferenças entre os esquemas, pois uma simples mudança do nome de uma dimensão não significa que a dimensão como um todo deixou de existir no esquema e que outra dimensão diferente foi incluída. Por isso guardamos o histórico das alteração dos nomes dos elementos, de forma que fique explícita a operação de mudança de nomes.

Os elementos que descrevem o esquema do DW receberam então dois atritutos temporais para caracterizar o período em que cada entidade, atributo ou relação do esquema, esteve vigente e imutável. Na figura 5.3 é apresentado um meta-esquema relacional dos elementos do DW. Note que muitos atributos (nome do nível, domínio da métrica, função da métrica, domínio do atributo de nível) foram transformados em entidades independentes, para permitir a existência de mais de um valor ao longo do tempo. Essas novas entidades aparecem em tom mais claro na figura, enquanto que as entidades primárias do modelo aparecem sombreadas num tom mais escuro. A entidade

versão não foi apresentada neste meta-esquema, mas ela se relaciona indiretamente com todas as outras entidades, como mostra o exemplo da figura 5.4.

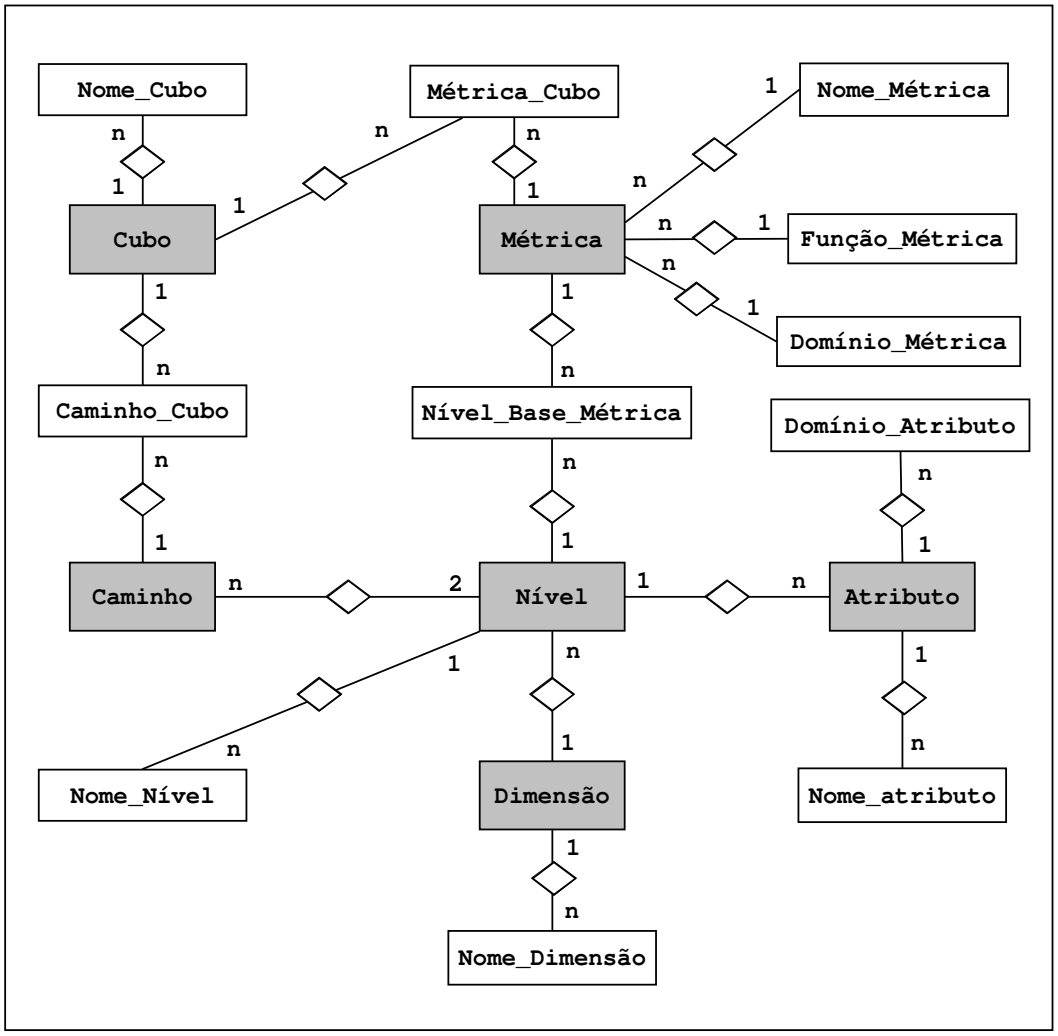

Figura 5.3: Meta-esquema das definições do DW

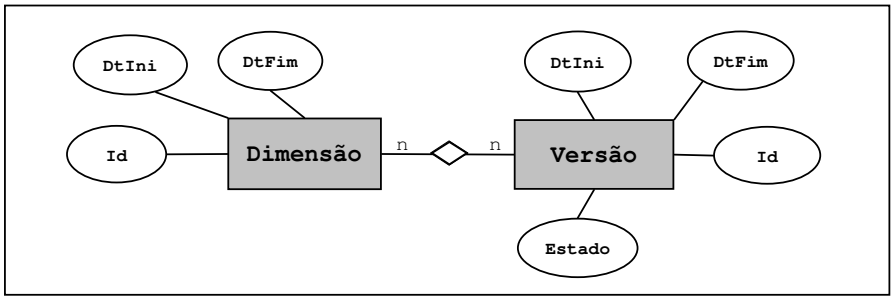

Figura 5.4: Relacionamento entre as entidades Versão e Dimensão

O relacionamento da versão com cada uma das entidades do meta-esquema ocorre através da seguinte regra temporal: uma entidade pertence a uma versão se a data de início de vigência da versão pertence ao intervalo de validade da entidade. Vale lembrar que a data de final de validade pode ser indefinida, significando que a entidade é válida para qualquer data igual ou superior à sua data de início de validade.

A seguir são apresentadas com detalhes as modificações feitas sobre o modelo conceitual para permitir o armazenamento do histórico das definições.

5.2.2 Definição de Cubo Versionado

Um cubo, além de ser criado e excluído, pode sofrer alterações no seu nome, alterações na composição de suas métricas (inclusão e exclusão de métricas) e alterações nas hierarquias de suas dimensões (inclusão ou exclusão de caminhos de classificação) (Günzel, 2000). Para que cada alteração realizada seja devidamente armazenada, possibilitando o perfeito entendimento da mudança ocorrida no cubo, a definição inicial de cubo precisou ser desmembrada da seguinte forma:

Definição original:

$Cubo_{id}$ = <Id, Nome, F, CC>

Novas Definições:

```
Cubo_id         = <IdCubo, DtIniCubo, DtFimCubo >
Nome_Cubo     = <IdCubo, NomeCubo, DtIniNome ,DtFimNome>
Metrica_Cubo = <IdCubo, IdMetrica, DtIniMetrica,
                  DtFimMetrica>
Caminho_Cubo = <IdCubo, IdCaminho, DtIniCaminho,
                  DtFimCaminho>
```

```
Onde:    IdCubo           =   identificador único do cubo;
         DtIniCubo        =   data de criação do cubo;
         DtFimCubo        =   data de exclusão do cubo;
         NomeCubo         =   nome do cubo;
         DtIniNome        =   primeiro dia de validade desse
                              nome de cubo;
         DtFimNome        =   ultimo dia de validade desse
                              nome de cubo;
         IdMetrica        =   idenficador único da métrica
                              associada ao cubo;
         DtIniMetrica     =   primeiro dia de validade da
                              associação da métrica ao cubo;
         DtFimMetrica     =   último dia de validade da
                              associação da métrica ao cubo;
         IdCaminho        =   identificador único do caminho;
         DtIniCaminho     =   primeiro dia  da associação do
                              caminho ao cubo;
         DtFimCaminho     =   último dia da associação do
                              caminho ao cubo.
```

Observações

Os nomes, métricas e caminhos do cubo foram transformados em relacionamentos temporais atômicos, permitindo o armazenamento detalhado de qualquer operação de modificação do cubo. Para se encontrar a definição dos cubos de uma determinada versão, basta verificar quais são os cubos com data de criação menor ou igual a data inicial de vigência da versão com data de exclusão maior que a data inicial de vigência. Para se saber qual a configuração do cubo nesta versão, basta seguir a mesma lógica para os relacionamentos de nomes, métricas e caminhos.

A figura 5.5 apresenta um exemplo da identificação dos cubos válidos de uma determinada versão. O gráfico mostra o período de existência dos cubos através de uma linha, podendo-se identificar que somente os cubos 1, 3 e 4 são válidos na versão ativada no dia 20/03/2002. O cubo 2 foi excluído exatamente na ativação da versão do dia 20/03/2002 e o cubo 5 somente foi criado na versão de 01/05/2003.

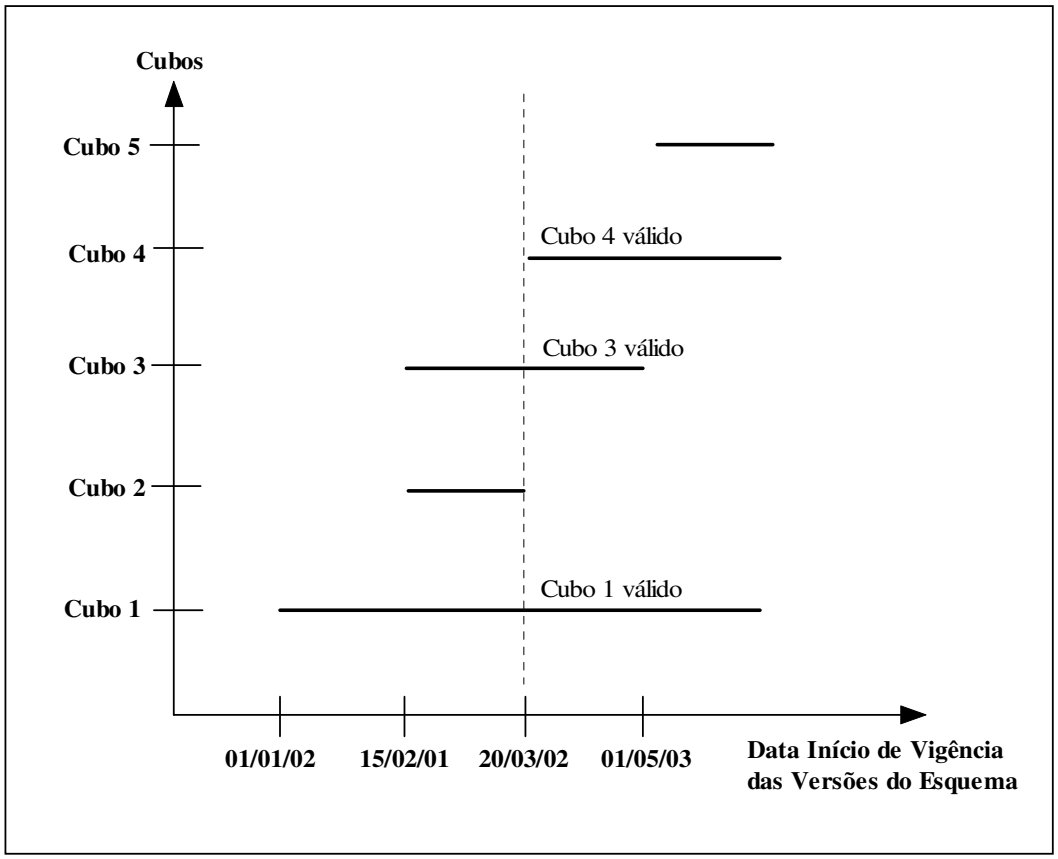

Figura 5.5: Cubos da versão de esquema de 20/03/2002

5.2.3 Definição de Métrica Versionada

Uma métrica, além de ser criada e excluída, pode sofrer alterações no seu nome, domínio, função de agregação e lista de níveis dimensionais base.

Definição original:
M_{id} = <Id, Nome, Tipo_Domínio, f_Agreg ,MHD>

Novas Definições:
```
M_id               = <IdMetrica, DtIniMetrica, DtFimMetrica>
Nome_Metrica       = <IdMetrica, NomeMetrica, DtIniNome,
                      DtFimNome>
Dominio_Metrica    = <IdMetrica, Tipo_Dominio, DtIniDominio,
                      DtFimDominio>

Funcao_Metrica     = <IdMetrica, f_Agreg, DtIniFunc, DtFimFunc>
NivelBase_Metrica = <IdMetrica, IdNivel, DtIniBase,
                       DTFimBase>
```

```
Onde:    IdMetrica         =   identificador único da métrica;
         IdNivel           =   identificador único do nível;
         DtIniMetrica      =   data de criação da métrica;
         DtFimMetrica      =   data de exclusão da métrica;
         DtIniNome         =   primeiro dia de validade desse
                               nome para a métrica;
         DtFimNome         =   último dia de validade desse
                               nome para a métrica;
         DtIniDominio      =   primeiro dia de validade desse
                               domínio para a métrica;
         DtFimDominio      =   último dia de validade desse
                               domíno para a métrica;
         DtIniFunc         =   primeiro dia de validade dessa
                               função para a métrica
         DtFimFunc         =   último dia de validade dessa
                               função para a métrica
         DtIniBase         =   primeiro dia de validade desse
                               nível base para a métrica
         DtFimBase         =   último dia de validade desse
                               nível base para a métrica
         NomeMetrica       =   nome dado à métrica;
         Tipo_dominio      =   tipo e domínio da métrica;
         f_Agreg           =   função de agregação da métrica.
```

5.2.4 Definição de Dimensão Versionada

Uma dimensão, além de ser criada e excluída, pode ter o seu nome modificado.

Definição original:
D_{id} = <Id, Nome>

Novas Definições:
```
D_id            = <IdDimensão, DtIniDimensão, DtFimDimensão>
Nome_Dimensão = <IdDimensão, NomeDimensão, DtIniNome,
                  DtFimNome>
```

```
Onde:    IdDimensão        =   identificador único da dimensão;
         DtIniDimensão     =   data de criação da dimensão;
         DtFimDimensão     =   data de exclusão da dimensão;
         DtIniNome         =   primeiro dia de validade desse
```

```
                              nome para a dimensão;
        DtFimNome         =   último dia de validade desse
                              nome para a dimensão;
        NomeDimensão      =   nome dado à dimensão.
```

5.2.5 Definição de Nível de Dimensão Versionado

Um nível de dimensão, além de ser criado e excluído, pode ter o seu nome e níveis pais modificados.

Definição original:
N_{id} = <Id, Nome, D_{id}, NP>

Novas Definições:

```
N_id           = <IdNivel, IdDimensão, DtIniNivel, DtFimNivel>
Nome_Nivel     = <IdNivel, NomeNivel, DtIniNome,
                  DtFimNome>

Onde:   IdNivel       =   identificador único do nível;
        IdDimensão    =   identificador único da
                          dimensão;
        DtIniNivel    =   Data de criação do nível;
        DtFimNivel    =   Data de exclusão do nível;
        DtIniNome     =   primeiro dia de validade desse
                          nome para o nível;
        DtFimNome     =   último dia de validade desse
                          nome para o nível;
        NomeNivel     =   nome dado ao nível.
```

5.2.6 Definição de Atributo de Dimensão Versionado

Um atributo de dimensão, além de ser criado e excluído, pode ter o seu nome e domínio modificados.

Definição original:
A_{id} = <Id, Nome, N_{id}, Tipo_domínio>

Novas Definições:

```
A_id                = <IdAtributo, IdNivel, DtIniAtributo,
                       DtFimAtributo>
Nome_Atributo       = <IdAtributo, NomeAtributo, DtIniNome,
                       DtFimNome>
Dominio_Atributo    = <IdAtributo, Tipo_dominio, DtIniDominio
                       DtFimDominio>

Onde:   IdAtributo        =   identificador único do atributo
                              descritivo;
        IdNivel           =   identificador único do nível do
                              atributo;
        DtIniAtributo     =   data de criação do atributo;
        DtFimAtributo     =   data de exclusão do atributo;
        DtIniNome         =   primeiro dia de validade desse
```

```
                          nome para o atributo;
        DtFimNome         =   último dia de validade desse nome
                              para o atributo;
        DtIniDominio      =   primeiro dia de validade desse
                              domínio para o atributo;
        DtFimDominio      =   último dia de validade desse
                              domínio para o atributo;
        NomeAtributo      =   nome dado ao atributo;
        Tipo_dominio      =   tipo e domínio do atributo.
```

5.2.7 Definição de Caminho de Classificação Versionado

Um caminho de classificação somente pode ser criado ou excluído.

Definição original:
C_{id} = <Id, NF_{id}, NP_{id} >

Novas Definições:

C_{id} = <IdCaminho, IdNivelFilho, IdNivelPai, DtIniCaminho, DtFimCaminho>

```
Onde:   IdCaminho    =   identificador único do caminho
                         de classificação;
        IdNivelFilho =   identificador único do nível filho;
        IdNivelPai   =   identificador único do nível pai;
        DtIniCaminho =   data de criação do caminho;
        DtFimCaminho =   data de exclusão do caminho;
```

5.3 Operações sobre as Versões

Apresentadas as modificações necessárias no modelo para o controle das versões do esquema, vamos detalhar as operações de criação e mudança de estado das versões, mostrando como os rótulos temporais são inicializados e modificados em cada tipo de operação.

5.3.1 Operação de Derivação da Versão

Todas as versões, com excessão do primeiro esquema, são derivadas a partir do esquema atual, que é por definição o esquema com maior data inicial de vigência. A operação de derivação cria uma nova versão, idêntica ao esquema atual, mas sem dados populados. A nova versão recebe um identificador único e inicia sua existência no estado "em trabalho". As datas de início e final de vigência permanecem indefinidas neste estado.

$VERSAO_{id}$ = <Id, DataIni, DataFim, Estado>
$VERSAO_2$ = <2, Indeterminado, Indeterminado, "Em Trabalho">

Por motivos de redução de espaço, em alguns lugares será utilizado o símbolo "?" para representar uma data indeterminada.

Exemplo: $VERSAO_2$ = <2, ?, ?, "Em Trabalho">

5.3.2 Operações de Mudança do Esquema

Os esquemas em trabalho são os únicos que permitem a execução de operações de mudança de esquema, sendo que cada operação executada (criação de métricas, modificações de cubos, exclusão de dimensões, etc) é armazenada em uma lista ordenada de alterações de esquema, que será utilizada na operação de ativação da versão.

Exemplo de uma lista de alterações de esquema

Tendo como base o exemplo de esquema definido no capítulo 2, deseja-se incluir um novo nível na dimensão tempo e utilizar o novo nível no cubo Vendas. Também deseja-se excluir a métrica Nº de itens vendidos do DW.

Lista das operações de alteração de esquema necessárias:

1) Inclusão do nível de dimensão Ano na dimensão Tempo;
2) Inclusão do caminho Mês => Ano para a hierarquia da dimensão Tempo;
3) Associação do caminho Mês => Ano para o cubo de Vendas;
4) Remoção da métrica Nº de itens vendidos do Cubo Vendas;
5) Exclusão da métrica Nº de itens vendidos;

Lista de comandos:

```
1) CREATE LEVEL "Ano"
        TO DIMENSION "Tempo";
2) CREATE DIMENSION PATH "Ano"
        TO "Mês";
3) UPDATE CUBO "Vendas"
        INSERT PATHES "Mês" TO "Ano";
4) UPDATE CUBO "Vendas"
        REMOVE METRICS "N° de itens vendidos";
5) DELETE METRIC "N° de itens vendidos";
```

5.3.3 Operação de Verificação de Consistência da Versão

A operação de verificação de consistência executada sobre um esquema em trabalho, tem como principal objetivo relatar ao administrador do DW o efeito que as mudanças executadas irão causar nas instâncias. Esta operação verifica todas as instâncias afetadas pela modificação do esquema, gerando um relatório das adaptações que serão feitas para adequar as instâncias ao novo esquema.

Com este relatório, o administrador pode verificar problemas semânticos relacionados com as instâncias, bem como antecipar a criação de seus *scripts* de adaptação, que serão executados após a ativação da nova versão.

Neste trabalho não serão tratadas com detalhes as diversas políticas e estratégias de migração dos dados que podem ser utilizadas, uma vez que elas são fortemente dependentes da forma de implementação, regras e ferramentas dos SGBDs utilizados. Entretanto, pode-se afirmar que de uma forma geral, os SGBDs comerciais não

oferecem um bom suporte para adaptação das instâncias, sendo a maior parte deste trabalho destinado ao administrador do BD (Blaschka, 1999).

5.3.4 Operação de Ativação da Versão

A operação de ativação da versão, executada sobre um esquema em trabalho, realiza as seguintes tarefas:

1) verificação da consistência global do esquema a ser ativado, não permitindo, por exemplo, a existência de dimensões não utilizadas por nenhuma métrica;
2) geração da lista de alterações de esquema e instâncias e população do novo esquema com os dados vindos da versão "atual" do esquema;
3) modificação do estado da versão "atual" para "ativa" ;
4) modificação do estado da versão "em trabalho" para "atual", atribuindo a data corrente da execução da operação no atributo data início de vigência da versão;
5) atribuição da data corrente como data de início de validade para todas as novas definições (cubos, dimensões, métricas) incluídas na nova versão em comparação com a versão antecessora (operações de *CREATE*);
6) atribuição da data imediatamente anterior à data corrente (data corrente – 1) como data de fim de validade para todas as definições excluídas da versão anterior do esquema (operações de *DELETE*);
7) para as definições modificadas (operações de *UPDATE*), atribuição da data imediatamente anterior à data corrente (data corrente – 1) na data final de validade das definições alteradas, e criação de nova ocorrência da definição com data de início de validade igual à data corrente.

Exemplo de atribuição das datas de validade

Tomando o mesmo exemplo de modificação de esquema apresentado na seção 5.3.2, os tempos de validade das definições seriam alterados da seguinte forma para a data de ativação de 05/09/2003:

1) A inclusão do nível de dimensão Ano na dimensão Tempo cria duas novas definições:
   N_{ano} = <"ano", "temp", 05/09/2003, ?>
   Nome_Nivel = <"ano", "ano", 05/09/2003, ?>

2) A inclusão do caminho Mês => Ano para a hierarquia da dimensão Tempo cria uma nova definição:
   $C_{mês->ano}$ = <"mês->ano", "mês", "ano", 05/09/2003, ?>

3) A associação do caminho Mês => Ano para o cubo de Vendas cria uma nova definição:
   Caminho_Cubo = <"vend", $C_{mês->ano}$, 05/09/2003, ?>

4) A remoção da métrica Nº de itens vendidos do Cubo Vendas fecha o período de validade da métrica para a composição do cubo. Caso o cubo não tivesse outra métrica associada, ele também teria seu período de validade fechado.

```
Antes: Metrica_Cubo = <"vend", "valor_vend", 01/01/2000, ?>
Depois: Metrica_Cubo = <"vend", "valor_vend", 01/01/2000,
                        04/09/2003>
```

5) A exclusão da métrica Nº de itens vendidos fecha o período de validade da métrica e de todos os seus atributos e relacionamentos (nome, domínio, função, níveis base):

M_{nro_vend} = <"nro_vend", 01/01/2000, 04/09/2003>
Nome_Metrica = <"nro_vend", "N° de itens vendidos", 01/01/2000, 04/09/2003>
Dominio_Metrica = <"nro_vend", "Inteiro Positivo", 01/01/2000, 04/09/2003>
Funcao_Metrica = <"nro_vend", "SOMA", 01/01/2000, 04/09/2003>
NivelBase_Metrica = <"nro_vend", N_{dt}, 01/01/2000, 04/09/2003>
NivelBase_Metrica = <"nro_vend", N_{cid}, 01/01/2000, 04/09/2003>
NivelBase_Metrica = <"nro_vend", N_{pr}, 01/01/2000, 04/09/2003>

5.3.5 Operação de Congelamento da Versão

Uma versão congelada é uma versão do esquema que possui o período de vigência determinado, sendo que a data de início de vigência da versão indica o dia em que ocorreu a ativação da versão e a data de fim de vigência, o dia em que as instâncias desta versão pararam de ser atualizadas. As versões congeladas, de um modo geral, não são mais vistas e acessadas pela maioria dos usuários do DW. Sua principal finalidade é permitir consultas eventuais aos seus dados para a compreensão de consultas e relatórios gerados no passado, principalmente consultas para auditoria ou entendimento de um cenário anterior. Os dados de uma versão congelada servem como um tipo de *backup* das definições e dados do passado.

A operação de congelamento de uma versão é muito simples, consistindo somente da atribuição da data corrente para a data de fim de vigência da versão. A única validação que precisa ser feita é verificar se nenhum *Data Mart* ou aplicação OLAP faz referência a esta versão do esquema.

5.3.6 Operação de Exclusão Física da Versão em Trabalho

As versões em trabalho podem ser descartadas a qualquer momento através da operação de exclusão física, que remove a versão do esquema sem guardar o histórico de suas definições. Esta operação geralmente é executada por dois motivos:

- abandono da versão em trabalho – o desenvolvedor resolve não mudar o esquema atual ou pretende recomeçar suas alterações a partir de uma nova derivação da versão atual;
- eleição de um esquema para ativação – o desenvolvedor pode derivar e construir diferentes versões de esquema, de forma a analisar e decidir por um deles na hora da ativação. No caso de existir mais de um esquema em trabalho no momento da conclusão da operação de ativação de um esquema, todos os

outros esquemas em trabalho são automaticamente excluídos fisicamente, garantindo assim a linearidade da derivação e da evolução das definições do DW.

5.4 Referência e Consulta às Versões de Esquema

As versões do esquema do DW são identificadas pelos usuários finais e outros sistemas (*Data Marts*, aplicações OLAP) pela sua data de início de vigência, que é única. Entretanto, é muito importante a utilização de uma camada intermediária entre o DW e as aplicações, que permita a execução de consultas sem referência direta a uma versão.

Segundo Vassiliadis (2000) um ambiente de DW deve ser formado por módulos o mais independentes possíveis, de forma que as modificações em um módulo não afetem o funcionamento de outras partes do DW. Por isso, para manter a compatibilidade entre as aplicações que desconhecem o versionamento dos esquemas e o DW, é preciso utilizar uma camada intermediária para a escolha da versão do esquema apropriada para cada aplicação (Nica, 1999). Vários critérios podem ser utilizados para a escolha da versão, dentre as quais destacam-se as seguintes:

- atribuição explícita da versão que cada aplicação deve utilizar pelo administrador do DW. Ou seja, uma pessoa controla um cadastro das versões a serem utilizadas por cada aplicação;
- utilização do esquema atual no caso de não especificação da versão desejada;
- solicitação explícita da identificação da versão no momento do acesso ao DW (Eder; Koncilia, 2002);
- utilização de heurísticas para escolha de uma boa versão – a *interface* pode analisar os parâmetros das consultas como o intervalo temporal de pesquisa, os nomes das tabelas e atributos pesquisados, referenciando a primeira ou a melhor versão que satisfaça os parâmetros da consulta (Body et al., 2002).

5.4.1 Consulta ao Histórico de Modificações do DW

Uma vez que temos armazenado o histórico das alterações do esquema, podemos fazer dois tipos básicos de consultas sobre as definições do esquema:

- consulta sobre as definições de uma determinada versão do esquema;
- consulta sobre a evolução de certas definições ao longo das versões do DW.

No primeiro caso, a consulta é feita a partir de uma única versão devidamente identificada, tendo como objetivo pesquisar a estrutura do DW em uma determinada data. Para a identificação da versão desejada, existem duas opções: ou o usuário informa explicitamente a data de início de vigência da versão (que é um identificador único) que ele deseja pesquisar, ou ele informa uma data qualquer, deixando para o processador de consultas a tarefa de encontrar a versão "atual" do DW para a data informada.

No caso de consultas feitas para descobrir quais as versões em que uma ou mais definições são válidas, é preciso utilizar a regra que diz que uma entidade pertence a uma versão se a data de início de vigência da versão pertence ao intervalo de validade da entidade.

A seguir são demostrados alguns exemplos de consultas sobre as definições das versões do esquema feitas em SQL.

1) Quais as dimensões que compõem a métrica Nº de itens vendidos na versão 01/01/2000 ?

```
SELECT nome_dimensao.nome
FROM dimensao, metrica, nome_metrica,
     niv_base_metrica, nome_dimensao
WHERE nome_metrica.nome = "N° de itens vendidos" AND
       metrica.idmetrica = nome_metrica.idmetrica AND
       niv_base_metrica.idmetrica = metrica.idmetrica AND
       niv_base_metrica.datini <= 01/01/2000 AND
       (niv_base_metrica.datfim >= 01/01/2000 OR
       niv_base_metrica.dtfim IS NULL)  AND
       dimensao.iddimensao = niv_base_metrica.iddimensao AND
       dimensao.datini <= 01/01/2000 AND
      (dimensao.datfim >= 01/01/2000 OR dimensao.datfim IS NULL) AND
       nome_dimensao.iddimensao = dimensao.iddimensao AND
       nome_dimensao.datini <= 01/01/2000 AND
       (nome_dimensao.datfim >= 01/01/2000 OR
       nome_dimensao.datfim IS NULL)
```

Resultado: `Tempo  Local  Produto`

Observações:

- como a única informação necessária da versão é a data de início de vigência, não foi preciso colocá-la na cláusula FROM;
- também não foi preciso acrescentar o nível na cláusula FROM, pois basta o campo iddimensao da tabela niv_base_metrica para se chegar às dimensões.

2) Quais os níveis da dimensão Tempo da versão atual de 08/10/2003 ?

```
SELECT nome_nivel.nome
FROM dimensao, nome_dimensao, nivel, nome_nivel, versao ver_atual
WHERE ver_atual.datini <= 08/10/2003 AND
      NOT EXISTS (SELECT *
                  FROM versao ver_outras
                  WHERE ver_outras.datini <= 08/10/2003 AND
                        ver_outras.datini > ver_atual.datini) AND
       dimensao_nome.nome = "Tempo" AND
       dimensao.iddimensao = dimensao_nome.iddimensao AND
       dimensao.datini <= 08/10/2003 AND
      (dimensao.datfim >= 08/10/2003 OR dimensao.datfim IS NULL) AND
       nivel.iddimensao = dimensao.iddimensao AND
       nivel.datini <= 08/10/2003 AND
```

```
(nivel.datfim >= 08/10/2003 OR nivel.datfim IS NULL) AND
nome_nivel.iddimensao = dimensao.iddimensao AND
nome_nivel.datini <= 01/01/2000 AND
(nome_nivel.datfim >= 01/01/2000 OR
nome_nivel.datfim IS NULL)
```

Resultado: `Data  Semana  Mês  Ano`

3) Quais os níveis que a dimensão Tempo possuiu no período de 01/05/2000 a 01/05/2004 e quais os seus períodos de validade ?

```
SELECT nome_nivel.nome nivel.dtini nivel.dtfim
FROM dimensao, nome_dimensao, nivel, nome_dimensao
WHERE nome_dimensao.nome = "Tempo" AND
      dimensao.iddimensao = nome_dimensao.iddimensao
      nivel.iddimensao = dimensao.idnivel AND
      nivel.dtini >= 01/05/2000 AND
      (nivel.dtfim <= 01/05/2004 or nivel.dtfim IS NULL) AND
       NOT EXISTS (SELECT *
                   FROM nome_nivel nome_outros
                   WHERE nome_outros.idnivel = nivel.idnivel AND
          nome_outros.datini > nome_nivel.datini )
```

Resultado:

Nome do Nível	Data Início	Data Fim
Data	01/01/2000	NULL (indeterminada)
Semana	01/01/2000	NULL (indeterminada)
Mês	01/01/2000	NULL (indeterminada)
Ano	05/09/2003	NULL (indeterminada)

Observações:

- a cláusula "*not exist*" foi colocada para retornar somente o nome mais atual do nível para o período pesquisado, pois o objetivo da consulta não é mostrar os diferentes nomes que o nível possuiu no intervalo pesquisado.

4) Quais as versões de esquema do DW que possuem a métrica Valor das Vendas e qual o estado destas versões ?

```
SELECT versao.id versao.dtini versao.dtfim
FROM   versao, metrica, nome_metrica
WHERE  nome_metrica.nome = "Valor das Vendas" AND
       metrica.idmetrica = nome_metrica.idmetrica AND
       metrica.dtini <= versao.dtini AND
       (metrica.dtfim >= versao.dtini OR metrica.dtfim IS NULL)
```

Resultado:

```
Id          Data Início   Data Fim                Estado
-----       -----------   -------------------     --------
1           01/01/2000    NULL (indeterminada)    Ativa
2           05/09/2003    NULL (indeterminada)    Atual
```

5.5 Armazenamento dos Dados de uma Versão

Sempre que uma versão de esquema é ativada, é feita a população dos dados do novo esquema a partir dos dados do esquema antecessor. Isto implica em criar uma relação entre os dados e as versões de esquema. Ou seja, é preciso indicar a qual versão do esquema pertence uma determinada instância.

Esta ligação entre instâncias e versões pode ser feita de duas formas:

- incluindo explicitamente relacionamentos N x N entre as instâncias e as versões de esquema;
- armazenando as instâncias em repositórios diferentes, sendo um repositório de dados para cada versão.

A inclusão de relacionamentos entre instâncias e versões diminui a redundância de dados, mas aumenta a complexidade de gerenciamento. A performance também é muito prejudicada devido principalmente ao aumento (aproximadamente de duas vezes) do número de junções realizadas em cada consulta aos dados de uma versão. Isto sem contar os problemas técnicos causados por limitações atuais, como por exemplo, a impossibilidade de se gravar valores com domínios (tipos) diferentes em um mesmo campo de tabela (Roddick, 1994). Essa situação pode ocorrer nos atributos descritivos ou nas métricas de versões diferentes do esquema.

Já o armazenamento de cada versão em um repositório distinto, apesar do aumento de redundância global dos dados, mantém a performance muito próxima da performance de um DW não versionado. A pequena perda desempenho deve-se a identificação da versão a ser referenciada, trabalho executado pela camada intermediária de comunicação entre as aplicações e o DW.

Neste trabalho será explorada a alternativa de repositórios distintos para cada versão, por ser a solução mais viável para implementação atualmente.

5.6 Modificações nas Definições das Instâncias

Para identificar a versão de esquema de uma instância, foi adicionado o ID da versão nas tuplas que definem os membros e os valores das métricas. Este ID indica a que versão do esquema pertence a instância. As novas definições das instâncias são apresentadas a seguir:

5.6.1 Definição dos Membros de Dimensão

Definição original:

MD_{id} = <Id, Nome, N_{id}, MDP, MAD>

Nova Definição:

MD_{id} = < IdVersao, IdMembro, Nome, IdNivel, MDP, MAD>

```
Onde:    IdVersao = identificador único da versão;
         IdMembro = identificador único do membro;
         Nome     = nome do membro;
         IdNivel  = identificador único do nível;
         MDP      = conjunto de membros pais;
         MAD      = conjunto ordenado de valores
                    atribuídos aos atributos do nível.
```

5.6.2 Definição dos Valores das Métricas

Definição original:

VB = <M_{id}, $MD_{id\ x}$ $MD_{id\ x}$ ... MD_{id}, VAL >

Nova Definição:

VB = < IdVersao, IdMetrica, $MD_{id\ x}$ $MD_{id\ x}$ ... MD_{id} , VAL >

```
Onde:    IdVersao     = identificador único da versão;
         IdMetrica    = identificador único do métrica;
         MDid         = identificador único de membro
                        de dimensão;
         VAL          = valor correspondente à interseção
                        dos membros indicados.
```

5.7 Considerações Finais

Neste capítulo foi apresentado uma forma prática de gerenciamento da evolução do esquema de um DW. Todas as definições do esquema são devidamente associadas a datas de início e término de existência, permitindo o rastreamento de todas as transformações feitas no esquema do DW ao longo de sua vida.

A utilização de múltiplos repositórios independentes para armazenamento dos dados (um para cada versão do esquema) permite a execução de consultas sobre os dados de qualquer uma das versões, sem grandes perdas de desempenho. Além disso, é possível manter várias versões do esquema ativas, de modo que as aplicações não precisam ser necessariamente revistas e alteradas a cada modificação do esquema do DW.

Grande ênfase foi dada ao controle da transição das instâncias de um esquema para outro, no momento da ativação de uma versão. O processo de adaptação das instâncias deve ser seguro e transparente. Isto é, ele deve respeitar todas as regras

semânticas do modelo multidimensional, bem como ser perfeitamente compreendido pelos administradores do DW. Motivo pelo qual é indicado a utilização de operações de verificação de consistência e geração de *scripts* de atualização.

No próximo capítulo são feitas novas modificações nas definições das instâncias das versões para permitir o armazenamento do histórico de modificações dos dados de uma versão do esquema. Com isso, possibilitaremos consultas sobre os valores dos dados de uma determinada versão do esquema em um determinado instante do tempo, garantido assim uma visão completa da evolução dos dados (instâncias) e metadados (esquemas) do DW.

6 SUPORTE À EVOLUÇÃO DE ESQUEMAS E DADOS

No capítulo anterior foi definido um modelo para permitir o armazenamento das versões de esquema do DW, sendo os dados destas versões armazenados em repositórios distintos (Galante; Edelweiss; Santos, 2002). Entretanto, para se ter o histórico completo da evolução de um DW é importante guardar também as modificações ocorridas nas instâncias de cada esquema.

Para possibilitar o armazenamento e a consulta do histórico de valores das instâncias (membros e valores das métricas) foi adicionado um período de validade a todos os elementos das definições destas instâncias que são suscetíveis de modificações, como é apresentado na seção 6.1.

Além disso, é preciso controlar os períodos de vigência das instâncias, de forma a acrescentar novas regras e procedimentos nas operações básicas de inclusão, alteração e exclusão de dados (Edelweiss, 1998; Chamoni; Stock, 1999). As seções 6.2 e 6.3 tratam do gerenciamento da validade das instâncias, explicando como deve ser feito o seu controle.

Outro aspecto muito importante, tratado na seção 6.4 e 6.5, é a forma de acesso aos dados, isto é, a linguagem e interface de consulta utilizada para pesquisar os dados de uma ou mais versões dos esquemas e instâncias.

6.1 Novas Definições para Armazenar o Histórico das Instâncias

Para permitir o armazenamento do histórico dos dados do DW, foram incluídos dois novos elementos nas definições dos membros e valores das métricas: data início e data fim de validade. Também foi preciso incluir período de validade nos atributos que permitem alteração de seus valores, como por exemplo o nome e nível do membro. A seguir são apresentadas, em detalhes, as modificações feitas nas definições das instâncias.

6.1.1 Definição dos Membros de Dimensão

Definição anterior:

MD_{id} = <IdMembro, IdVersao, Nome, IdNivel, MDP, MAD>

Novas Definições:

```
MDid              = <IdMembro, IdVersao, DtIniMembro, tFimMembro>
Nome_Membro       = <IdMembro, IdVersao, Nome, DtIniNome,
                    DtFimNome>
Nivel_Membro      = <IdMembro, IdVersao, IdNivel, DtIniNivel,
```

DtFimNivel>
Pai_Membro = <IdMembro, IdVersao, IdMembroPai, DtIniPai, DtFimPai>
Atributo_Membro = <IdMembro, IdVersao, IdAtributo, Valor, DtIniValor, DtFimValor>

onde:
- IdMembro = identificador único do membro;
- IdVersao = identificador único da versão;
- IdNivel = identificador único do nível;
- IdMembroPai = identificador único do membro pai;
- IdAtributo = identificador único do atributo;
- Nome = nome do membro;
- Valor = valor para o atributo do membro;
- DtIniMembro = data de criação do membro;
- DtFimMembro = data de exclusão do membro;
- DtIniNome = primeiro dia de validade do nome para o membro;
- DtFimNome = último dia de validade do nome para o membro;
- DtIniNivel = primeiro dia de validade do nível para o membro;
- DtFimNivel = último dia de validade do nível para o membro;
- DtIniPai = primeiro dia de validade desse pai para o membro;
- DtFimPai = último dia de validade do pai para o membro;
- DtIniValor = primeiro dia de validade do valor para o atributo do membro;
- DtFimValor = último dia de validade do valor para o atributo do membro.

sendo:

- todos os DtIniNome, DtIniNivel, DtIniPai, DtIniValor de um Membro ≥ DtIniMembro deste mesmo membro;
- todos os DtFimNome, DtFimNivel, DtFimPai, DtFimValor de um Membro ≤ DtFimMembro deste mesmo membro;
- todos os DtIniValor de um membro ≥ DtIniNivel do nível do atributo do membro;
- todos os DtFimValor de um atributo de membro ≤ DtFimNivel do mesmo nível do atributo do membro.

6.1.2 Definição dos Valores das Métricas

Definição original:

VB = <M_{id}, $MD_{id\ x}$ $MD_{id\ x}$ ... MD_{id}, VAL >

Nova Definição:

VB = <IdMetrica, $MD_{id\ x}$ $MD_{id\ x}$ ... MD_{id} , IdVersao, DtIniVB, DtFimVB, VAL>

onde:
- IdMetrica = identificador único do métrica;
- IdVersao = identificador único da versão;
- MD_{id} = identificador único de membro de dimensão;
- DtIniVB = primeiro dia de validade do valor;
- DtFimVB = último dia de validade do valor;
- VAL = valor correspondente à interseção dos membros indicados.

sendo:
- DtIniVB ≥ DtIniMembro de todos os membros de dimensão indicados;
- DtFimVB ≤ DtFimMembro de todos os membros de dimensão indicados.

6.1.3 Resumo das definições do modelo

O modelo final com suporte a evolução de esquemas e instâncias possui o conjunto de definições apresentadas a seguir:

$VERSAO_{id}$ = <IdVersao, DataIni, DataFim, Estado>

$Cubo_{id}$ = <IdCubo, DtIniCubo, DtFimCubo>

Nome_Cubo = <IdCubo, NomeCubo, DtIniNome ,DtFimNome>

Caminho_Cubo = <IdCubo, IdCaminho, DtIniCaminho, DtFimCaminho>

Metrica_Cubo = <IdCubo, IdMetrica, DtIniMetrica, DtFimMetrica>

M_{id} = <IdMetrica, DtIniMetrica, DtFimMetrica>

Nome_Metrica = <IdMetrica, NomeMetrica, DtIniNome, DtFimNome>

Dominio_Metrica = <IdMetrica, Tipo_Dominio, DtIniDominio, DtFimDominio>

Funcao_Metrica = <IdMetrica, f_Agreg, DtIniFunc, DtFimFunc>

NivelBase_Metrica = <IdMetrica, IdNivel, DtIniBase, DTFimBase>

D_{id} = <IdDimensao, DtIniDimensao, DtFimDimensao>

Nome_Dimensão = <IdDimensão, NomeDimensão, DtIniNome, DtFimNome>

N_{id} = <IdNivel, IdDimensão, DtIniNivel,

DtFimNivel>

Nome_Nivel = <IdNivel, NomeNivel, DtIniNome, DtFimNome>

A_{id} = <IdAtributo, IdNivel, DtIniAtributo, DtFimAtributo>

Nome_Atributo = <IdAtributo, NomeAtributo, DtIniNome, DtFimNome>

Dominio_Atributo = <IdAtributo, Tipo_dominio, DtIniDominio DtFimDominio>

C_{id} = <IdCaminho, IdNivelFilho, IdNivelPai, DtIniCaminho, DtFimCaminho>

MD_{id} = <IdMembro, IdVersao, DtIniMembro, DtFimMembro>

Nome_Membro = <IdMembro, IdVersao, Nome, DtIniNome, DtFimNome>

Nivel_Membro = <IdMembro, IdVersao, IdNivel, DtIniNivel, DtFimNivel>

Pai_Membro = <IdMembro, IdVersao, IdMembroPai, DtIniPai, DtFimPai>

Atributo_Membro = <IdMembro, IdVersao, IdAtributo, Valor, DtIniValor, DtFimValor>

VB = <IdMetrica, $MD_{id\ x}\ MD_{id\ x}\ \ldots MD_{id}$, IdVersao, DtIniVB, DtFimVB, VAL>

6.2 Controle da Validade das Instâncias

O SGBD do DW deve possuir regras e ações para garantir a integridade e consistência dos períodos de validade das instâncias. Ou seja, o SGBD deve garantir o correto seqüenciamento dos períodos de validade, impedindo os seguintes erros:

- sobreposição de períodos de validade de uma instância.

  Exemplo: [01/01/2000 a 10/02/2002] e [05/02/2002 a 10/02/2003];

- incoerência dos períodos de validade de instâncias dependentes.

  Exemplo: membro de dimensão com período de validade de [01/01/2000 a 10/02/2002] e nome do mesmo membro com período de validade maior que a validade do membro [05/01/1999 a 12/02/2002];

- existência de períodos de validade com data de início de validade indefinida.

Para evitar estas inconsistências é imprescindível a utilização de gatilhos (*triggers*), procedimentos disparados nas operações de gravação do BD (Ceri; Cochrane; Widom, 2000), que façam a verificação da consistências dos períodos de validade relacionados aos registros modificados. Os gatilhos também podem ser utilizados para atribuir os períodos de validade dos dados automaticamente, sem a necessidade da informação explícita das datas de validade pelo usuário. Para isto, seria utilizada a data corrente como parâmetro para a atribuição das datas de início e fim de validade dos dados. A seguir são apresentadas as ações que precisam ser executadas no caso da utilização de gatilhos para o controle automático das validades das instâncias.

6.2.1 Operações de Criação

A criação de uma entidade deve atribuir o valor da data corrente na data de início de validade da entidade. A data final de vigência deve possuir o valor nulo ou indefinido, indicando a existência indeterminada da entidade a partir da data atual.

6.2.2 Operações de Exclusão

A exclusão de uma entidade deve atribuir o valor da data corrente na data de fim de validade da entidade, indicando um período definido de existência.

Caso seja feita a exclusão de um membro de dimensão, todas as outras entidades que fazem referência ao membro deverão ser também excluídas: valores base das métricas, nomes, níveis, relacionamento com membros pais e valores dos atributos descritivos.

É importante ressaltar que o conceito de exclusão utilizado não apaga fisicamente o registro do BD, sendo feita somente a atribuição de término de validade para o registro. Isto é chamado de exclusão lógica do registro.

6.2.3 Operações de Alteração

A alteração de uma entidade realiza o fechamento da validade da versão atual da instância e cria uma nova versão da instância com a modificação realizada. Os passos para esta operação são:

1) criar um novo registro para a instância, com os mesmos dados da instância antes da modificação;
2) atribuir na data de fim de validade da instância criada a data imediatamente anterior a data atual (data de ontem);
3) executar as alterações na instância ;
4) atribuir na data de início de validade da instância a data corrente, deixando a data de fim de validade indefinida.

A figura 6.1 apresenta um exemplo destes passos na alteração do nome do membro SP do nível Estado da dimensão Local .

Dentre as operações de alteração que podem ser realizadas nas instâncias do DW, a alteração do nível de dimensão de um membro merece uma atenção especial. Isto porque ela pode afetar duas outras definições: os atributos do membro e os valores base das métricas (Franconi; Kamble, 2003). Neste caso, todos os valores dos atributos e os todos os valores das métricas que referenciam o membro precisam ser também excluídos logicamente, os valores dos atributos porque as definições dos atribuos entre os níveis são diferentes e os valores das métricas porque o membro deixa de fazer parte do nível base da métrica.

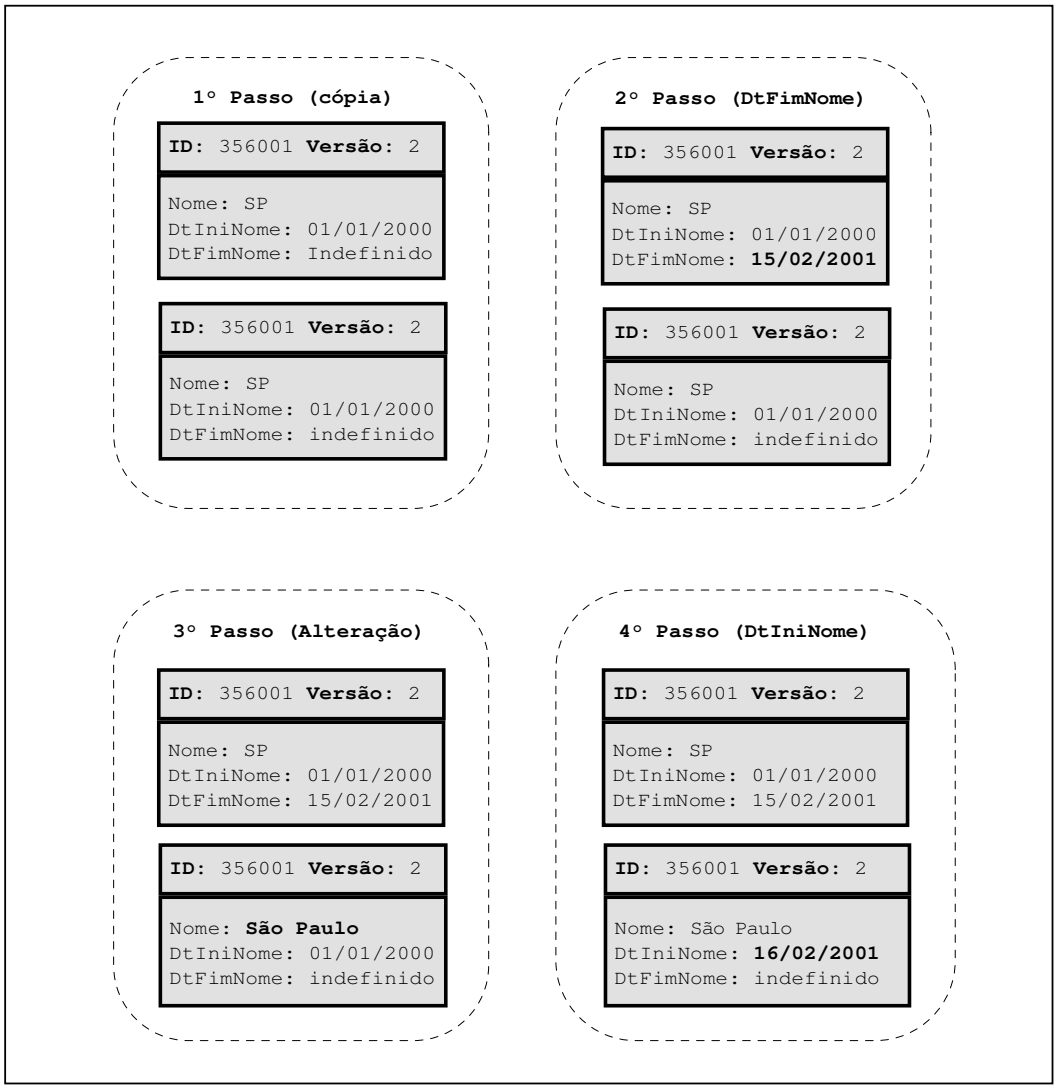

Figura 6.1: Passos da operação de alteração do nome do membro

6.3 Validade das Instâncias na Mudança de Esquema

Quando é feita a operação de ativação de uma versão, os dados populados do esquema antecessor são copiados e adaptados para a nova versão. As datas de validade desses dados não sofrem nenhuma modificação, pois elas representam o valor do dado em um certo intervalo. Assim sendo, somente os novos dados incluídos após a operação de ativação da versão possuirão data de início de validade superior à data de início de vigência desta versão.

Os rótulos temporais das definições do esquema e os rótulos temporais das instâncias possuem naturezas diferentes. Diferente das definições de esquema, onde os tempos de validade retratam sempre o momento da criação, alteração ou exclusão de uma definição do DW (tempo de transação), os tempos de validade das instâncias indicam o momento em que uma informação existiu no mundo real.

As operações de adaptação feitas na ativação da versão não incluem novas informações ao DW, elas apenas realizam transformações em dados pré-existentes.

Desta forma, todas as instâncias criadas ou alteradas para o novo esquema na ativação da versão terão suas datas de validade atribuídas de acordo com as datas de validade de seus dados originais, e não com a data corrente da ativação.

A seguir são apresentados dois tipos de mudança de esquema com adaptação automática das instâncias que ilustram bem os significados das validades das instâncias.

6.3.1 Exclusão de um Nível de Dimensão Intermediário

Um nível de dimensão intermediário é um nível que possui pelo menos um nível superior e pelo menos um nível inferior. No caso de sua exclusão, além da eliminação de todos os membros deste nível, é preciso ajustar todos os relacionamentos de pai-filho entre membros que possuam membros pais pertencentes ao nível excluído.

Este ajuste é feito através da criação de novos relacionamentos de pai-filho, utilizando-se a seguinte regra: para cada relacionamento de pai-filho do esquema antecessor que possua como pai um membro do nível excluído são criados *N* novos relacionamentos de pai-filho, onde os pais são os membros superiores do membro do nível excluído e *N* o número de pais do membro excluído.

Para entender melhor as alterações realizadas veja a exclusão do nível estado realizado na dimensão local , no exemplo mostrado a seguir.

Definições do esquema das versões 01/01/2000 e 05/02/2004:

$VERSAO_1$ = < "1", 01/01/2000, indefinido, "Ativa" >
$VERSAO_2$ = < "2", 05/02/2004, indefinido, "Atual" >
D_{loc} = <"loc", "01/01/2000", indefinido>
Nome_Dimensão = <"loc", "Local" , 01/01/2000, indefinido>
N_{cid} = <"cid", "loc", 01/01/2000, indefinido>
Nome_Nivel = <"cid", "cidade", 01/01/2000, indefinido>
N_{est} = <"est", "loc", 01/01/2000, 04/02/2004>
Nome_Nivel = <"est", "estado", 01/01/2000, 04/02/2004>
N_{pais} = <"pais", "loc", 01/01/2000, indefinido>
Nome_Nivel = <"pais", "país", 01/01/2000, indefinido>
$C_{cid->est}$ = <"cid->est", "cid", "est", 01/01/2000, 04/02/2004>
$C_{est-pais}$ = <"est->pais", "est", "pais", 01/01/2000, 04/02/2004>
$C_{cid-pais}$ = <"cid->pais", "cid", "pais", 05/02/2004, indefinido>

Instâncias da versão 01/01/2000:

MD_{c1} = <"c1", "1", 12/01/2001, indefinido>
Nome_Membro = <"c1", "1", "cidade 1", 12/01/2001, indefinido>
Nivel_Membro = <"c1", "1", "cid", 12/01/2001, indefinido>
Pai_Membro = <"c1", "1", "e2", 12/01/2001, indefinido>
MD_{c2} = <"c2", "1", 12/01/2001, indefinido>
Nome_Membro = <"c2", "1", "cidade 2", 12/01/2001, indefinido>
Nivel_Membro = <"c2", "1", "cid", 12/01/2001, indefinido>
Pai_Membro = <"c2", "1", "e2", 12/01/2001, indefinido>
MD_{e1} = <"e1", "1", 12/01/2001, indefinido>

Nome_Membro = <”e1”, “1”, “estado 1”, 12/01/2001, indefinido>
Nivel_Membro = <”e1”, “1”, “est”, 12/01/2001, indefinido>
Pai_Membro = <”e1”, “1”, “p1”, 12/01/2001, indefinido>
MD_{e2} = <”e2”, “1”, 12/01/2001, indefinido>
Nome_Membro = <”e2”, “1”, “estado 2”, 12/01/2001, indefinido>
Nivel_Membro = <”e2”, “1”, “est”, 12/01/2001, indefinido>
Pai_Membro = <”e2”, “1”, “p1”, 12/01/2001, 01/01/2003>
Pai_Membro = <”e2”, “1”, “p2”, 02/01/2003, indefinido>
MD_{p1} = <”p1”, “1”, 12/01/2001, indefinido>
Nome_Membro = <”p1”, “1”, “pais 1”, 12/01/2001, indefinido>
Nivel_Membro = <”p1”, “1”, “pais”, 12/01/2001, indefinido>
MD_{p2} = <”p2”, “1”, 01/01/2003, indefinido>
Nome_Membro = <”p2”, “1”, “pais 2”, 01/01/2003, indefinido>
Nivel_Membro = <”p2”, “1”, “pais”, 01/01/2003, indefinido>

Instâncias da versão 05/02/2004 após a ativação:

MD_{c1} = <”c1”, “1”, 12/01/2001, indefinido>
Nome_Membro = <”c1”, “1”, “cidade 1”, 12/01/2001, indefinido>
Nivel_Membro = <”c1”, “1”, “cid”, 12/01/2001, indefinido>
Pai_Membro = <”c1”, “1”, “p1”, 12/01/2001, indefinido>
MD_{c2} = <”c2”, “1”, 12/01/2001, indefinido>
Nome_Membro = <”c2”, “1”, “cidade 2”, 12/01/2001, indefinido>
Nivel_Membro = <”c2”, “1”, “cid”, 12/01/2001, indefinido>
Pai_Membro = <”c2”, “1”, “p1”, 12/01/2001, 01/01/2003>
Pai_Membro = <”c2”, “1”, “p2”, 02/01/2003, indefinido>
MD_{p1} = <”p1”, “1”, 12/01/2001, indefinido>
Nome_Membro = <”p1”, “1”, “país 1”, 12/01/2001, indefinido>
Nivel_Membro = <”p1”, “1”, “pais”, 12/01/2001, indefinido>
MD_{p2} = <”p2”, “1”, 01/01/2003, indefinido>
Nome_Membro = <”p2”, “1”, “país 2”, 01/01/2003, indefinido>
Nivel_Membro = <”p2”, “1”, “pais”, 01/01/2003, indefinido>

As datas de validade dos novos relacionamentos pai-filho criados para a versão 05/02/2004 foram determinadas a partir das datas de validade dos relacionamentos pai-filho da versão 01/01/2000. Outra coisa importante ressaltada neste exemplo, é que as datas de início de validade das instâncias podem ser inferiores as datas de início de validade das definições do esquema e da versão, como é o caso do relacionamento pai-filho do país 1 com a cidade 2 (12/01/2001), o caminho cid->pais (05/02/2004) e a versão de 05/02/2004.

Como pode ser visto, uma nova versão do esquema é populada com todos os dados atuais e históricos do esquema antecessor que forem compatíveis com as estruturas do novo esquema, permitindo assim a correta determinação das validades de seus dados.

6.3.2 Exclusão de uma Dimensão

A exclusão de uma dimensão reflete diretamente nos valores base das métricas que utilizam esta dimensão (Franconi; Grandi; Mandreoli, 2000). Todos os valores base dessas métricas precisam ser adaptados para o novo esquema utilizando a seguinte algoritmo:

1) os valores base são agrupados pela composição das chaves de seus membros de dimensão e suas datas de validade, exceto pela chave dos membros da dimensão excluída;
2) para cada grupo de valores base os valores de seus elementos são sumarizados de acordo com a função de agregação da métrica, gerando um novo valor que será gravado como valor base para a métrica no novo esquema.

A seguir é mostrado um exemplo da adaptação realizada para os valores da métrica Nº de itens vendidos apresentada na tabela 2.6 do capítulo dois, no caso da exclusão da dimensão Local.

1) A tabela 6.1 apresenta os agrupamentos de valores base pela chave das dimensões Tempo e Produto, sendo suposto que as datas de validade de todos os valores são iguais para este exemplo.

Tabela 6.1: Agrupamento dos valores base

Dimensão Tempo	**Dimensão Produto**	**Valores Base de cada Grupo**
26/01/2003	N5120	4 (Canoas), 2 (Viamão), 3 (São Paulo), 4 (Campinas)
26/01/2003	M3320	2 (Canoas), 1 (Viamão), 2 (São Paulo), 1 (Campinas)
28/01/2003	N5120	1 (Canoas), 2 (Viamão), 1 (São Paulo), 2 (Campinas)
28/01/2003	M3320	5 (Canoas), 1 (Viamão), 0 (São Paulo), 0 (Campinas)
01/02/2004	N5120	2 (Canoas), 1 (Viamão), 2 (São Paulo), 1 (Campinas)
01/02/2004	M3320	0 (Canoas), 1 (Viamão), 1 (São Paulo), 0 (Campinas)
03/02/2003	N5120	1 (Canoas), 0 (Viamão), 2 (São Paulo), 0 (Campinas)
03/02/2003	M3320	9 (Canoas), 1 (Viamão), 2 (São Paulo), 2 (Campinas)

2) A métrica Nº de itens vendidos utiliza a soma como função de agregação, de modo que será feita uma soma dos valores de cada grupo para formar os valores base para a nova versão do esquema, apresentados na tabela 6.2. As datas de validade dos valores base não sofrem qualquer alteração, pois elas devem indicar o período em que a informação é válida no DW e não a mudança de esquema.

Tabela 6.2: Valores base para o novo esquema

Dimensão Tempo(T)	Dimensão Produto(P)	Função de Agregação	f(TxP)
26/01/2003	N5120	4 + 2 + 3 + 4	13
26/01/2003	M3320	2 + 1 + 2 + 1	6
28/01/2003	N5120	1 + 2 + 1 + 2	6
28/01/2003	M3320	5 + 1+ 0 + 0	6
01/02/2004	N5120	2 + 1+ 2 + 1	6
01/02/2004	M3320	0 + 1 + 1 + 0	2
03/02/2003	N5120	1 + 0 + 2 + 0	2
03/02/2003	M3320	9 + 1 + 2 + 2	14

6.3 Consulta aos Esquemas e Dados do DW

Tendo o histórico das instâncias armazenado, dois tipos de consultas podem ser executadas sobre os dados de uma versão do esquema: consulta aos dados válidos em uma certa data e consulta aos históricos dos dados.

Para a consulta ao dados válidos em um determinado ponto do tempo, basta a adição da data de referência nas consultas para que a pesquisa possa ser realizada através da verificação das instâncias válidas nesta data. Isto é, somente serão válidas as instâncias que possuírem data de início de validade menor ou igual à data de referência e data fim de validade maior ou igual a data de referência (ou data fim de validade igual à indefinido). No caso de uma consulta não informar uma data de referência, o DW pode utilizar a data corrente para a execução das consultas.

Já as consultas sobre os históricos dos dados necessitam de duas datas de referência para indicar o período que se deseja pesquisar. Estas consultas têm como objetivo permitir o conhecimento das modificações efetuadas nos dados do DW. Uma grande utilidade deste tipo de consulta é a descoberta e correção de erros de projeto, como por exemplo a descoberta de atributos de níveis de dimensão com muitas alterações em seus valores ao longo do tempo. Quando isto ocorre, existe uma forte indicação de que este atributo deve ser tratado como uma dimensão a parte. Com o histórico das alterações das instâncias, além de ser possível verificar a quantidade e a freqüência das modificações, é possível recuperar as relações temporais entre os atributos e as métricas possibilitando a população do DW após a criação da nova dimensão.

Tendo o histórico dos esquemas e dos dados é possível realizar os seguintes tipos de consultas:

- consulta aos dados atuais da versão atual do esquema (sem a necessidade de identificação explícita de datas de referência para versão e dados);
- consulta aos dados atuais de qualquer versão de esquema ativa (sem a necessidade de identificação explícita de data de referência para os dados);
- consulta aos dados válidos em uma certa data para qualquer versão do esquema sejam elas ativas ou congeladas;
- consulta ao histórico dos dados de qualquer versão do esquema;

- consulta às definições de qualquer versão do esquema;
- consulta ao histórico das definições (esquemas) do DW.

6.4 Arquitetura de Acesso aos Dados

Um dos pontos mais críticos para o sucesso ou fracasso de um DW é o desempenho das consultas OLAP, que precisam ser realizadas em poucos segundos para permitir uma interação satisfatória para o usuário. Para garantir tempos de resposta aceitáveis, os DW utilizam-se de várias estratégias de otimização, tais como:

- utilização massiva de índices, com a utilização de índices próprios para consultas OLAP e indexação de praticamente todos os campos (Moody; Kortink, 2000);
- pré-cálculo de valores sumarizados das métricas;
- utilização inteligente de memória *cache* e *buffers*, através de heurísticas e algoritmos de inteligência artificial;
- uso de técnicas de particionamento, redundância e distribuição dos dados, levando-se em conta a localidade dos dados (Özsu; Valduriez, 1991);
- utilização de redes de comunicação de alta velocidade.

Outro ponto crítico é a disponibilidade das informações. As cargas incrementais e periódicas dos valores das métricas e as mudanças nas hierarquias dos membros das dimensões costumam afetar a disponibilidade do DW. Mas nada se compara com a mudança do esquema lógico que pode deixar o DW indisponível por um longo período de tempo.

O modelo versionado de DW deste trabalho foi projetado pensando nestes dois fatores, desempenho e disponibilidade, além do objetivo principal, que é o armazenamento do histórico dos dados e das definições. Para evitar as perdas de desempenho em relação ao DW não versionado, cada versão do esquema possui um repositório de dados independente. No caso da disponibilidade, há uma grande vantagem em utilizar o modelo versionado, pois não existe mais a necessidade de se interromper o acesso à versão atual durante a mudança do esquema. A versão atual não sofre alterações físicas em suas definições e dados, não sendo preciso parar o serviço de acesso aos dados pelas aplicações.

A utilização de uma camada intermediária para processamento de consultas entre o DW versionado e as aplicações também é um requisito imprescindível numa implementação real. Esta camada que será chamada de "conversor de consultas" tem como principal tarefa resolver as referências às versões de esquemas e dados, localizando e acessando os dados dos diversos repositórios. O conversor de consultas resolve as consultas que não possuem referência a uma determinada versão de esquema escolhendo a melhor versão segundo seus critérios de seleção. Os principais passos executados pelo conversor de consultas para a resolução de uma consulta OLAP são:

1) identificar a versão do esquema adequada para a consulta, através da consulta a um BD com as definições de esquema de todas as versões;
2) executar a consulta OLAP no repositório de dados da versão escolhida adicionando cláusulas de seleção temporal para selecionar somente os registros válidos para a data de referência dos dados;
3) devolver os resultados da consulta para a aplicação.

A figura 6.2 apresenta as ações executadas pelo conversor de consultas, na ordem em que elas são executadas. No caso de consultas ao histórico de definições de esquema, o conversor de consultas somente precisa acessar o BD de definições.

Para melhorar a performance das consultas sobre os dados atuais é ainda possível criar um repositório a mais para cada versão ativa somente com os dados atuais. Este repositório, além de ser mais enxuto, não precisa ter os atributos e relacionamentos temporais de validade das instâncias, diminuindo assim o número de junções realizadas nessas consultas.

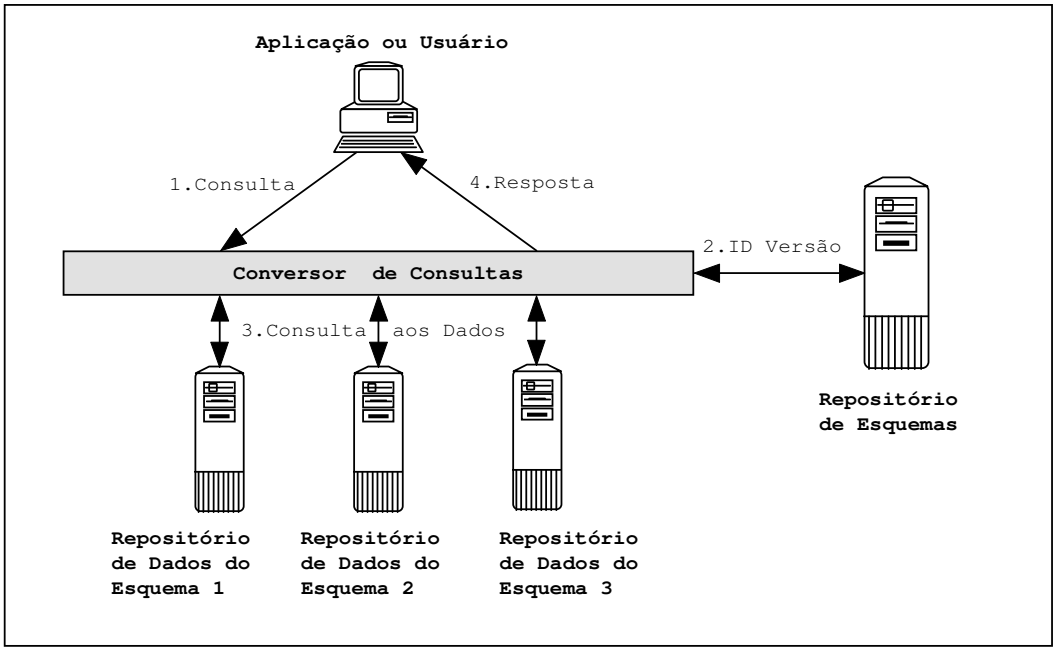

Figura 6.2: Principais passos de uma consulta

6.5 Considerações Finais

O modelo apresentado neste capítulo permite um ótimo acompanhamento e controle da evolução de um DW, pois armazena todas as versões dos esquemas e histórico de dados da vida do DW. A solução de repositórios independentes para cada versão, por sua vez, garante a performance, requisito indispensável a um DW.

Além disso, a utilização de um conversor de consultas como nível intermediário entre as aplicações e o DW permite uma maior compatibilidade, pois as consultas OLAP convencionais, feitas sobre os valores correntes dos dados, não precisam ser modificadas, indicando as validades dos dados. A resolução da versão do esquema e a adição das cláusulas de validade dos dados pode ser feita pelo conversor automaticamente.

Um problema grave que pode, porém, ocorrer no modelo apresentado é a falta de espaço para o armazenamento dos dados das várias versões. Neste caso, existem algumas soluções clássicas, cada qual com suas vantagens e desvantagens, tais como:

- armazenar os dados das versões mais antigas ou menos consultadas em meios de armazenamento secundários, como fitas magnéticas (Inmon, 1997);
- utilizar técnicas de compactação de dados nas versões menos consultadas;
- permitir a exclusão física dos dados das versões “congeladas”.

7 IMPLEMENTAÇÕES DO MODELO

O modelo de gerenciamento de versões para evolução de DW é um modelo conceitual que pode ser implementado sobre outros modelos de DW, permitindo o armazenamento dos históricos de modificações dos esquemas e dados destes outros modelos. Neste capítulo, são apresentadas as adaptações e mapeamentos necessários para permitir o controle de versões de um modelo de dados conceitual e de dois modelos de dados de nível lógico.

7.1 Adaptando o ME/R para o Gerenciamento da Evolução

O modelo ME/R (*Multidimensional Entity Relationship*) proposto por Sapia et al. (1999) é um modelo conceitual para DW que utiliza o paradigma relacional para representar as relações entre os dados do modelo multidimensional. O ME/R foi construído como uma extensão ao modelo Entidade/Relacionamento tradicional, ao qual foram introduzidas algumas regras de modelagem e de controle de integridade necessários para a consistência multidimensional.

Entre as características do ME/R destacam-se a utilização de uma notação gráfica específica para a modelagem dos elementos do DW e a existência de ferramentas gráficas de apoio a modelagem que geram automaticamente *scripts* para a criação das tabelas e relacionamentos em linguagem SQL ou em outras linguagens proprietárias de DWs comerciais (Hahn; Sapia; Blaschka, 2000). Também existem trabalhos sobre como realizar as evoluções de esquema deste modelo, incluindo adaptação automática das instâncias (Blaschka et al., 1999; Blaschka, 2000).

O modelo é composto por três tipos básicos de elementos:

- nodos fato – representa um conjunto de métricas associadas;
- nodos nível – representa os níveis de dimensão;
- nodos atributo – representa as métricas e os atributos dos níveis de dimensão.

Estes elementos formam quatro tipos de relações:

- atributo de um fato – indica que as métricas participantes de um fato;
- atributo de um nível de dimensão – indica os atributos descritivos de um nível;
- dimensão de um fato – indica os níveis base de um fato;
- caminho de classificação – indica os relacionamentos de classificação (pai-filho) entre os níveis de dimensão.

A figura 7.1 apresenta a notação gráfica destes elementos e de seus relacionamentos. Nos caminhos de classificação, a seta sempre aponta para o nível superior.

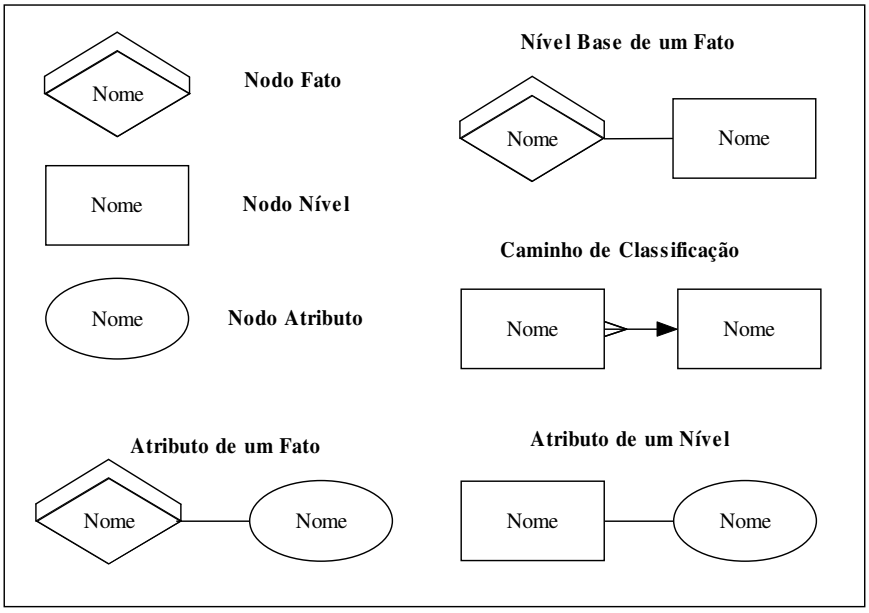

Figura 7.1: Notação gráfica do ME/R

7.1.1 Operações de evolução de esquema no ME/R

Em (Blaschka, 2000) são definidas quatorze operações atômicas de evolução de esquema para o ME/R, que são apresentadas a seguir:

- inclusão de um nível de dimensão;
- exclusão de um nível de dimensão;
- inclusão de um atributo;
- exclusão de um atributo;
- conecção de um atributo a um nível de dimensão;
- desconecção de um atributo de um nível de dimensão;
- conecção de um atributo a um fato;
- desconecção de um atributo de um fato;
- inclusão de um relacionamento de classificação entre dois níveis;
- exclusão de um relacionamento de classificação;
- inclusão de um fato;
- exclusão de um fato;
- inclusão de uma dimensão para um fato;
- exclusão de uma dimensão de um fato.

Além dessas operações, podemos acrescentar também as operações de mudança dos nomes dos fatos, níveis e atributos (métricas e atributos dos níveis).

O ME/R, por ser um modelo conceitual, não apresenta uma forma de representação das instâncias definida e única. Entretanto, Blaschka (2000) apresenta

algumas opções para a implementação das instâncias em SGBDs relacionais. Nesta seção o objetivo é mostrar como pode ser feito o mapeamento das definições do ME/R para o modelo de DW versionado, possibilitando o controle do histórico de modificações dos esquemas.

71.2 Mapeamento dos Elementos e Relacionamentos

Para utilizar o modelo de versões para representar as evoluções de esquema do ME/R são realizados os mapeamentos de equivalência entre os elementos dos dois modelos, mostrados da tabela 7.1.

Tabela 7.1: Mapeamento do ME/R

Modelo ME/R		**Modelo de versões**
nodo fato	=	$Cubo_{id}$ + Nome_Cubo
nodo nível	=	N_{id} {IdDimensão = nulo}
nodo atributo	=	A_{id} {IdNivel = nulo} + Nome_Atributo ou M_{id} {IdMetrica = nulo} + Nome_Metrica
atributo de um fato	=	Metrica_Cubo
atributo de um nível	=	A_{id} + Nome_Atributo
nível base de um fato	=	NivelBase_Metrica
caminho de classificação	=	C_{id}
dimensão	=	D_{id}

A representação gráfica do ME/R não possui uma entidade para representar uma dimensão, mas, segundo as regras do modelo, cada conjunto de níveis ligados por caminhos de classificação forma uma dimensão diferente. Sapia et al. (1999) tratam os atributos dos fatos e os atributos de um nível de forma indiferente (nodo atributo), havendo dois relacionamentos de cardinalidade (1 x N) específicos (atributo de um fato e atributo de um nível) para indicar a funcionalidade do atributo. No modelo de versões este relacionamento já faz parte da definição da métrica ou do atributo descritivo (idmetrica e idatributo).

O elemento fato do ME/R é equivalente à definição de cubo do modelo de versões, sendo os relacionamentos de nodo atributo e nodo fato mapeados em relacionamentos de Metrica_Cubo e os relacionamentos de nível base de um fato mapeados em relacionamentos de NivelBase_Metrica.

Como pode ser visto, o ME/R pode ser representado por um subconjunto do modelo de versões, sendo atribuídos valores iguais para todos os intervalos de validade dos elementos e relacionamentos (o ME/R não possui histórico de evolução). Para permitir que o ME/R armazene e controle múltiplas versões de esquema, permitindo consultas históricas sobre as mudanças das definições ao longo do tempo, basta retirar a restrição de intervalos de validade iguais e utilizar as regras de integridade temporal do modelo de versões.

A seguir é apresentado o subconjunto de definições do modelo de versões necessário para a definição do ME/R versionado:

$VERSAO_{id}$ = <IdVersao, DataIni, DataFim, Estado>

$Cubo_{id}$ = <IdCubo, DtIniCubo, DtFimCubo>

Nome_Cubo = <IdCubo, NomeCubo, DtIniNome ,DtFimNome>

Metrica_Cubo = <IdCubo, IdMetrica, DtIniMetrica, DtFimMetrica>

Caminho_Cubo = <IdCubo, IdCaminho, DtIniCaminho, DtFimCaminho>

M_{id} = <IdMetrica, DtIniMetrica, DtFimMetrica>

Nome_Metrica = <IdMetrica, NomeMetrica, DtIniNome, DtFimNome>

NivelBase_Metrica = <IdMetrica, IdNivel, DtIniBase, DTFimBase>

D_{id} = <IdDimensao, DtIniDimensao, DtFimDimensao>

N_{id} = <IdNivel, IdDimensão, DtIniNivel, DtFimNivel>

Nome_Nivel = <IdNivel, NomeNivel, DtIniNome, DtFimNome>

A_{id} = <IdAtributo, IdNivel, DtIniAtributo, DtFimAtributo>

Nome_Atributo = <IdAtributo, NomeAtributo, DtIniNome, DtFimNome>

C_{id} = <IdCaminho, IdNivelFilho, IdNivelPai, DtIniCaminho, DtFimCaminho>

Observações:

- no ME/R não é utilizado o mesmo conceito de cubo (subconjunto de métricas, com mesmas dimensões e caminhos de classificação) do modelo de versões. No ME/R uma métrica só pode fazer parte de um cubo (nodo fato) e os caminhos de classificação deste cubo são determinados por todos os caminhos de classificação existentes acima do nível base do fato;
- o detalhamento das definições e as regras de integridade temporal entre os elementos e seus relacionamentos são apresentados no capítulo cinco.

7.1.3 Exemplo de Mapeamento do ME/R para o Modelo de Versões

Na figura 7.2 é apresentada a modelagem do DW de vendas de telefones celulares do capítulo dois utilizando as regras e as notações do ME/R. Na figura 7.3 é apresentada uma nova versão do esquema do DW de vendas no ME/R com algumas modificações.

As modificações realizadas na versão 01/01/2000 do esquema do DW que geraram a versão 05/05/2002 são descritas as seguir:

- exclusão do nível de dimensão cidade;
- inclusão do nível de dimensão ano;
- exclusão do atributo de nível de dimensão dia da semana.

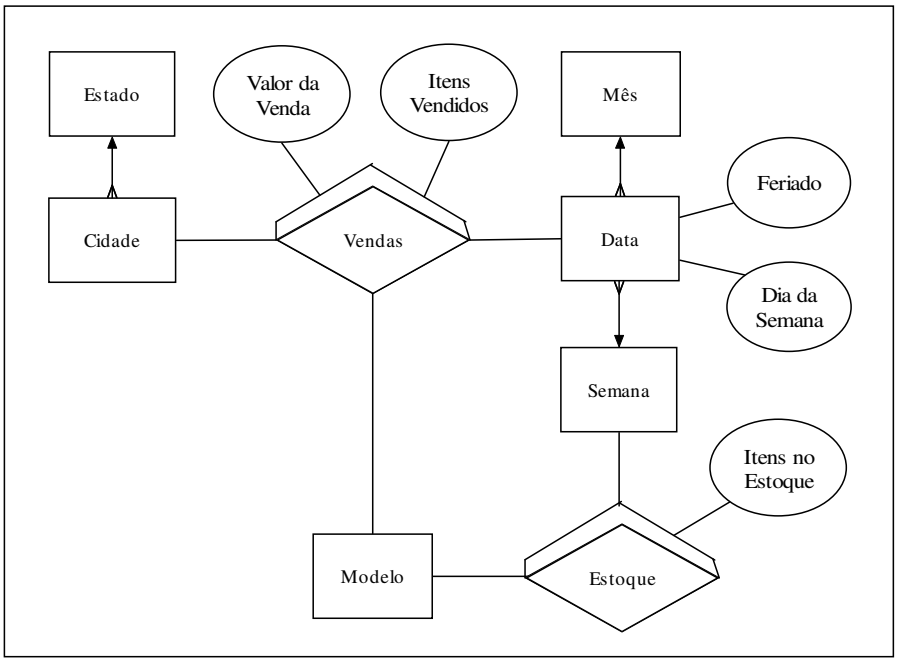

Figura 7.2: Modelagem ME/R para a versão 01/01/2000 do DW

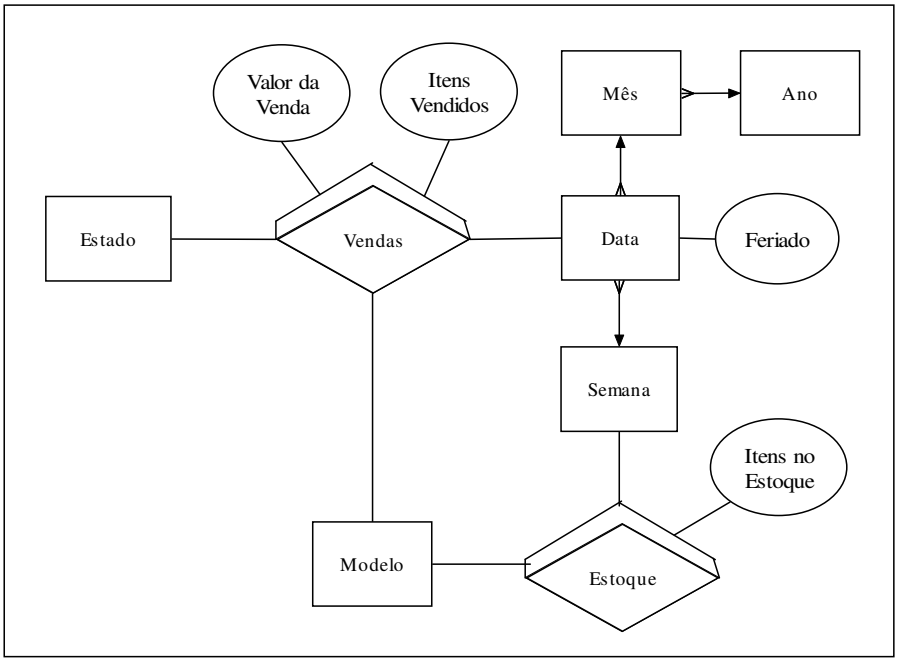

Figura 7.3: Modelagem ME/R para a versão 05/05/2002 do DW

Para representar estes dois esquemas são utilizadas as seguintes construções no modelo de versões:

$VERSAO_1$ = <1,01/01/2000, 04/05/2002, Ativa>
$VERSAO_2$ = <2,05/05/2002, ?, Atual>
$Cubo_{vend}$ = <"vend", 01/01/2000, ?>
$Cubo_{est}$ = <"est", 01/01/2000, ?>
Nome_Cubo = <"vend", "Vendas", 01/01/2000, ?>
Nome_Cubo = <"est", "Estoque", 01/01/2000, ?>
Metrica_Cubo = <"vend", "nro_vend", 01/01/2000, ?>
Metrica_Cubo = <"vend", "valor_vend", 01/01/2000, ?>
Metrica_Cubo = <"est", "nro_est ", 01/01/2000, ?>
Caminho_Cubo = <"vend", "cid->est", 01/01/2000, 04/05/2002>
Caminho_Cubo = <"vend", "est->todos", 01/01/2000, ?>
Caminho_Cubo = <"vend", "dt->sem", 01/01/2000, ?>
Caminho_Cubo = <"vend", "dt->mes", 01/01/2000, ?>
Caminho_Cubo = <"vend", "mes->todos", 01/01/2000, 04/05/2002>
Caminho_Cubo = <"vend", "mes->ano", 05/05/2002,?>
Caminho_Cubo = <"vend", "ano->todos", 05/05/2002,?>
Caminho_Cubo = <"vend", "mod->todos", 01/01/2000,?>
Caminho_Cubo = <"est", "sem->todos", 01/01/2000,?>
Caminho_Cubo = <"est", "mod->todos", 01/01/2000,?>
M_{nro_vend} = <"nro_vend", 01/01/2000, ?>
M_{valor_vend} = <"valor_vend", 01/01/2000, ?>
M_{nro_est} = <"nro_est", 01/01/2000, ?>
Nome_Metrica = <"nro_vend", "Itens Vendidos", 01/01/2000, ?>
Nome_Metrica = <"valor_vend", "Valor da Venda", 01/01/2000, ?>
Nome_Metrica = <"nro_est", "Itens no Estoque", 01/01/2000, ?>
NivelBase_Metrica = <"nro_vend", "cid", 01/01/2000, 04/05/2002>
NivelBase_Metrica = <"nro_vend", "est", 05/05/2002, ?>
NivelBase_Metrica = <"nro_vend", "dt", 01/01/2000, ?>
NivelBase_Metrica = <"nro_vend", "mod", 01/01/2000, ?>
NivelBase_Metrica = <"valor_vend", "cid", 01/01/2000, 04/05/2002>
NivelBase_Metrica = <"valor_vend", "est", 05/05/2002, ?>
NivelBase_Metrica = <"valor_vend", "dt", 01/01/2000, ?>
NivelBase_Metrica = <"valor_vend", "mod", 01/01/2000, ?>
NivelBase_Metrica = <"nro_est", "sem", 01/01/2000, ?>
NivelBase_Metrica = <"nro_est", "mod", 01/01/2000, ?>
D_{local} = <"local", 01/01/2000, ?>
D_{tempo} = <IdDimensao, 01/01/2000, ?>

$D_{produto}$ = <IdDimensao, 01/01/2000, ?>

N_{id} = <"cid", "local", 01/01/2000, 04/05/2002>

N_{id} = <"est", "local", 01/01/2000, ?>

N_{id} = <"dt", "tempo", 01/01/2000, ?>

N_{id} = <"sem", "tempo", 01/01/2000, ?>

N_{id} = <"mes", "tempo", 01/01/2000, ?>

N_{id} = <"ano", "tempo", 05/05/2002, ?>

N_{id} = <"mod", "produto", 01/01/2000, ?>

Nome_Nivel = <"cid", "Cidade", 01/01/2000, 04/05/2002>

Nome_Nivel = <"est", "Estado", 01/01/2000, ?>

Nome_Nivel = <"dt", "Data", 01/01/2000, ?>

Nome_Nivel = <"sem", "Semana", 01/01/2000, ?>

Nome_Nivel = <"ano", "Ano", 05/05/2002, ?>

Nome_Nivel = <"mod", "Modelo", 01/01/2000, ?>

A_{fer} = <"fer", "dt", 01/01/2000, ?>

A_{dia_sem} = <"dia_sem", "dt", 01/01/2000, 04/05/2002>

Nome_Atributo = <"fer", "Feriado", 01/01/2000, ?>

Nome_Atributo = <"dia_sem", "Dia da Semana", 01/01/2000, 04/05/2002>

$C_{cid->est}$ = <"cid->est", "cid", "est", 01/01/2000, 04/05/2002 >

$C_{est->todos}$ = <"est->todos", "est", ⊥, 01/01/2000, ?>

$C_{dt->sem}$ = <"dt->sem", "dt" , "sem", 01/01/2000, ?>

$C_{dt->mes}$ = <"dt->mes", "dt", "mes", 01/01/2000, ?>

$C_{mes->todos}$ = <"mes->todos", "mes", ⊥, 01/01/2000, 04/05/2002>

$C_{mes->ano}$ = <"mes->ano", "mes" , "ano", 05/05/2002, ?>

$C_{sem->todos}$ = <"sem->todos", "sem", ⊥, 01/01/2000, ?>

$C_{ano->todos}$ = <"ano->todos", "ano", ⊥, 05/05/2002, ?>

$C_{mod->todos}$ = <"mod->todos", "mod", ⊥, 01/01/2000, ?>

7.2 Implementação de um Esquema Estrela

A principal regra do modelo estrela é a utilização de dimensões desnormalizadas, ou seja, todos os níveis de uma dimensão são representados em uma única tabela (Tryfona; Busborg; Christiansen, 1999). Neste caso, o modelo de dados não apresenta explicitamente a hierarquia dos níveis, que é controlada somente pelas aplicações OLAP. A grande vantagem deste modelo é a simplicidade e o ganho de performance devido ao pequeno número de junções realizadas nas consultas (Kimball, 1996; Kimball et al., 1998). As principais desvantagens são a redundância provocada pela desnormalização e a falta da representação dos níveis e das hierarquias, que empobrece a semântica multidimensional deste modelo (Trujillo; Palomar; Gómez, 2000).

Os principais elementos e relacionamentos de uma modelo estrela são:

- tabela fato;
- tabelas dimensão;
- métricas de um fato;
- atributos de uma dimensão;
- relacionamentos de um fato com suas dimensões.

A figura 7.4 apresenta um modelo estrela para o fato vendas do DW de venda de telefones celulares do capítulo dois. Uma tabela fato é formada por um conjunto de atributos numéricos (métricas) e um conjunto de atributos chaves para as dimensões (uma chave para cada dimensão do fato). Uma tabela dimensão, por sua vez, é formada por um atributo chave e um conjunto de atributos classificatórios e descritivos. Os atributos classificatórios são os atributos que representam níveis de dimensão, como por exemplo os atributos data, mês e ano. Os atributos descritivos, por outro lado, são informações adicionais sobre um membro, como por exemplo os atributos feriado e dia da semana. No modelo estrela não existe distinção explícita entre atributos classificatórios e descritivos.

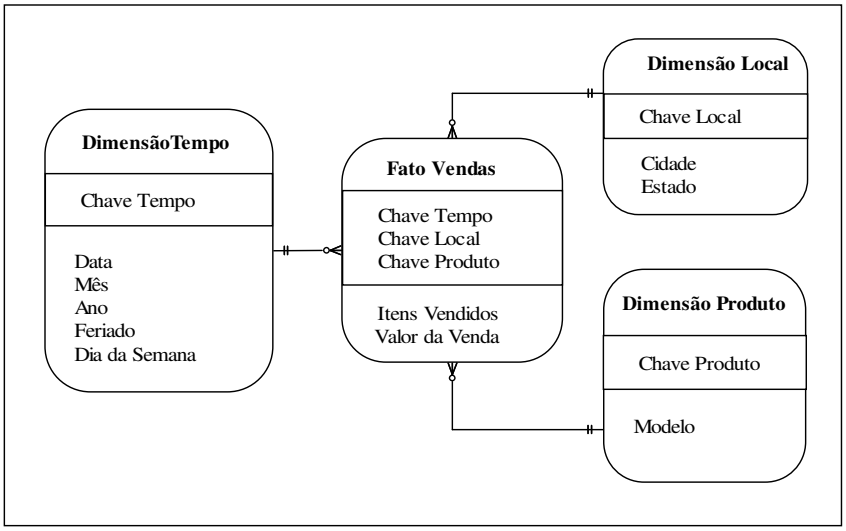

Figura 7.4: Exemplo de Modelo Estrela

Um DW é composto por vários esquemas estrelas relacionados entre si pelo compartilhamento de dimensões, chamado na literatura de constelação (Moody; Kortink, 2000). Entretanto, como nem todos as tabelas fato do DW possuem a mesma granularidade e hierarquia para uma mesma dimensão, é preciso criar múltiplas versões (tabelas) para as dimensões, uma para cada hierarquia diferente. A figura 7.5 apresenta uma constelação formada por dois esquemas estrelas que compartilham a dimensão tempo e produto. Note que a dimensão produto possui a mesma granularidade e hierarquia sendo utilizada uma única tabela, enquanto que a dimensão tempo possui uma granularidade diferente para os dois fatos, sendo preciso criar duas tabelas diferentes.

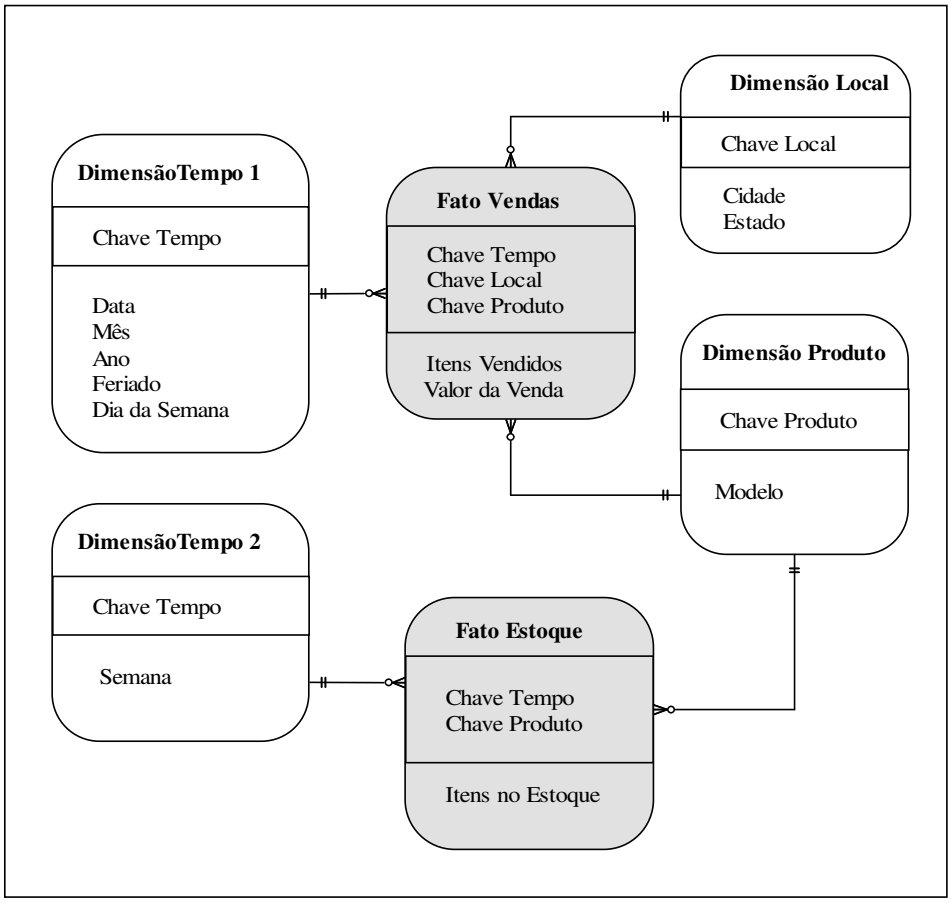

Figura 7.5: Dimensões compartilhas por dois Esquemas Estrela

7.2.1 Mapeamento dos elementos e relacionamentos

Para utilizar o modelo de versões para armazenar o histórico de mudanças do esquema e dos dados de um modelo estrela são feitos os mapeamentos de definições, apresentados na tabela 7.2. As linhas sombreadas indicam as definições das instâncias e as linhas não sombreadas as definições do esquema.

Para implementar o modelo de versões sobre as estruturas do modelo estrela são suprimidas as informações dos relacionamentos hierárquicos entre os níveis de dimensão. Como não existe o conceito de níveis neste modelo de dados, todas as dimensões são representadas por somente um nível do modelo de versões.

Um registro da tabela fato é representado por um conjunto de métricas que possuem o mesmo conjunto de membros de dimensões associados. Um registro da tabela dimensão é representado por um membro com um atributo nome e um conjunto de atributos descritivos.

Tabela 7.2: Mapeamento do Modelo Estrela

Modelo Estrela		**Modelo de versões**
tabela fato	=	$Cubo_{id}$ + Nome_Cubo
tabela dimensão	=	D_{id} + Nome_Dimensão + N_{id}
métrica	=	M_{id} + Nome_Metrica
atributo de dimensão	=	A_{id} + Nome_Atributo
relacionamento fato x dimensão	=	NivelBase_Metrica
registro de um fato	=	{VB_1 , VB_2 ... VB_n}
registro de uma dimensão	=	MD_{id} + Nome_Membro + { $Atributo\_Membro_1$, $Atributo\_Membro_2$ ... $Atributo\_Membro_n$}
valor de uma métrica	=	VB
valor de um atributo de dimensão	=	Atributo_Membro

A seguir é apresentado o subconjunto de definições do modelo de versões necessário para o armazenamento do histórico de esquemas e de dados do modelo estrela:

$VERSAO_{id}$ = <IdVersao, DataIni, DataFim, Estado>

$Cubo_{id}$ = <IdCubo, DtIniCubo, DtFimCubo>

Nome_Cubo = <IdCubo, NomeCubo, DtIniNome ,DtFimNome>

Metrica_Cubo = <IdCubo, IdMetrica, DtIniMetrica, DtFimMetrica>

M_{id} = <IdMetrica, DtIniMetrica, DtFimMetrica>

Nome_Metrica = <IdMetrica, NomeMetrica, DtIniNome, DtFimNome>

Dominio_Metrica = <IdMetrica, Tipo_Dominio, DtIniDominio, DtFimDominio>

NivelBase_Metrica = <IdMetrica, IdNivel, DtIniBase, DTFimBase>

D_{id} = <IdDimensao, DtIniDimensao, DtFimDimensao>

Nome_Dimensão = <IdDimensão, NomeDimensão, DtIniNome, DtFimNome>

N_{id} = <IdNivel, IdDimensão, DtIniNivel, DtFimNivel>

Nome_Nivel = <IdNivel, NomeNivel, DtIniNome, DtFimNome>

A_{id} = <IdAtributo, IdNivel, DtIniAtributo, DtFimAtributo>

Nome_Atributo = <IdAtributo, NomeAtributo, DtIniNome, DtFimNome>

Dominio_Atributo = <IdAtributo, Tipo_dominio, DtIniDominio DtFimDominio>

MD_{id} = <IdMembro, IdVersao, DtIniMembro, DtFimMembro>

Nome_Membro = <IdMembro, IdVersao, Nome, DtIniNome, DtFimNome>

Nivel_Membro = <IdMembro, IdVersao, IdNivel, DtIniNivel, DtFimNivel>

Atributo_Membro = <IdMembro, IdVersao, IdAtributo, Valor, DtIniValor, DtFimValor>

VB = <IdMetrica, $MD_{id\ x}\ MD_{id\ x}\ \ldots MD_{id}$, IdVersao, DtIniVB, DtFimVB, VAL>

7.3 Implementação de um Esquema SnowFlake

O modelo *SnowFlake* (floco de neve) possui os níveis das dimensões representados de forma explícita e normalizada. Suas principais vantagens são a menor redundância dos dados e a consistência semântica da hierarquia de níveis no próprio modelo de dados. A maioria dos autores sugere a utilização do modelo *SnowFlake* para a modelagem de DWs, devido a semântica mais rica e à maior integração, e a utilização do modelo estrela para a modelagem de *Data Marts*, devido à melhor performance das consultas. Além disso, é possível fazer um mapeamento direto de um modelo *SnowFlake* (normalizado) para um modelo Estrela (desnormalizado). Já o mapeamento de um modelo estrela para um modelo *SnowFlake* (operação inversa) não é uma tarefa fácil.

Os elementos e relacionamentos que compõem um esquema *SnowFlake* são:

- tabela fato;
- tabela nível de dimensão;
- métricas de um fato;
- atributos de níveis de dimensão;
- relacionamento de um fato com seus níveis base de dimensão;
- relacionamentos de hierarquia (pai-filho) entre níveis de dimensão.

Ao contrário do modelo estrela, no modelo *SnowFlake* não é preciso criar novas versões das tabelas dimensões para fatos que possuem granularidades diferentes de uma mesma dimensão. Cada tabela de nível de dimensão possui seus próprios atributos descritivos e uma chave única, que pode ser referenciada pelas tabelas fatos.

7.3.1 Mapeamento dos elementos e relacionamentos

Na tabela 7.3 são apresentados os mapeamentos necessários para a implementação do modelo de versões sobre o modelo *SnowFlake*.

No modelo *SnowFlake* existem as tabelas fatos e as tabelas níveis de dimensões, que são representadas pelos cubos e níveis do modelo de versões, respectivamente. O relacionamento do fato com suas dimensões é representado pelos níveis base de suas métricas. O relacionamento de hierarquia entre os níveis é representado pelos caminhos de classificação (relacionamento de pai-filho).

Tabela 7.3: Mapeamento do Modelo SnowFlake

Modelo *SnowFlake*		**Modelo de versões**
tabela fato	=	$Cubo_{id}$ + Nome_Cubo
tabela nível de dimensão	=	N_{id} + Nome_Nivel
métrica	=	M_{id} + Nome_Metrica
atributo de nível de dimensão	=	A_{id} + Nome_Atributo
relacionamento fato x dimensão	=	NivelBase_Metrica
relacionamento hierárquico entre dois níveis	=	C_{id}
registro de um fato	=	{VB_1 , VB_2 ... VB_n}
registro de um nível de dimensão	=	MD_{id} + Nome_Membro + { $Atributo\_Membro_1$, $Atributo\_Membro_2$ ... $Atributo\_Membro_n$}
valor de uma métrica	=	VB
valor de um atributo de nível de dimensão	=	Atributo_Membro
relacionamento de pai-filho entre registos de níveis	=	Pai_Membro

Na implementação física deste modelo, geralmente, cada nível é implementado com três tipos de atributos:

- chave do nível;
- chaves de todos os níveis superiores;
- atributos descritivos do nível.

Uma dimensão é caracterizada por um relacionamento com uma tabela fato e por todos os níveis ligados entre si pelos relacionamentos de pai-filho. O nome da dimensão costuma ser deduzido dos nomes dos níveis base, não havendo a rigor uma estrutura física para seu armazenamento.

A seguir é apresentado o subconjunto de definições do modelo de versões necessário para o armazenamento do histórico dos esquemas e dos dados do modelo estrela:

$VERSAO_{id}$ = <IdVersao, DataIni, DataFim, Estado>

$Cubo_{id}$ = <IdCubo, DtIniCubo, DtFimCubo>

Nome_Cubo = <IdCubo, NomeCubo, DtIniNome ,DtFimNome>

Caminho_Cubo = <IdCubo, IdCaminho, DtIniCaminho, DtFimCaminho>

Metrica_Cubo = <IdCubo, IdMetrica, DtIniMetrica, DtFimMetrica>

M_{id} = <IdMetrica, DtIniMetrica, DtFimMetrica>

Nome_Metrica = <IdMetrica, NomeMetrica, DtIniNome, DtFimNome>

```
   Dominio_Metrica = <IdMetrica, Tipo_Dominio, DtIniDominio,
                      DtFimDominio>
 NivelBase_Metrica = <IdMetrica, IdNivel, DtIniBase, DTFimBase>
               Did = <IdDimensao, DtIniDimensao, DtFimDimensao>
               Nid = <IdNivel, IdDimensão, DtIniNivel,
                      DtFimNivel>
        Nome_Nivel = <IdNivel, NomeNivel, DtIniNome, DtFimNome>
               Aid = <IdAtributo, IdNivel, DtIniAtributo,
                      DtFimAtributo>
    Nome_Atributo = <IdAtributo, NomeAtributo, DtIniNome,
                      DtFimNome>
 Dominio_Atributo = <IdAtributo, Tipo_dominio, DtIniDominio
                      DtFimDominio>
               Cid = <IdCaminho, IdNivelFilho, IdNivelPai,
                      DtIniCaminho, DtFimCaminho>
              MDid = <IdMembro, IdVersao, DtIniMembro,
                      DtFimMembro>
      Nome_Membro = <IdMembro, IdVersao, Nome, DtIniNome,
                      DtFimNome>
     Nivel_Membro = <IdMembro, IdVersao, IdNivel, DtIniNivel,
                      DtFimNivel>
       Pai_Membro = <IdMembro, IdVersao, IdMembroPai,
                      DtIniPai, DtFimPai>
  Atributo_Membro = <IdMembro, IdVersao, IdAtributo, Valor,
                      DtIniValor, DtFimValor>
               VB = <IdMetrica, MDid x MDid x ...MDid , IdVersao,
                      DtIniVB, DtFimVB, VAL>
```

Praticamente todas as construções do modelo de versões são utilizadas no modelo *SnowFlake*, com exceção da definição das funções de agregação das métricas e da utilização de nomes para caracterizar uma dimensão. Estas informações costumam ser definidas e cadastradas somente nas aplicações.

7.4 Considerações Finais

O modelo de versões pode ser aplicado a vários tipos de modelos de DW e DMs. Neste capítulo foram apresentados os mapeamentos necessários para três tipos de modelos:

- modelo ME/R (Sapia et al., 1999; Blaschka et al., 1999; Blaschka, 2000);
- modelo Estrela (Kimball, 1996; Kimball et al., 1998);
- modelo *SnowFlake*.

Estes três modelos foram escolhidos para o detalhamento dos mapeamentos por dois motivos principais: por serem muito referenciados na literatura e por possuírem implementação prática em produtos comerciais.

8 CONCLUSÕES

Data Warehouses são sistemas criados para representar a evolução dos negócios de uma organização permitindo sua análise e entendimento. Mas para que este objetivo seja plenamente alcançado é importante a compreensão da história das informações contidas no DW. Ou seja, é preciso armazenar todos os valores dos dados e suas respectivas versões de esquema para permitir a consulta destes dados no contexto do ambiente do tempo pesquisado.

O modelo com suporte a evolução de esquemas e dados auxilia os DBAs e usuários finais a compreender melhor os dados do DW, bem como a sua evolução. Tendo o histórico completo dos esquemas e dados armazenados, é possível rastrear todos os estados e modificações do DW, permitindo consultas sofisticadas para os mais diferentes propósitos, tais como:

- recuperação dos dados válidos em uma certo momento;
- auditoria dos dados;
- estudo da evolução do esquema através da comparação das definições de um ou mais esquemas;
- estudo da dinâmica das dimensões e dos relacionamentos temporais entre os dados;
- análise da taxa de alteração dos membros das dimensões, visando correções do modelo conceitual para minimizar a freqüência destas alterações.

O trabalho também apresenta as principais regras de integridade e consistência semântica dos elementos de um DW. Os tipos de operações de alteração do esquema e de alteração das instâncias são discutido em detalhes, possibilitando um entendimento global das relações entre os dados e metadados de um DW. A preocupação com a adaptação das instâncias no momento da ativação de um novo esquema é outro ponto forte do gerenciamento do modelo.

O modelo também trata, com o devido cuidado, três pontos críticos dos ambientes de DW que são:

- a performance das consultas – através da utilização de repositórios diferentes para cada versão do esquema em vez de um único repositório consolidado;
- a disponibilidade dos dados – através da possibilidade de existência de múltiplas versões de esquema ativas;
- a compatibilidade do DW com suas aplicações – através do controle de acesso às múltiplas versões ativas, realizado pelo conversor de consultas.

A possibilidade de existência de duas ou mais versões de esquema ativas simultaneamente permite uma adaptação posterior e gradual das aplicações para o

esquema atual, tornando assim a modificação do esquema uma tarefa mais simples, rápida e segura.

Em resumo, este trabalho apresentou um modelo de dados consistente e prático para o gerenciamento e consulta das modificações de esquema e dos dados de um DW.

8.1 Futuros trabalhos

A especificação de um modelo de dados e o seu gerenciamento representa somente a ponta do enorme *iceberg* que é a evolução de um DW, de modo que ainda existem muitos pontos para serem trabalhados e aperfeiçoados, tais como:

- detalhamento do funcionamento do conversor de consultas (camada intermediária entre o DW e as aplicações) , através da definição de regras para a escolha da versão de esquema e algoritmos para a resolução das consultas;
- definição de estratégias e alternativas para a adaptação automática das instâncias;
- criação de mecanismos avançados de consultas aos históricos, como por exemplo, a definição de uma linguagem de consulta específica para consulta a dados históricos (com construções temporais e multidimensionais de alto nível), ou o estudo de otimizações para a consulta dos históricos dos dados e das modificações dos esquemas;
- especialização do modelo para sua utilização em ferramentas comerciais, uma vez que cada SGBD possui limitações e características próprias que exigem certas adaptações ao modelo genérico apresentado.

REFERÊNCIAS

ABELLÓ, A.; SAMOS, J.; SALTOR, F. A Framework for the Classification and Description of Multidimensional Data Models. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, DEXA, 12., 2001, Munich, Germany. **Proceedings...** Berlin: Springer, 2001. p. 668-677.

ABELLÓ, A.; MARTÍN, C. A Bitemporal Storage Structure for a Corporate Data Warehouse. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, ICEIS, 5., 2003, Angers, France. **Proceedings...** Dordrecht, Netherlands: Kluwer Academic Publishers, 2003. p. 177-183.

AGRAWAL, A.; GUPTA, A.; SARAWAGI, S. Modeling Multidimensional Databases. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 13., 1997, Birmingham, U.K. **Proceedings...** Los Alamitos: IEEE Computer Society 1997. p. 232-243.

BARBIERI, C. **BI – Business Intelligence – Modelagem & Tecnologia.** Rio de Janeiro: Axcel Books, 2001. 424 p.

BLASCHKA, M.; SAPIA, C.; HÖFLING, G. Finding your way through multidimensional data models. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, DEXA, 1998. **Proceedings...** Berlin: Springer, 1998. p. 198-203.

BLASCHKA, M.; SAPIA, C.; HÖFLING, G. On Schema Evolution in Multidimensional Databases. In: INTERNATIONAL CONFERENCE ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY, DAWAK, 1., 1999, Florence, Italy. **Proceedings...** Berlin: Springer, 1999. p.153-164.

BLASCHKA, M. **FIESTA**: a Framework for Schema Evolution in Multidimensional Databases. 2000. Ph.D. Thesis - Technische Universität München, Germany.

BODY, M.; MIQUEL, M.; BÉDARD, Y.; TCHOUNIKINE. A. Handling Evolutions in Multidimensional Structures. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 2003, Bangalore, India. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p. 581- 592

BODY, M.; MIQUEL, M., BÉDARD; Y., TCHOUNIKINE, A. A Multidimensional and Multiversion Structure for OLAP Applications. In: INTERNATIONAL WORKSHOP ON DATA WAREHOUSING AND OLAP, DOLAP, 5., 2002, McLean, USA. **Proceedings...** New York: ACM Press, 2002.

BRUCKNER, R. M.; LIST, B.; SCHIEFER J.; TJOA A. M. Modeling Temporal Consistency in Data Warehouse. In: WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, DEXA, 12., 2001, Munich, Germany. **Proceedings**... Los Alamitos: IEEE CS Press, 2001. p. 467-471.

BRUCKNER R. M.; TJOA A. M. Capturing Delays and Valid Times in Data Warehouses – Towards Timely Consistent Analyses. **International Journal of Intelligent Information Systems**, Netherland, v. 19, n. 2. p. 169 –190, 2002.

CABIBBO, L.; TORLONI, R. A Logical approach to multidimensional databases. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, EDBT, 6., 1998, Valencia, Spain. **Proceedings...** Berlin: Springer, 1998. p. 183-197.

CHAMONI, P.; STOCK, S. Temporal Structures in Data Warehousing. In: INTERNATIONAL CONFERENCE ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY, DAWAK, 1999, Florence, Italy. **Proceedings...** Berlin: Springer, 1999. p.353-358.

CERI S.; COCHRANE R J.; WIDOM J. Practical Applications of Triggers and Constraints: Success and Lingering Issues. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES, VLDB, 26., 2000, Cairo Egypt. **Proceedings...** San Francisco, USA: Morgan Kaufmann, 2000. p. 254 - 262.

DEVLIN, B. Managing Time in the Data Warehouse. **InfoDB**, [S. l.], v. 11, n. 1, p. 7-11, 1997.

EDELWEISS, N. Banco de Dados Temporais: teoria e prática. In. JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, JAI, 17., 1998, Belo Horizonte, Brasil. **Anais...** Belo Horizonte: SBC, 1998. p. 225-282.

EDER, J.; KONCILIA, C. Changes of Dimension Data in Temporal Data Warehouses. In: INTERNATIONAL CONFERENCE ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY, DAWAK, 3., 2001, Munich, Germany. **Proceedings...** Berlin: Springer, 2002. p. 284-293.

EDER, J.; KONCILIA, C.; MORZY, T. The COMET Metamodel for Temporal Data Warehouses. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, CAiSE, 14., 2002, Toronto, Canada. **Proceedings...** Berlin: Springer, 2002. p. 83-99.

EDER, J.; KONCILIA, C.; MITSCHE, D. Automatic detection of structural changes in Data Warehouses. In: INTERNATIONAL CONFERENCE ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY, DAWAK, 5., 2003, Prague, Czech Republic. **Proceedings...** Berlin: Springer 2003. p. 119-128. (Lecture Notes in Computer Science, 2737).

FRANCONI, E.; GRANDI, F.; MANDREOLI, F. A Semantic Approach for Schema Evolution and Versioning in Object-Oriented Databases. In: INTERNATIONAL CONFERENCE ON RULES AND OBJECTS IN DATABASES, DOOD, 6., 2000, London, UK. **Proceedings...** Berlin: Springer, 2002. p. 1048 –1062.

FRANCONI, E.; KAMBLE, A. The GMD Data Model for Multidimensional Information: A Brief Introduction. In: INTERNATIONAL CONFERENCE ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY, DAWAK, 5., 2003, Prague, Czech Republic. **Proceedings...** Berlin: Springer, 2003. p.55-65.

GALANTE, R.; EDELWEISS, N.; SANTOS, C. S. dos. Change Management for a Temporal Versioned Object-Oriented Database. In: INTERNATIONAL WORKSHOP ON EVOLUTION AND CHANGE IN DATA MANAGEMENT, ECDM, 2002, Tampere, Finland. **Proceedings...** Berlin: Springer-Verlag, 2002. p. 1-12.

GOLFARELLI, M.; MAIO, D.; RIZZI, S. The Dimensional Fact model: a Conceptual Model for Data Warehouses. **International Journal of Cooperative Information Systems**, Netherlands, v.7, n.2&3, p. 215-247, 1998.

GOPALKRISHNAN, V.; LI, Q.; KARLAPALEM, K. Star/snow-flake schema driven object-relational data warehouse design and query processing strategies. In: INTERNATIONAL WORKSHOP ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY, DAWAK, 1., 1999, Florence, Italy. **Proceedings...** Berlin: Springer, 1999. p. 11-22.

GÜNZEL, H. Versioning for Data Warehouse - the TEMPS approach. In: WORKSHOP ON TEMPORAL ASPECTS, 2000. **Proceedings...** [S. l.]: Univesitat Erlangen-Nürnberg, 2000.

GYSSENS, M.; LAKSHMANAN, V. S. A Foundation for multi-dimensional databases. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES, VLDB, 23., 1997, San Francisco, USA. **Proceedings...** San Francisco: Morgan Kaufmann, 1997. p. 106-115.

HAHN K.; SAPIA C.; BLASCHKA M. Automatically Generating OLAP Schemata from Conceptual Graphical Models. In: INTERNATIONAL WORKSHOP ON DATA WAREHOUSING AND OLAP, 3., 2000, McLean, USA. **Proceedings...** New York: ACM Press, 2000. p. 9-16.

HURTADO, C.; MENDELZON, A.; VAISMAN, A. Updating OLAP Dimensions. In: INTERNATIONAL WORKSHOP ON DATA WAREHOUSING AND OLAP, 3., 1999, Kansas City, USA. **Proceedings**... New York, ACM Press, 1999. p.60-66.

HURTADO, C.; MENDELZON, A.; VAISMAN, A. Maintaining Data Cubes Under Dimension Updates. In: IEEE INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 15., 1999, Sydney, Australia. **Proceedings...** Los Alamitos: IEEE Computer Society, 1999. p. 346-355.

HURTADO C.; MENDELZON A. O. OLAP Dimension Constraints. In: SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS, 2002, Madison, USA. **Proceedings...** New York: ACM, 2002. p. 169-179.

HÜSEMANN, B.; LECHTENBÖRGER, J.; VOSSEN, G. Conceptual Data Warehouse design. In: INTERNATIONAL WORKSHOP ON DESIGN AND MANAGEMENT OF DATA WAREHOUSES, DMDW, 2., 2000, Stockholm, Sweden. **Proceedings...** Aachen, Germany: RWTH, 2000. p. 6.1–6.11.

INMON, W. H. **Como Construir o Data Warehouse**. Rio de Janeiro: Campus, 1997.

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

JENSEN, C. S.; DYRESON, C. E. The consensus glossary of temporal database concepts. In: ETZION, O.; JAJODIAS, S.; SRIPADA, S. (Ed.). Temporal Databases: Temporal Databases: research and practice. Berlin: Springer, 1998. p. 367-405.

KIMBALL, R. **The Data Warehouse Toolkit**. New York, USA: John Willey & Sons, 1996.

KIMBALL, R.; REEVES, L.; ROSS, M.; THORNTHWAITE, W. **The Data Warehouse Lifecycle Toolkit – Expert Methods for Designing, Developing and Deploying Data Warehouses**. New York, USA: John Willey & Sons, 1998. 771 p.

KOOLLER, A.; RUNDENSTEINER, E.; HACHEM, N**. Integrating the Rewriting and Ranking Phases of View Synchronization.** Worcester, USA: Worcester Polytechnic Institute, 1998. (Technical Report WPI-CS-TR-98-23).

LEHNER, W. Modeling Large Scale OLAP Scenarios. In: INTERNATIONAL. CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, EDBT, 6., 1998, Valencia, Spain. **Proceedings...** Berlin: Springer, 1998. p.153-167.

MACHADO, F. N. R. **Projeto de Data Warehouse:** uma visão multidimensional. São Paulo: Érica, 2000. 248 p.

MANGISENGI, O.; TJOA, A.; WAGNER, R. Multidimensional Modeling Approaches for OLAP Based on Extended Relational Concepts. In: INTERNATIONAL DATABASE CONFERENCE ON HETEROGENEOUS AND INTERNET DATABASES, 9., 1999, Hong Kong. **Proceedings...** [S. l. : s. n.], 1999.

MARTIN, C.; ABELLÓ, A. A Temporal Study of Data Sources to Load a Corporate Data Warehouse. In: INTERNATIONAL CONFERENCE ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY, DAWAK, 5., 2003, Prague, Czech Republic. **Proceedings...** Berlin: Springer, 2003. p.109-118.

MENDELZON, A.; VAISMAN, A. Temporal queries in OLAP. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES, VLDB, 26., 2000, Cairo, Egypt. **Proceedings...** San Francisco: Morgan Kaufmann, 2000. p. 254 - 262.

MOODY, D.; KORTINK, M. From enterprise models to dimensional models: A methodology for data warehouse and data mart design. In: INTERNATIONAL WORKSHOP ON DESIGN AND MANAGEMENT OF DATA WAREHOUSES, DMDW, 2., 2000, Stockholm, Sweden, **Proceedings...** Aachen, Germany: RWTH, 2000. p. 5.1 - 5.12

MORO, M. **Modelo Temporal de Versões**. 2001. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

MORZY, T.; WREMBEL, R. Modeling a multidimensional Data Warehouse: A Formal Approach. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, ICEIS, 5., 2003, France. **Proceedings...** Dordrecht, Netherlands: Kluwer Academic Publishers, 2003. p. 112-119

NGUYEN, T. B.; TJOA, A. M.; WAGNER, R. An object oriented multidimensional data model for OLAP. In: INTERNATIONAL CONFERENCE ON WEB-AGE INFORMATION MANAGEMENT, WAIM, 1., 2000, Shanghai, China. **Proceedings...** Berlin: Springer, 2000. p. 69-82.

NICA, A. **View Evolution Support for Information Integration Systems over Dynamic Distributed Information Spaces**. 1999. Ph.D. thesis - University of Michigan, USA.

ÖZSU, M.; VALDURIEZ, T. **Principles of Distributes Database Systems.** New Jersey, USA: Prentice Hall, 1991.

PEDERSEN, T.; JENSEN, C.; DYRESON, C. A foundation for capturing and querying complex multidimensional data. **Information Systems**, Oxford, v. 26, n.5, p. 383-423, 2001.

PEREIRA W. **Data Warehouse**. 1999. Trabalho Individual (Mestrado em Informática) – Instituto de Informática – PUC, Porto Alegre.

RODDICK, J. F. **A Model for Temporal Inductive Inference and Schema Evolution in Relational Database Systems**. 1994. Ph.D Thesis - Departament of Computer Schience and Computer Enginnering, La Trobe University, Melbourne, Australia.

SAPIA, C.; BLASCHKA, M.; HÖFLING G.; DINTER B. Extending the E/R model for the multidimensional paradigm. In. INTERNATIONAL WORKSHOP ON DATA WAREHOUSE AND DATA MINING, DWDM, 1998. **Proceedings...** Berlin: Springer 1998. p. 105-116.

TRUJILLO, J.; PALOMAR, M.; GÓMEZ, J. Applying Object-Oriented Conceptual Modeling Techniques to the Design of Multidimensional Databases and OLAP applications. In: INTERNATIONAL CONFERENCE ON WEB-AGE INFORMATION MANAGEMENT, WAIM, 1., 2000, Shanghai, China. **Proceedings...** Berlin: Springer, 2000. p. 83-94.

TRYFONA, N.; BUSBORG, F. ; CHRISTIANSEN, J. StarER: A conceptual model for data warehouse design. In: ACM INTERNATIONAL WORKSHOP ON DATA WAREHOUSING AND OLAP, DOLAP, 2., 1999, Kansas City, USA. **Proceedings...** New York: ACM Press 1999. p. 374-387.

VAISMAN, A. **Updates View Maintenance and Time Management on Multidimensional Databases.** 2001 Ph.D. Thesis - Universidad de Buenos Aires, Buenos Aires.

VAISMAN, A.; MENDELZON, A. A Temporal Query Language for OLAP: Implementation and a Case Study. In: DATABASE PROGRAMMING LANGUAGES, INTERNATIONAL WORKSHOP, DBPL, 8., 2001, Frascati, Italy. **Proceedings...** Berlin: Springer, 2002. p. 78-96.

VAISMAN, A.; MENDELZON, A.; RUARO, W.; CYMERMAN, S. Supporting Dimension Updates in an OLAP Server. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, CAiSE, 2002. **Proceedings...** Berlin: Springer 2002. p. 67-82.

VASSILIADIS P. **Data Warehouse Modeling an Quality Issues**. 2000. Ph.D. thesis - Departament of Electrical and Computer Enginnering, National Technical University of Athens, Greece.