Universidade Federal de Santa Catarina
Programa de Pós-Graduação em
Engenharia da Produção

Sueli de Fátima Poppi Borba

METODOLOGIA PARA IMPLANTAÇÃO DE MODELOS MULTIDIMENSIONAIS EM BANCO DE DADOS ORIENTADO A OBJETOS

Tese de Doutorado

Florianópolis
2006

Sueli de Fátima Poppi Borba

METODOLOGIA PARA IMPLANTAÇÃO DE MODELOS MULTIDIMENSIONAIS EM BANCO DE DADOS ORIENTADO A OBJETOS

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina como requisito parcial para obtenção do grau de Doutor em Engenharia de Produção

Orientador: Prof. Aran Bey Tcholakian Morales, Dr.

Florianópolis
2006

B726m Borba, Sueli de Fátima Poppi
Metodologia para implantação de modelos multidimensionais em banco de dados orientado a objetos / Sueli de Fátima Poppi Borba ; orientador Aran Bey Tcholakian Morales. – Florianópolis, 2006.
228 f.

Tese (Doutorado) – Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Produção, 2006.

Inclui bibliografia

1. Banco de dados. 2. Modelagem multidimensional. 3. Programação orientada a objetos (Computação). 4. UML (Computação).
I. Morales, Aran Bey Tcholakian.
II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Produção. IV. Título.

CDU:658.5

Catalogação na fonte por: Onélia Silva Guimarães CRB-14/071

Sueli de Fátima Poppi Borba

METODOLOGIA PARA IMPLANTAÇÃO DE MODELOS MULTIDIMENSIONAIS EM BANCO DE DADOS ORIENTADO A OBJETOS

Esta tese foi julgada e aprovada para a obtenção do grau de doutora em Engenharia da Produção no Programa de Pós-graduação em Engenharia da Produção da Universidade Federal de Santa Catarina.

Florianópolis, 22 de junho de 2006.

_____

Prof. Edson Pacheco Paladini, Dr.
Coordenador do Programa

Banca examinadora

Prof. Aran Bey Tcholakian Morales, Dr. Universidade Federal de Santa Catarina Orientador	Profa. Maria Madalena Dias, Dr. Universidade Estadual de Maringá Examinador externo
Prof. Pedro Paulo da Silva Ayrosa, Dr. Universidade Estadual de Londrina Examinador externo	Prof. Rosvelter Coelho da Costa, Dr. Universidade Federal de Santa Catarina Examinador externo
Prof. José Leomar Todesco, Dr. Universidade Federal de Santa Catarina Examinador	Prof. Vinícius Medina Kern, Dr. Universidade Federal de Santa Catarina Moderador

Aos meus filhos,
Gabriele, Iago e Lucas,
fontes de força e inspiração.

Agradecimentos

A Deus, pela minha existência.
A minha família, que me apoiou em todos os momentos da realização desta pesquisa.
Ao Prof. Dr. Edson Pacheco Paladini, coordenador do programa, por ter concedido a oportunidade de ingressar no programa.
Agradecimentos especiais ao professor orientador, Prof. Dr. Aran Bey Tcholakian Morales, conduzindo e incentivando-me com paciência e competência, na busca de novos conhecimentos.
Aos dirigentes e professores da Unipar Cianorte, pelo apoio e incentivo.
A todos aqueles que direta ou indiretamente colaboraram com o desenvolvimento deste trabalho.

Resumo

BORBA, Sueli de Fátima Poppi. **Metodologia para implantação de modelos multidimensionais em banco de dados orientado a objetos.** 2006. 228f. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, UFSC. Florianópolis, SC.

O paradigma da orientação a objetos apresenta-se como um padrão para a modelagem de sistemas de informação e sua representação através dos diagramas da UML auxiliam profissionais da área. A utilização das propriedades de persistência de objetos pode ser utilizada em ambientes de gerenciamento, como o *data warehousing*, fornecendo flexibilidade na aplicação do modelo multidimensional. A pesquisa propõe uma metodologia para implantar o modelo multidimensional em banco de dados orientado a objetos, seguindo a representação através dos diagramas da UML e o padrão da linguagem de definição de objetos da ODMG. A proposta aborda cinco etapas definidas na metodologia e sua posterior validação. Cada uma das etapas relaciona as atividades a serem realizadas para o processo de definição do modelo multidimensional seguindo os parâmetros da orientação a objetos. As etapas baseiam-se na definição do negócio a partir do ambiente operacional, seguindo pela geração do modelo multidimensional, sua representação através dos diagramas da UML, o posterior mapeamento em ODL e sua efetiva implementação e persistência em banco de dados orientado a objetos. Cada uma das etapas da metodologia inclui procedimentos sistematicamente definidos, abordando as características do modelo multidimensional, conceitos da orientação a objetos, padronização da ODMG e a persistência de objetos em banco de dados. O estudo de caso para verificar a proposta segue todas as etapas definidas na metodologia, relacionando os resultados em cada uma delas.

Palavras-chave: Banco de dados. Modelagem multidimensional. Orientação a objetos. UML.

Abstract

BORBA, Sueli de Fátima Poppi. **Metodologia para implantação de modelos multidimensionais em banco de dados orientado a objetos.** 2006. 228f. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, UFSC. Florianópolis, SC.

The object orientation paradigm is presented as a standard for the modelling of information systems and its representation through the UML diagrams have been used for computer science professionals. The use of the object persistence properties can be used in management environment, the data warehouse being an example, providing flexibility in the application of the multidimensional model. This research proposes a methodology to implant a multidimensional model in object-oriented database, using UML diagrams and the ODMG pattern for object definition language. The proposal reports five stages defined in the methodology and its posterior validation. Each one of the stages shows the activities to be made for the definition process of the multidimensional model following the object orientation parameters. The stages have been based on the business definition starting from the operational environment, being followed by the generation of the multidimensional model, its representation using UML diagrams, the posterior mapping in ODL and the effective implementation and persistence in object-oriented database. Each one of the stages of the methodology includes procedures systematically defined, reporting the multidimensional model characteristics, object orientation concepts, ODMG standardization and the persistence in object-oriented database. The case study to validate the proposal follows all the stages defined in the methodology, reporting the results in each one of them.

Key words: Database. Multidimensional modeling. Object orientation. UML

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

1FN - Primeira Forma Normal
3FN - Terceira Forma Normal
API - *Application Programming Interface*
BD - Banco de Dados
BDOO - Banco de Dados Orientado a Objetos
BDR - Banco de Dados Relacional
CRM - *Customer Relationship Management*
DAG - Diagrama Acíclicos Direcionados
DC - *Dimension Class*
DDE - *Dynamic Data Exchange*
DDL - *Data Definition Language*
DER - Diagrama Entidade Relacionamento
DML - *Data Manipulation Language*
DOLAP - *Desktop On-line Analytical Processing*
DW - *Data Warehouse*
ER - Entidade-Relacionamento
ERP - *Enterprise Resource Planning*
ETL - *Extract, Transform, Load*,
FC - *Dimension Fact*
FK - *Foreign Key*
IL - *Intermediate Level*
LL - *Lower Level*
MD - Modelo Multidimensional
MER - Modelo Entidade-Relacionamento
MOLAP - *Multidimensional On-line Analytical Processing*
O3LAP - *Object Oriented On-line Analytic Processing*
ODL - *Object Definition Language*
ODMG - *Object-Oriented Database Management Group*
OID - Identificador de Objeto
OLAP - *On-line Analytical Processing*
OLTP - *On-line Transaction Processing*
OMG - *Object Management Group*
OMT - *Object Modeling Technique*
OO - Orientado a Objetos
OOSE - *Object-Oriented Software Engineering*
OQL - *Object Query Language*

OREF - *Object REFerence*
PK - *Primary Key*
ROLAP - *Relational On-line Analytical Processing*
SBDOO - Sistemas de Banco de Dados Orientados a Objetos
SGBD - Sistema de Gerenciamento de Banco de Dados
SQL - *Structure Query Language*
UL - *Upper Level*
UML - *Unified Modeling Language*
UP - *Unified Process*
XP -*Extreme Programming*

METODOLOGIA PARA IMPLANTAÇÃO DE MODELOS MULTIDIMENSIONAIS EM BANCO DE DADOS ORIENTADO A OBJETOS

1 INTRODUÇÃO

A evolução tecnológica, acelerada a partir da década de 1980, auxilia o processo decisório das empresas, imposto pela globalização e competitividade, tornando os sistemas de informação essenciais à sobrevivência das organizações e seus negócios. O sucesso organizacional converge diretamente para o fator mais relevante, que é o valor da informação, pois nesta nova economia, determina-se a liderança pelo que se sabe e no tempo em que as informações são conhecidas. A solução de problemas e a tomada de decisão ocorrem em breves intervalos de tempo. Neste espaço as alternativas analisadas podem mudar, sendo necessário incluir novos requisitos ou buscar propostas diversificadas. Portanto, analisar informações é um processo irreversível para a tomada de decisão.

Neste contexto, alguns conceitos e ferramentas surgem com o objetivo de auxiliar este processo. A tecnologia de *Data Warehousing* é voltada para o estudo de técnicas e ferramentas utilizadas em aplicações que envolvam análise intensiva de dados e integração de fontes de dados heterogêneas, de forma a prover flexibilidade e agilidade na gerência, manutenção e acesso a estes dados, oferecendo uma maneira flexível e eficiente de obter informações concisas e integradas (Inmon, 2001).

Toda organização necessita de uma visão de seu desempenho sobre suas operações. Os dados normalmente estão distribuídos por múltiplos sistemas de informação (CRM, ERP, *Supply Chain*). Estes sistemas normalmente estão rodando em diferentes plataformas tecnológicas, com distintas partes do negócio, com suas estruturas de dados físicas e esquemas de identificação diferentes. A complexidade de sumarizar e manter estes dados de forma significante para que a organização tenha uma visão clara e global de seu desempenho registra os principais objetivos do *Data Warehouse* - DW (Longman, 2004).

A visão multidimensional de dados não é algo novo, pois gestores de informação observam a evolução dos dados gerenciais em dimensões, como produtos, clientes, período de tempo e outros. A necessidade de se obter de maneira rápida e simples

todo o histórico de informação do sistema operacional faz com que as organizações busquem alternativas para estruturar e acessar estes dados (Cavero et al., 2003).

Ainda hoje, tradicionalmente, sistemas de bancos de dados são usados para o processamento de dados aplicativos, e muitos SGBDs - Sistemas de Gerenciamento de Banco de Dados - direcionam-se a isto. Porém, os métodos tecnológicos mudam rapidamente, e estas mudanças fazem aumentar a expansão dos domínios naturais das tecnologias de banco de dados - BD. Através da visão multidimensional do DW gera-se o modelo de dados, que atualmente, utiliza dois principais esquemas: *Star Schema* (Modelo Estrela) e *Snowflake Schema* (Modelo Floco de Neve), aplicados ao modelo Relacional de Banco de Dados (Kimball, 1998; Levene, 2003).

Muitos estudos e pesquisas voltam seus objetivos para a modelagem multidimensional, e alguns novos modelos têm sido propostos, considerando a abordagem relacional para banco de dados. Nos últimos anos, observa-se que a comunidade técnico-científica está direcionando seus trabalhos para um paradigma que se consolida, gradativamente, no processo conceitual de desenvolvimento de sistemas - a orientação a objetos (Trujillo et. al 2000, 2003; Abelló et al. 2000b, 2002, 2005).

A realidade dos bancos de dados orientados a objetos vem de encontro a estas metodologias, transformando o processo em um ciclo de modelagem e implementação do modelo conceitual ao físico, considerando que os objetos agregam informação e comportamento de forma intuitiva (Lee et al., 2005). Os objetos podem ser reutilizados e compartilhados entre aplicações, representando significativamente a realidade.

O modelo de banco de dados orientado a objetos implementa requisitos da abordagem Orientada a Objetos - OO, observando que anteriormente estes requisitos eram apenas conceituados, mas não implementados. Novas aplicações de dados, dentre elas a tecnologia de DW, têm a característica de tratar um grande volume de dados. Portanto, aliar a tecnologia de Sistemas de Banco de Dados Orientados a Objetos - SBDOO à tecnologia de DW mostra-se como uma opção que suporta dados como objetos e permite ao gerenciador ver tais dados como objetos.

Porém, as atuais abordagens de modelagem de DW não apresentam mapeamentos para o tratamento dos objetos de um banco de dados, subutilizando a tecnologia da orientação a objetos. A maioria dos modelos busca mapear os objetos para o modelo relacional (Kimball, 1998; Levene, 2003).

Estudos apontam que até 40% de qualquer código de aplicação é envolvido para mapear objetos para e de um banco de dados relacional - BDR, considerando que as soluções deste modelo de banco de dados não mapeiam relacionamentos complexos, sendo necessário realizar junções manualmente. Assim, embora os objetos sejam mapeados em banco de dados relacional, não é possível conseguir o desempenho exigido com o modelo de objetos, gerando uma simplificação do modelo e seus relacionamentos, buscando reduzir o número de junções. Uma vez que a camada de mapeamento foi implementada, tem que ser mantida, e deve ser capaz de evoluir como a aplicação evolui. Com o aumento de complexidade, o esforço e custo para desenvolver e manter estes mapeamentos aumentam exponencialmente (Versant, 2001).

A impedância mal sucedida entre o modelo de dados relacional e o de objetos requer desmontar os objetos para seu armazenamento. O mapeamento de objetos para tabelas relacionais pode ser particularmente complexa quando abrange herança, dificultando sua implementação (Paterson e Haddow, 2004; Philippi, 2004).

Para gerar um modelo OO em um banco de dados relacional faz-se necessário considerar as dificuldades da implementação de herança e objetos complexos. O paradigma da orientação a objetos não apresenta uma solução simples para o armazenamento da persistência e os SBDOOs apresentam-se como a abordagem natural para tal objetivo (Philippi, 2004).

Considerando a utilização do modelo de objetos e sua persistência, o modelo de banco de dados orientado a objetos apresenta-se como a tecnologia para a implementação direta de objetos, mapeando relacionamentos complexos, através do tratamento de *links* de relacionamentos como coleções (Cattell et al., 2000).

Portanto, no contexto da modelagem multidimensional representada em um modelo orientado a objetos, o desafio que se apresenta pode ser identificado como a resolução das seguintes questões:

- como gerar um modelo multidimensional utilizando o paradigma da orientação a objetos?
- como aplicar os conceitos da orientação a objetos na modelagem multidimensional?

Considerando a persistência dos objetos em BDOO, apresentam-se alguns elementos a serem considerados para a solução do problema apresentado:

- quais os parâmetros e etapas para utilizar os conceitos da orientação a objetos na modelagem multidimensional?
- todos os conceitos da orientação a objetos e, conseqüentemente, de banco de dados orientado a objetos podem ser aplicados na persistência do modelo multidimensional?
- quais são as vantagens na utilização de banco de dados orientados a objetos na implementação de um modelo multidimensional, observando seu mapeamento, desempenho e representação?

Diante de tais questões, faz-se necessário estabelecer os procedimentos necessários para a resolução do problema apresentado.

1.1 Objetivos

O presente trabalho tem como objetivo propor uma metodologia para a implementação da modelagem multidimensional, aliada às novas tecnologias de banco de dados orientados a objetos, a partir do nível conceitual e do paradigma da orientação a objetos.

Esta tese contempla o estudo do modelo multidimensional e suas representações, analisando a representatividade da OO e sua real aplicação em banco de dados, propondo um conjunto de técnicas e processos a serem utilizados para sua efetiva implementação.

Para atingir este objetivo, observam-se algumas questões:

- quais são os modelos existentes propostos para a modelagem multidimensional?
- quais são as diferenças existentes entre os modelos que utilizam o paradigma da orientação a objetos para representar a modelagem multidimensional?
- como validar a aplicação dos conceitos e componentes da orientação a objetos na modelagem multidimensional?

1.2 Objetivos Específicos

Para atingir o objetivo principal deste trabalho e resolver as questões anteriormente mencionadas, faz-se necessário um estudo amplo sobre os conceitos e aplicações envolvidos na modelagem multidimensional, em orientação a objetos e em banco de dados orientado a objetos. Para tanto, foram estabelecidos alguns objetivos específicos:

- analisar e comparar os modelos já existentes que propõem a utilização de conceitos da orientação a objetos no modelo multidimensional;
- verificar a aplicabilidade dos modelos analisados para o paradigma da orientação a objetos;
- analisar e comparar os mapeamentos existentes do modelo multidimensional;
- verificar as vantagens da tecnologia de banco de dados orientada a objetos comparada a tecnologia relacional;
- definir um conjunto de técnicas e processos, que compõem a metodologia proposta, para a implementação de um modelo multidimensional em banco de dados orientado a objetos;
- verificar a proposta metodológica através de uma aplicação, utilizando persistência de objetos.

Portanto, este trabalho consiste na proposta de uma metodologia para a modelagem multidimensional, aplicando-se o paradigma da orientação a objetos e persistência de objetos.

1.3 Justificativas e Relevância do Estudo

1.3.1 Contextualização da proposta na Engenharia da Produção

A Engenharia de Produção e Sistemas é uma área voltada ao projeto e gerência de sistemas que envolvam pessoas, materiais, tecnologias e o próprio ambiente deste processo. A competência desta área está assim definida (ABEPRO, 2005):

> Compete à Engenharia de Produção o projeto, a implantação, a operação, a melhoria e a manutenção de sistemas produtivos integrados de bens e serviços, envolvendo homens, materiais, tecnologia, informação e energia. Compete ainda especificar, prever e avaliar os resultados obtidos destes

> sistemas para a sociedade e o meio ambiente, recorrendo a conhecimentos especializados da matemática, física, ciências humanas e sociais, conjuntamente com os princípios e métodos de análise e projeto da engenharia.

O avanço da produção de serviços, associada a evolução da tecnologia da informação, faz com que a Engenharia de Produção e Sistemas analise e avalie a aplicação de seus sistemas e processos perante as necessidades da sociedade moderna, buscando minimizar os problemas nas organizações e otimizar seu desempenho.

Toda melhoria na estratégia de construção de processos de produção baseia-se na integração do planejamento efetivo do projeto, monitoramento e técnicas de controle, que forneçam um nível de integração organizacional de todos os parâmetros e funções do processo, incorporando estratégicas tecnológicas (Hiremath e Skibniewski, 2004).

Neste contexto, surge a inteligência organizacional, que busca aplicar tais conhecimentos na gestão dos negócios, abrangendo aspectos estratégicos da utilização da informação em novas aplicações, como através da tecnologia dos novos modelos de SGBDs, que permitem o tratamento de novos tipos de dados.

1.3.2 Trabalhos existentes e pesquisa proposta

Nos últimos anos, muitos trabalhos desenvolvidos na área de banco de dados estão direcionados à modelagem multidimensional. Alguns destes trabalhos voltam-se à modelagem multidimensional em nível formal, apresentando mecanismos formais do cálculo ou álgebra relacional como operadores das várias dimensões (Sapia, 1998). Outros trabalhos apresentam o modelo físico através da implementação da multidimensionalidade implementada em um SGBD relacional específico (Zendulka, 2001). Alguns trabalhos de modelagem conceitual enfocam o modelo Estrela e buscam seu mapeamento a partir do modelo Entidade-Relacionamento - ER (Monteiro, 1998) ou a geração automática de um modelo multidimensional a partir do modelo ER (Phipps e Davis, 2002; Freitas et al., 2002; Peralta et al., 2003*)*. Outros trabalhos voltam-se ainda para representar as principais propriedades da modelagem multidimensional em nível conceitual, gerando notações específicas (Golfarelli et al., 1998a; Sapia, 1998b; Hüsemann et al., 2000; Nguyen et al., 2000; Torlone, 2003).

A orientação a objetos utiliza a UML - *Unified Modeling Language* - como linguagem padrão para o processo de modelagem e alguns poucos esforços buscam estender a modelagem multidimensional a partir deste paradigma, verificando sua aplicabilidade (Luján-Mora et al., 2002, 2004a). Os trabalhos voltados para esta nova concepção buscam minimizar o trabalho dos desenvolvedores no entendimento de novas metodologias e nomenclaturas para cada novo projeto a ser modelado. Porém, muitos dos trabalhos encontrados neste contexto, utilizam os conceitos da orientação a objetos simplesmente para a representação de seus diagramas (Nguyen et al., 2000; Trujillo, 2000), enquanto alguns outros abordam propriedades dinâmicas da modelagem multidimensional (Luján-Mora et al., 2002; Trujillo et al, 2003).

1.3.3 Diferencial da proposta

Embora abordem os aspectos da orientação a objetos, observa-se que os trabalhos apresentados não discorrem sobre uma metodologia para a utilização deste paradigma aliado ao modelo multidimensional e não propõem mapeamento para o modelo de banco de dados orientado a objetos, mostrando apenas a representação dos conceitos da modelagem multidimensional.

Neste trabalho, são apresentadas duas questões relacionadas ao modelo multidimensional que devem ser investigadas nesta pesquisa. Inicialmente, são analisados e comparados os modelos que propõem a utilização da OO para a modelagem multidimensional. Neste caso, foi elaborado um levantamento das propostas com a OO para a modelagem multidimensional de dados, mostrando como esta modelagem pode ter suas propriedades representadas através da UML.

Este problema pode ser compreendido quando se analisam trabalhos de pesquisa voltados à modelagem multidimensional, que é o fundamento dos DWs. Estes trabalhos propõem notação gráfica específica, que exige o conhecimento de um novo modelo, além da notação da própria modelagem (Golfarelli, 1998; Sapia, 1998; Tryfona, 1999; Baekgaard, 1999). Em contraposição, a UML aparece como uma linguagem padrão para modelagem orientada a objetos, representando vários aspectos dos projetos de sistemas. Portanto, a primeira parte da pesquisa foi dedicada a um estudo e posterior análise das propostas que indicam a utilização da OO e de sua notação em UML da modelagem multidimensional.

O segundo problema é verificar os mapeamentos existentes a partir do modelo multidimensional, propondo um mapeamento para o banco de dados orientado a objetos. Pesquisas desenvolvidas focam o mapeamento entre os modelos ER e Estrela (Monteiro, 1998; Kimball, 1998; Oliveira, 1998; Moody e Kortink, 2000). Os trabalhos mais recentes na área da modelagem multidimensional apontam para o paradigma da orientação a objetos como uma abordagem para as novas aplicações, utilizando principalmente, dados, estruturas ou aplicações complexas, nas quais se encontra o DW. Considerando-se que alguns trabalhos apontam para o modelo conceitual orientado a objetos, a proposta de trabalho pretende abranger tais modelos, considerando suas principais propriedades e características (Luján-Mora et al., 2002; Trujillo et al., 2001 e 2003; Abelló et al., 2005).

1.3.4 Relevância da proposta

Em contraste com o modelo de dados relacional, não existe um padrão para modelos de dados multidimensionais (Lechtenbörger e Vossen, 2003; Torlone, 2003). Nos trabalhos analisados, a modelagem multidimensional, aliada às novas tecnologias de banco de dados orientados a objetos, não apresenta uma metodologia de implementação, considerando que praticamente todas as abordagens consideram a modelagem multidimensional aplicada ao modelo relacional, indistintamente de um procedimento para sua modelagem e posterior implementação. A presente pesquisa, portanto, foca seus objetivos para a proposta de uma metodologia que atenda tais requisitos.

1.4 Limites da Pesquisa

O presente trabalho abrange a modelagem multidimensional e sua representação na OO, além do mapeamento da modelagem utilizando um padrão para banco de dados orientado a objetos, compondo uma metodologia de implementação do modelo multidimensional orientado a objetos.

Considerando tal proposta, o trabalho limita-se ao estudo comparativo das propostas de modelos conceituais da modelagem multidimensional e sua representação através da UML, não abordando outras metodologias ou linguagens, tais como a OMT (*Object Modeling Technique*), XP (*Extreme Programming*), OOSE (*Object-Oriented Software Engineering*) ou o *Fusion Method.*

Observando a seqüência do trabalho, que consiste em aplicar a proposta através de um estudo de caso, o mapeamento restringe-se ao banco de dados pós-relacional Caché.

1.5 Metodologia

A metodologia abordada na pesquisa fundamenta-se nas seguintes etapas:

1. realizar um estudo bibliográfico sobre a modelagem de dados, abrangendo os níveis de modelos de dados, a modelagem Entidade-Relacionamento e a Modelagem Multidimensional;
2. descrever os componentes do paradigma orientado a objetos, aplicados à modelagem multidimensional;
3. levantar e comparar os modelos que utilizam os conceitos da OO para representar o modelo multidimensional;
4. estudar e comparar as propostas de mapeamento do modelo multidimensional;
5. elaborar um conjunto de técnicas e processos, gerando as etapas de uma metodologia para a implementação do modelo multidimensional no paradigma da orientação a objetos, utilizando sua representação em UML;
6. aplicar a proposta apresentada através de um estudo de caso.

Este trabalho abrange o estudo dos conceitos do modelo multidimensional, suas aplicações e modelos propostos; os conceitos e componentes da UML e sua efetiva aplicação no modelo multidimensional; e as características e aplicações do banco de dados orientado a objetos. A integração destas tecnologias funde-se no mapeamento do modelo multidimensional gerado a partir da UML para um banco de dados orientado a objetos.

A figura 1 representa as atividades do desenvolvimento da presente pesquisa, descrita em suas várias etapas. A primeira etapa deve conceituar os níveis de modelos de dados (conceitual, lógico e físico), apresentar os elementos do Modelo ER e do Modelo Multidimensional, além de relacionar estes conceitos com a tecnologia de DW, considerando o ambiente operacional.

Na segunda etapa, são apresentados os principais conceitos da orientação a objetos e da UML, além das características da OO aplicadas ao modelo multidimensional.

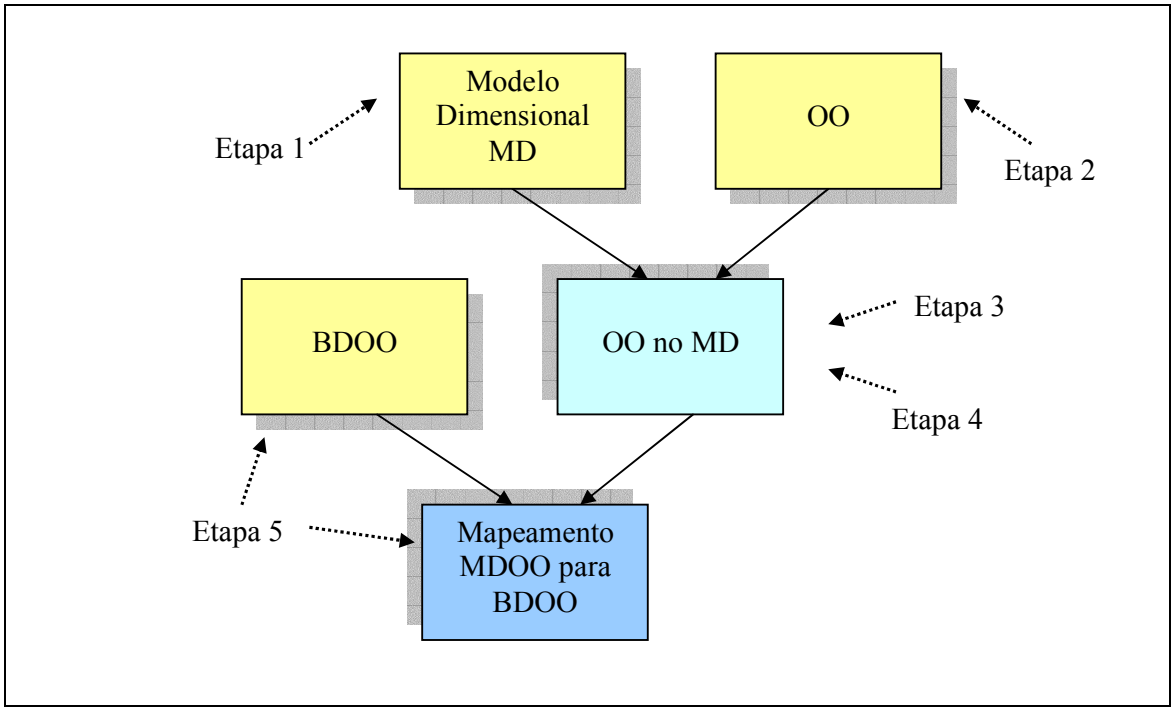

Figura 1 - Abrangência da presente pesquisa.

Na terceira etapa da pesquisa, através de um levantamento dos trabalhos até então apresentados, são comparados os modelos que propõem a utilização da OO para a modelagem multidimensional, verificando suas diferenças. O estudo comparativo busca tabular suas características e propriedades, além de conhecer o nível de modelagem proposto em cada modelo. Esta etapa visa identificar os principais modelos existentes.

A etapa seguinte propõe-se a estudar e analisar os aspectos de mapeamentos dos modelos multidimensionais. Os aspectos desta etapa estão relacionados aos requisitos e estágios do mapeamento.

A última etapa do trabalho propõe elaborar, através da descrição de técnicas e processos, uma metodologia com as etapas de geração do modelo multidimensional a partir do modelo operacional, utilizando sua representação através da UML implementada em um modelo de banco de dados orientado a objetos e, finalmente, aplicar a proposta metodológica através de um estuo de caso, utilizando as etapas

do modelo apresentado. Nesta etapa são analisadas as contribuições e os resultados alcançados com a pesquisa.

1.6 Resultados esperados

Com a conclusão do trabalho proposto, espera-se apresentar como resultados uma análise comparativa dos modelos multidimensionais que utilizam a OO para sua representação conceitual e uma comparação dos mapeamentos existentes do modelo multidimensional.

Porém, a maior contribuição do trabalho centra-se na proposta da metodologia para gerar um modelo multidimensional, modelado através da UML e implementado em um banco de dados orientado a objetos, utilizando a persistência de dados.

A metodologia de análise e projeto de sistemas utiliza a UML para aplicar os conceitos da OO, porém, observa-se que a tecnologia de banco de dados orientado a objetos é relativamente nova e são poucos os produtos comerciais que implementam as características da orientação a objetos. Muitos produtos focam seus esforços na tecnologia objeto-relacional, que não implementa totalmente os conceitos da OO, como a herança múltipla, por exemplo. Estes conceitos são implementados segundo a abordagem relacional, gerenciados na *interface* com o usuário.

Neste contexto, a utilização da UML para a modelagem multidimensional requer uma conversão para sua implementação em um modelo de bancos de dados relacional. Com a utilização de um modelo de SBDOO, as características e conceitos da orientação a objetos são preservados e aplicados segundo sua definição e modelagem.

Portanto, o trabalho pretende auxiliar os desenvolvedores no processo de geração de dados através da implementação de um modelo multidimensional segundo o paradigma da orientação a objetos.

1.7 Organização do trabalho

A pesquisa foi dividida em capítulos, pertinentes aos objetivos anteriormente definidos.

O capítulo 2 aborda os aspectos conceituais da modelagem de dados, incluindo os conceitos da modelagem Entidade-Relacionamento e da modelagem

multidimensional, elencando os elementos básicos de cada uma delas e as vantagens da modelagem multidimensional no ambiente de DW, comparada a modelagem ER. O capítulo apresenta os principais modelos de dados multidimensionais. Ainda são inseridos os conceitos de Data Warehouse, seus benefícios e vantagens para o apoio a decisão organizacional, além do ciclo de vida e as fases de um projeto de data warehouse, contextualizando o trabalho. Visando o objetivo do trabalho, o capítulo apresenta uma rápida revisão sobre banco de dados, relacionando os modelos: relacional, multidimensional e orientado a objetos.

O capítulo 3 aborda a modelagem segundo o paradigma da orientação a objetos, suas principais definições e apresenta os requisitos da modelagem a partir da representação na linguagem UML para o modelo multidimensional, abordando seus aspectos conceituais.

O capítulo 4 apresenta as metodologias de desenvolvimento do modelo multidimensional, apresentando um estudo comparativo destes trabalhos. Neste capítulo as duas primeiras etapas da proposta são concluídas. São analisados e comparados os modelos que utilizam a OO para a representação do modelo multidimensional, verificando a aplicabilidade de cada um deles para o paradigma da orientação a objetos. Também são analisados os mapeamentos existentes para o modelo multidimensional.

No capítulo 5 é apresentada a proposta da metodologia para a modelagem multidimensional segundo o paradigma da OO, representada pela UML. A metodologia está elaborada em um total de cinco etapas, definidas subseqüentemente para que possa ser utilizada como aplicação prática.

A aplicação da metodologia é apresentada no capítulo 6, através da implementação de um estudo de caso, que desenvolve cada uma das etapas da metodologia de um modelo em um banco de dados orientado a objetos.

Finalmente, no capítulo 7 são apresentadas as conclusões deste trabalho e propostas para novos trabalhos.

2 MODELAGEM DE DADOS

2.1 Introdução

A modelagem de dados é a representação das informações existentes em um determinado contexto (Machado, 2004). Esta abordagem fornece níveis de abstração, omitindo detalhes do armazenamento dos dados e permitindo uma visão geral do problema, garantindo que todos os objetos de dados requeridos pela aplicação possam estar completamente representados.

Para conceber a primeira etapa do trabalho proposto, este capítulo aborda os aspectos conceituais da modelagem de dados, incluindo os conceitos básicos do Modelo Entidade-Relacionamento (ER) e do Modelo Multidimensional (MD).

A seção dois do capítulo aborda os tipos de modelos de dados, segundo a visão das propostas de Cougo (1997) e Chen (1976).

A terceira seção apresenta o Modelo Entidade-Relacionamento (MER) proposto por Chen (1976), com os elementos básicos desta proposta e as etapas para sua elaboração, considerando que este modelo é o método mais comum da modelagem para bancos de dados e utilizada no ambiente operacional.

Na seção quatro são introduzidos os conceitos inicias de DW, além das fases dos modelos de projeto propostos por Kimball (2002) e Hüsemann et al. (2000), contextualizando a modelagem conceitual, escopo do trabalho. Ainda são abordados os benefícios deste tipo de ambiente.

A quinta seção apresenta o MD, com seus princípios e objetivos, discutindo as principais vantagens desta abordagem, além das definições das mais conhecidas operações multidimensionais. A seção mostra uma comparação entre o clássico padrão de modelagem de dados do ambiente operacional - MER - com o modelo padrão de modelagem do ambiente de DW - MD.

Na seção seguinte são apresentados os modelos mais usados para esta abordagem, que são os Modelos Estrela e sua variação, o Floco de Neve, além de outros modelos propostos em trabalhos recentemente realizados.

A seção sete aborda os níveis da modelagem de dados e suas equivalências nos modelos ER e multidimensional.

A seção oito visa subsidiar a fase final do presente trabalho, que implementa a metodologia em um banco de dados orientado a objetos, através da persistência de

dados. Para cumprir com este objetivo, a seção apresenta os conceitos dos modelos de bancos de dados relacional, multidimensional e orientado a objetos.

Finalizando o capítulo, a seção nove aborda o mapeamento entre os modelos conceituais e modelos de dados.

2.2 Modelo de Dados

Um modelo de dados é uma representação gráfica dos dados de uma área de interesse específica, e um bom modelo deve descrever: entidades (tabelas), atributos (colunas), relacionamentos entre os dados, sua cardinalidade, chaves primárias e estrangeiras (Singh, 2001).

O modelo de dados é aplicado como um meio para a obtenção de estruturas de dados que levem ao projeto de banco de dados e é utilizado para favorecer o processo de comunicação, facilitando o entendimento e a transmissão de conceitos, especificações e regras. Os modelos de dados repassam uma visão precisa, concisa, objetiva e com pouca, ou nenhuma ambigüidade. O modelo de dados é considerado um conjunto de ferramentas conceituais que pode descrever os dados que serão armazenados, os relacionamentos entre estes dados, sua semântica e regras de consistência (Silberschatz, 1999).

Um modelo de dados genérico deveria apresentar as seguintes características (Inmon, 2002):

- um modelo ER de alto nível exibindo os principais assuntos e as relações entre estes assuntos;
- um nível intermediário de modelo de dados em que cada área de estudo principal tem sua própria representação, contemplando:
    - as chaves dos grupos de dados que pertencem a mesma área de estudo,
    - alguns dos atributos mais comuns de cada agrupamento de dados,
    - os relacionamentos entre os agrupamentos de dados, incluindo:
        - relacionamentos hierárquicos 1:n;
        - tipos de relacionamentos.

“O modelo de dados fornece uma estrutura arquitetônica para construir futuros sistemas e bancos de dados”, tendo por objetivo mostrar os relacionamentos entre

os dados da organização, suportando a consolidação e integração das informações ao longo de aplicações legadas (Singh, 2001, p. 100). O modelo de dados pode ser classificado em: conceitual, lógico e físico (Cougo, 1997, p.28). A seguir, estes tipos de modelos de dados são descritos.

2.2.1 Modelo conceitual

Modelo conceitual é aquele em que "os objetos, suas características e relacionamentos têm a representação fiel ao ambiente observado, independente de limitações quaisquer impostas por tecnologia, técnicas de implementação ou dispositivos físicos" (Cougo, 1997, p.28).

As particularidades de implementação são ignoradas neste modelo, portanto, sua caracterização deve ser a mesma, permanecendo imutável para uma futura implementação em um SGBD relacional ou outro qualquer. Este modelo associa-se às fases de análise e nunca às fases de projeto.

2.2.2 Modelo lógico

O modelo lógico requer o planejamento da estrutura lógica de dados para o banco de dados, envolvendo a análise do ambiente de aplicação e dos tipos de estruturas lógicas disponíveis no sistema de banco de dados (Chen, 1976).

O modelo lógico de dados é aquele em que "os objetos, suas características e relacionamentos têm a representação de acordo com as regras de implementação e limitação impostos por algum tipo de tecnologia" (Cougo, 1997, p. 29). Porém, tal representação ainda é independente do armazenamento físico e suas estruturas de dados.

O modelo lógico é gerado a partir do modelo conceitual e está associado à fase de projeto. O modelo lógico deve ser caracterizado pela representação de conceitos, tais como: chaves, métodos de acesso, formatos de campos, etc., diferenciando-se, portanto, da definição formal da abordagem Entidade-Relacionamento, proposta por Chen (1976).

2.2.3 Modelo físico

O modelo físico é aquele em que “a representação dos objetos é feita sob o foco do nível físico de implementação das ocorrências, ou instâncias das entidades e seus relacionamentos” (Cougo, 1997, p.30).

A implementação física está diretamente ligada ao SGBD adotado para armazenar os dados, com diferentes estruturas de armazenamento, endereçamento, acesso e alocação física, mapeando diferentemente o modelo lógico, segundo o sistema adotado.

“A finalidade do projeto físico de banco de dados é selecionar a estrutura física de dados que seja mais adequada para determinado ambiente de aplicação” (Chen, 1990, p.5).

2.3 Modelo Entidade-Relacionamento

A concepção do modelo ER proposto por Chen (1976) foi o de acrescentar um estágio intermediário ao projeto lógico de banco de dados, através da identificação das entidades e dos relacionamentos de interesse no projeto a ser desenvolvido. No modelo ER, deve-se examinar os dados como um todo, considerando a representação do mundo real para se gerar um esquema da empresa e, em uma segunda etapa, traduzí-lo para um esquema do usuário.

O modelo ER propõe a representação dos dados através de um diagrama contendo as entidades e os relacionamentos entre estas entidades. Kimball (1997) afirma que o modelo ER é uma técnica de projeto lógico que busca remover a redundância de dados.

O modelo ER é implementado nas seguintes etapas (Chen, 1990, p.20):

- identificar tipos de entidade;
- identificar tipos de relacionamentos;
- desenhar um diagrama E-R com tipos de entidade e relacionamentos;
- identificar tipos de valores e atributos;
- traduzir o diagrama E-R em um diagrama de estrutura de dados;
- projetar formatos de registros.

2.3.1 Elementos básicos

2.3.1.1 Entidades

Entidade é “uma ‘coisa’ que pode ser distintamente identificada” (Chen, 1990, p. 20). As entidades de um modelo podem depender da existência de uma outra entidade, caracterizando uma entidade fraca.

2.3.1.2 Relacionamentos

Relacionamentos podem existir entre entidades e podem ser classificados em diferentes tipos, que são mapeados em relacionamentos um-para-um (1:1), um-para-muitos (1:n) e muitos-para-muitos (n:n) (Chen, 1976). Os relacionamentos envolvendo entidades fracas podem gerar um mapeamento 1:n ou n:n.

Uma hierarquia em um modelo ER é qualquer seqüência de entidades unidas por relacionamentos 1:n, todos alinhados na mesma direção (Moody e Kortink, 2000).

2.3.1.3 Atributos

“Entidades e relacionamentos têm propriedades que podem ser expressas em termos de pares atributo-valor” (Chen, 1990, p. 24). As entidades são compostas de atributos e os atributos, por sua vez, possuem valores.

Em alguns casos, um atributo pode ter mais de um valor para uma determinada entidade. Para solucionar este problema, a indicação de 1:n no ponteiro da entidade identifica um atributo de valores múltiplos. Também existem casos de atributos de relacionamentos, que indicam valores para as ações de relacionamentos entre as entidades.

2.3.1.4 Identificadores

As entidades devem ser identificadas de forma absoluta através de seus atributos identificadores, que são as chaves primárias nos bancos de dados relacionais.

Os identificadores de relacionamentos são os identificadores das entidades envolvidas no relacionamento.

2.4 Data Warehouse

Um *Data Warehouse* - DW - é um conjunto de dados que oferece subsídios ao processo decisório. É caracterizado como uma cópia de dados de uma ou mais bases de dados, sendo uma valiosa alternativa para os tradicionais acessos passivos de dados, vindos de autônomas e heterogêneas fontes de informação. O acesso ao DW é particularmente útil quando alto desempenho de resposta é desejado, ou quando as fontes de informação são caras ou transitórias.

DW é “um conjunto de dados baseado em assuntos, integrado, não-volátil, e variável em relação ao tempo, de apoio às decisões gerenciais” (Inmon, 1997, p.33).

As corporações possuem vários sistemas OLTP no ambiente operacional e seus dados fazem parte da infra-estrutura corporativa, sendo detalhados, não redundantes, atualizáveis e refletem os valores do período atual. Porém, os dados requeridos para o processo de decisão são freqüentemente sumarizados, abrangendo um período extenso de tempo, não atualizáveis e são redundantes para suportar diversas visões. Para fornecer uma visão para o suporte a decisão, os dados relevantes são extraídos do meio operacional, filtrados, codificados e sumarizados, gerando um ambiente de data warehousing (Singh, 2001, p.79).

Data warehousing pode ser considerado o projeto e a implementação de um processo, ferramentas e facilidades para se gerenciar e gerar informação completa, oportuna e compreensível para o processo decisório nas organizações.

Data warehousing é “o processo de integrar os dados de uma corporação em um único repositório a partir do qual os usuários finais podem criar relatórios e executar análises *ad hoc* de dados” (Singh, 2001, p.143).

2.4.1 Projeto do Data Warehouse

O projeto de DW requer atividades integradas, pois, apesar de considerar o DW como um processo contínuo, observa-se que cada projeto de implementação deve ter um ciclo finito, determinando início e fim (Kimball, 2002).

O trabalho de Luján-Mora e Trujillo (2004) propõe um método de desenvolvimento do DW baseado na *Unified Modeling Language* (UML) e no *Unified Process* (UP), considerando as fases definidas no UP para o ciclo de vida do projeto: definição, elaboração, construção e transição, requisitos, análise, projeto, implementação e teste.

O DW pode ser definido em três níveis distintos (Luján-Mora et al, 2004b):

- conceitual: define o DW em uma visão conceitual;
- lógico: apresenta aspectos lógicos do projeto de DW, como a definição para o processo de extração, transformação e carga;
- físico: define aspectos físicos do DW, como o armazenamento das estruturas lógicas em diferentes discos ou a configuração de servidores de banco de dados que suportam o DW.

O modelo proposto por Kimball (2002, p. 380) abrange as principais atividades do processo, orientando na idealização do DW. A figura 2 mostra o ciclo de vida multidimensional, representando as atividades principais de um projeto de data warehouse. No diagrama pode-se observar a modelagem multidimensional como a fase em que se traduz os requisitos em um modelo de dados.

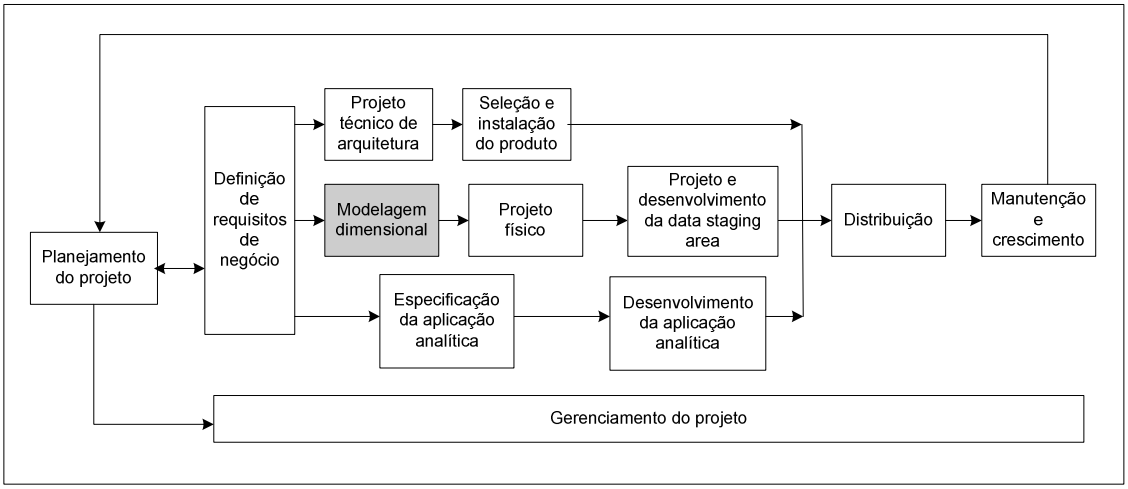

Figura 2 - Diagrama do ciclo de vida multidimensional.
Fonte: Kimball, 2002, p. 381.

O ciclo de vida definido por Kimball (2002) apresenta as seguintes fases para seu desenvolvimento:

- Planejamento do projeto: nesta fase deve-se considerar a exigência e viabilidade da criação de um DW pela necessidade de gestão do negócio da organização. Nesta fase define-se o escopo do DW e sua justificativa, considerando questões de prioridades e de gerenciamento, além das vantagens e custos do projeto.

- Definição dos requisitos de negócio: um dos objetivos do DW é auxiliar no processo de decisão da organização, portanto, o DW é projetado com base na definição dos requisitos do negócio.
- Projeto técnico de arquitetura: fase em que se define o requisito da estrutura geral para a integração das tecnologias que serão adotadas, detalhando os componentes e subsistemas, além dos requisitos de segurança e infra-estrutura física.
- Seleção e instalação de produtos: a partir do projeto técnico, são selecionados os produtos para o desenvolvimento e implantação do projeto de DW.
- Modelagem multidimensional: nesta fase, deve-se avaliar a granularidade, a consistência histórica, os valores válidos e a disponibilidade de atributos e sua documentação e, ainda, registrar os nomes de tabelas e atributos e as regras de cálculo para fatos e dimensões.
- Projeto físico: especificação dos detalhes para o banco de dados físico, como definição de colunas, restrições de nulidade, criação de índices e critérios de agregação, além do particionamento do banco de dados.
- Projeto e desenvolvimento da *data staging area*: nesta fase define-se como será a preparação dos dados do sistema operacional para o modelo multidimensional, através do processo de extração, transformação e limpeza dos dados operacionais.
- Especificação da aplicação analítica: planejar e projetar padrões para as aplicações, visando atender os requisitos dos usuários.
- Desenvolvimento da aplicação analítica: na fase do desenvolvimento das aplicações, emprega-se a padronização anteriormente convencionada, observando as possíveis alterações de especificação do modelo de dados.
- Distribuição: as etapas anteriormente descritas convergem para a implantação do DW, porém, deve-se observar a necessidade de treinamento e suporte para que esta finalização ocorra com sucesso.
- Manutenção e crescimento: o processo de manutenção e crescimento é contínuo a partir da implantação do DW, para garantir seu equilíbrio quanto a demanda de novos serviços, aplicações e usuários, ou quanto ao aprimoramento dos já existentes.

A utilização de data warehousing e suas aplicações implicam em não utilizar os conceitos dos projetos normalizados. A princípio, um bom projeto de warehouse deve ter a forma de uma estrela, consistindo de uma tabela central de fatos, que contém os fatos de interesse para uma aplicação OLAP, conectada a várias tabelas de dimensão através das restrições de integridade referencial, baseadas nas várias chaves multidimensionais. A comunidade de pesquisa não tem voltado sua atenção para a padronização de modelos para DW e, especificamente para o modelo conceitual, estabelecendo diretrizes e restrições de integridade dentro do contexto de modelos multidimensionais. Como resultado, parece haver uma discrepância entre tradicionais projetos de banco de dados aplicados a bancos de dados operacionais, e os princípios de projetos aplicados a DW (Hüsemann et al., 2000).

2.4.2 Benefícios do Data Warehouse

Considerando o principal objetivo de um DW, que é prover informações para o processo de decisão da organização, um benefício está implícito em seus objetivos, caracterizando-se pelo aumento de produtividade dos tomadores de decisão organizacional.

O ambiente de DW ainda oferece outros benefícios ligados a suas características, pois com a criação de um repositório de dados, inicia-se a viabilidade do processo de gerenciamento de grandes volumes de dados. O DW é parte integrante dos sistemas de informação, pois proporciona uma localização centralizada de dados extraídos de diversos sistemas aplicativos do ambiente operacional (Singh, 2001).

Ainda enumeram-se como benefícios a disponibilidade de dados históricos corporativos e um melhor desempenho na busca de informações, considerando-se o isolamento do ambiente operacional. O quadro 1, adaptado da comparação elaborada por Inmon (1997), mostra algumas das principais diferenças entre o ambiente operacional e o ambiente de DW.

Complementando a visão de Inmon (1997), Moody e Kortink (2000), colocam as principais diferenças entre o banco de dados operacional e do DW:

- Acesso do usuário final: em um ambiente de DW, usuários escrevem consultas diretamente contra a estrutura do banco de dados, considerando

que em um ambiente operacional, usuários geralmente só acessam o banco de dados através de um sistema de aplicação *front end*. Em um sistema tradicional de aplicação, a estrutura do banco de dados é invisível ao usuário.

- Apenas leitura: DW são bancos de dados efetivamente apenas para leitura, usuários podem recuperar e analisar dados, mas não podem atualizá-los. Os dados armazenados em um DW são atualizados por processos de extração.

Ambiente operacional	**Ambiente Data warehouse**
Perspectiva de dados atuais	Perspectiva de dados no tempo
Com atualização dos dados	Sem atualização dos dados
Baseado em aplicações	Baseado em assuntos
Processamento repetitivo	Processamento heurístico
Com base em transações	Com base em análises
Suporte a decisões cotidianas	Suporte a decisões estratégicas de longo prazo
Serve a comunidade administrativa e transacional	Serve à alta administração
Dados exatos para o momento do acesso	Dados de momentos já decorridos
Não contemplam a redundância	A redundância não pode ser ignorada

Quadro 1 - Diferenças entre os ambientes operacional e data warehouse.

Fonte: Adaptado de Inmon (1997, p. 18).

2.5 Modelagem multidimensional

"O modelo de dados tem um papel fundamental para o desenvolvimento interativo do data warehouse" (Oliveira, 1998, p. 55). A elaboração de um modelo de dados ajuda na compreensão das regras de negócio que o DW gerencia. A modelagem de dados é uma das mais importantes diferenças entre o ambiente operacional e o ambiente de DW (Kimball, 1996, Trujillo et al., 2003).

O processo de modelagem de dados busca transformar modelos de dados orientados a processos, considerados modelos funcionais, em modelos de dados orientados a negócio, os modelos multidimensionais. Ou seja, através da modelagem de dados, transforma-se a visão de processo em visão de negócio. Aplicando o modelo de dados ao ambiente de DW, o modelo conceitual seria representado pelo modelo corporativo do DW e o modelo lógico pelo esquema multidimensional.

A modelagem multidimensional, tratando-se de um modelo relativamente novo, apresenta conceitos gerais, mas ainda não possui uma padronização em suas

técnicas de desenvolvimento, como o modelo ER (Machado, 2000). O ambiente de DW difere do ambiente operacional (Singh, 2001, p. 144):

> Em contraste com os sistemas OLTP que são projetados em torno de entidades, decomposição funcional, análise de transição de estado e inter-relacionamentos, o modelo de data warehouse baseia-se em dimensões, hierarquias, fatos e dispersão.

2.5.1 Objetivos da modelagem multidimensional

O problema em se usar técnicas tradicionais no projeto de banco de dados em um ambiente de DW é que as estruturas de banco de dados resultantes são muito complexas para os usuários finais entenderem e usarem. O objetivo da modelagem multidimensional é gerar estruturas de banco de dados que sejam fáceis para que os usuários finais possam entender e criar suas consultas. Um segundo objetivo é maximizar a eficiência das consultas. Este objetivo é alcançado principalmente pela minimização do número de tabelas e seus relacionamentos. Isto reduz a complexidade do banco de dados e minimiza o número de junções exigidas em consultas do usuário.

A modelagem multidimensional também oferece suporte a operações OLAP, que são utilizadas para a análise dos dados, tais como (Oliveira, 1998; Machado, 2000, Abelló, 2001b):

- *Drill-Down*: permite a navegação do mais alto nível até o dado detalhado, obtendo mais informações sobre os dados que estão sendo apresentados, seja descendo uma hierarquia ou adicionando dimensões que complementem a análise dos dados;
- *Drill-across*: permite que duas ou mais tabelas de fato que compartilham dimensões sejam combinadas numa única visão;
- *Roll-Up*: permite a navegação do nível de detalhe até o mais alto nível de sumarização dos dados;
- *Slice*: permite fatiar o cubo, mantendo a mesma perspectiva de visualização dos dados;
- *Dice*: permite a visualização dos dados, mudando a perspectiva da visão.

Usar o paradigma multidimensional durante as fases de desenvolvimento é necessário para definir um modelo de dados objetivo nos níveis conceitual, lógico e físico, desenvolvendo uma metodologia sólida que gera referências de como criar e

transformar estes modelos durante o processo de desenvolvimento (Dinter et al., 1999).

2.5.2 Vantagens da modelagem multidimensional

A modelagem multidimensional apresenta algumas vantagens (Kimball, 2002, p.27):

- a simplicidade e simetria;
- o número reduzido de tabelas e a utilização de descritores faz com que diminua a probabilidade de incidência de erros;
- o processamento é mais eficiente e com menos junções;
- a flexibilidade em mudanças, devido a sua estrutura previsível, suporta mudanças inesperadas no comportamento do usuário.

O modelo multidimensional foi adotado como a abordagem predominante para projetos práticos de data warehouse e representa uma contribuição importante para modelagem de dados e projetos de banco de dados (Moody e Kortink, 2000).

Em um DW os fatores de desempenho consistem na granularidade e particionamento dos dados, além da alteração da estrutura das chaves, para a inclusão do elemento temporal (Oliveira, 1998, p. 74). A necessidade da utilização da multidimensionalidade surge tipicamente por necessidades de negócio, incluindo (Pilot, 2003):

- tempo de resposta rápido;
- dados modelados segundo a estrutura do negócio;
- desempenho e consistência de relatórios.

“A maioria dos desenvolvedores aceita a realidade de que é melhor manter os elementos operacionais separados do warehouse” (Singh, 2001, p.274). O quadro 2 mostra de forma resumida, as características dos dados de cada ambiente: operacional e de data warehouse.

2.5.3 Elementos básicos do modelo multidimensional

A visão multidimensional representa como as informações são analisadas sob a forma de negócio, ou seja, o cruzamento das informações gerenciais. A visão multidimensional é representada através do cubo, que mostra visões sob vários aspectos da informação.

Dados em modelos ER em ambiente operacional	Dados em modelos multidimensionais em ambiente de data warehouse
Dados detalhados	Dados resumidos ou refinados
Vida curta, de modificação rápida	Vida longa, estático
Requerem acesso em nível de registro	Os dados são agregados em conjuntos, semelhante ao banco de dados relacional
Transações padrão repetitivas e padrões de acesso	Consultas *ad hoc* com alguns relatórios específicos
Atualização em tempo real	Atualização periódica com cargas maciças

Quadro 2 - Diferenças entre dados operacionais e do data warehouse.

Fonte: Singh, 2001, p. 274.

A idéia fundamental da modelagem multidimensional é que praticamente quase toda espécie de dados de negócios pode ser representada como um tipo de cubo de dados, em que as células do cubo contêm valores de medida e a extremidade do cubo define a dimensão natural dos dados. Mais que três dimensões são definidas em projetos, portanto, deve-se nomear o cubo como hipercubo, embora os termos cubo e cubo de dados sejam mais comumente usados (Kimball, 1998, p. 165).

A representação dos elementos da modelagem multidimensional faz-se através do cubo pela dificuldade em se visualizar um hipercubo, e se aplica a este tipo de modelagem (Machado, 2000). A figura 3 representa um fato Vendas por meio de um cubo, com três dimensões: Localização, Produto e Tempo. As duas primeiras dimensões possuem dois níveis de hierarquia.

Pode-se fazer a correspondência de cada eixo no espaço multidimensional com uma coluna de uma tabela relacional, em que cada ponto representa um valor que corresponde à intersecção destas colunas. Quaisquer dados podem ser considerados multidimensionais, mas normalmente a referência é feita a dados representando objetos ou eventos que podem ser descritos e, portanto, classificados por dois ou mais de seus atributos (Oliveira, 1998).

Todo modelo multidimensional é composto de uma tabela com uma chave composta de várias partes, chamada Tabela Fatos e um grupo de tabelas menores, chamadas Tabelas Dimensão. Cada tabela de dimensão possui uma chave primária (Pk - *primary key*) de parte única, que corresponde exatamente a uma das partes da chave da tabela de fatos (Kimball, 1998).

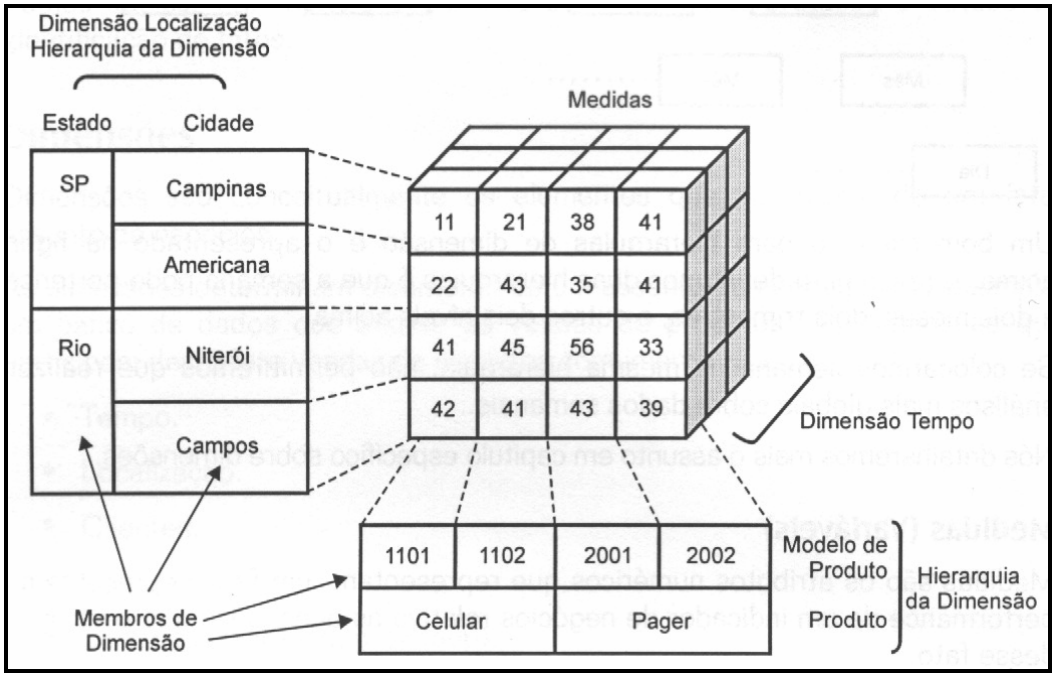

Figura 3 - Representação do modelo multidimensional através de um cubo.
Fonte: Machado, 2000, p. 66.

A modelagem multidimensional apresenta os seguintes elementos básicos (Machado, 2000; Singh, 2001; Kimball, 2002):

- Fatos: coleção de itens de dados, que se compõem de dados de medida e de contexto, normalmente representados por dados numéricos e que são o foco da investigação do suporte à decisão, sendo considerada a principal tabela de um modelo multidimensional;
- Dimensões: elementos que participam de um fato, normalmente não possuem atributos numéricos e constituem-se de agrupamentos lógicos de atributos com uma chave de relacionamento comum;
- Medidas: atributos numéricos que representam um fato, normalmente qualificadores métricos conceituais.

2.5.3.1 Fatos

O modelo multidimensional discerne entre fatos e atributos. Um fato é normalmente alguma coisa que não é conhecido antecipadamente. Muitos fatos na área de negócios são numéricos, embora alguns poucos podem ser valores textuais. Muitos campos de dados podem ser inicialmente definidos como atributos, mas na

verdade, em uma análise mais aprofundada, podem ser definidos como fatos (Kimball, 1998, p.165).

Uma tabela de fatos “é a principal tabela de um modelo multidimensional em que as medições numéricas de desempenho da empresa estão armazenadas” (Kimball, 2002, p.21).

A multidimensionalidade está baseada na dualidade fatos-dimensões, ou seja, fatos são analisados em referência a dados nas dimensões. Um fato representa um assunto de análise, enquanto suas dimensões mostram as diferentes visões que se pode usar para estudá-los (Abelló et al., 2001c).

A tabela de fatos possui como característica a esparsidade, ou seja, se não existe um cruzamento para alguns valores das dimensões, a tabela de fatos não armazena zeros. Quando os dados possuírem a característica de esparsidade em um modelo, é muito importante que a implementação OLAP não reserve espaço de armazenamento para combinações que não são utilizadas (Pilot, 2003).

A dispersão não é uma entidade e não é representada por tabelas. A dispersão é manipulada de forma implícita. Alguns fatos podem existir apenas para uma pequena fração do mercado ou período. O esquema estrela gerencia a dispersão simplesmente não gravando registros onde estas combinações são inválidas (Singh, 2001).

As tabelas de fatos “contêm múltiplas colunas de fatos, relacionadas por uma chave multidimensional comum, que geralmente consiste de atributos de mais de uma dimensão” (Singh, 2001, p.88).

A “tabela de fatos contém registros tirados dos dados operacionais, com uma chave primária composta de chaves externas para as tabelas multidimensionais”, considerando que podem conter dados consolidados (Oliveira, 1998, p. 36).

Fatos representam elementos de informação atômicos em um banco de dados multidimensional. Um fato consiste em quantificar valores armazenados em medidas e um contexto qualificativo que é determinado por níveis de dimensão. Cada nível de dimensão contém um conjunto de instâncias ou elementos. Um esquema Fatos representa o contexto multidimensional para um conjunto de fatos que partilham os mesmos níveis multidimensionais terminais (Hüsemann et al., 2000).

Um fato é definido como um item de interesse do negócio, que é descrito por um grupo de atributos chamados medidas ou atributos de fato - atômicos ou derivados - que estão contidos em células ou pontos em um cubo de dados (Trujillo et al., 2001).

2.5.3.2 Dimensões

Uma dimensão é uma coleção de textos como atributos que são altamente relacionados uns aos outros. Os termos dimensão e fatos tiveram sua origem em um projeto de pesquisa realizado na década de 1960 pela General Mills e Dartmouth University. Estes termos foram utilizados uma década mais tarde, de modo consistente para descrever dados corporativos pela AC Nielsen e IRI (Kimball, 2002, p.20).

Uma dimensão pode ser assim definida (Kimball, 2002, p.28).

> Toda dimensão é equivalente, e todas as dimensões são pontos de entrada simetricamente iguais para a tabela de fatos, e os dados mais granulares ou atômicos possuem a maior multidimensionalidade, considerando que os dados atômicos que não foram agregados são os mais expressivos e que as dimensões estão sempre acompanhando uma tabela de fatos, contendo descritores textuais.

A mais importante característica do paradigma da modelagem multidimensional é a divisão dos dados em fatos (composto de medidas) e dimensões, para fornecer dados em um nível satisfatório de granularidade (Luján-Mora e Trujillo, 2003).

Estas tabelas de dimensão representam relações hierárquicas em uma empresa ou negócio e, normalmente, não apresentam muita normalização. Na maioria das vezes as dimensões representam hierarquias, que são armazenadas em uma única tabela de dimensão, e não em várias tabelas normalizadas. Isso faz com que o desempenho das consultas aumente, considerando que não são necessárias junções para a obtenção dos dados. Normalmente apresentam muitos atributos.

As dimensões apresentam o contexto para analisar os fatos (Trujillo et al., 2001):

- Dimensões membros: uma dimensão membro é um nome distinto ou identificador usado para determinar uma posição dos itens de dados, como por exemplo, todos os meses e anos que compõem uma dimensão tempo e todas as cidades, estados e regiões que compõem uma dimensão localização. Uma dimensão contém muitas dimensões membros.
- Dimensões hierárquicas: pode-se organizar os membros de uma dimensão em uma ou mais hierarquias. Cada hierarquia pode também ter vários níveis de hierarquia. Cada membro de uma dimensão está alocado em

uma estrutura hierárquica. A hierarquia define o relacionamento entre atributos da dimensão que identificam os diferentes níveis existentes.

Hierarquias de dimensão são classificadas em dois tipos básicos: hierarquia simples que consiste em um caminho de agregação linear dentro de uma dimensão, como por exemplo: dia → mês → ano, em uma dimensão Tempo; e a hierarquia múltipla que contém pelo menos dois caminhos de agregação em uma dimensão (Hüsemann et al, 2000).

Uma dimensão especial e que deve receber atenção especial é a dimensão Tempo. Esta dimensão pode apresentar variação de hierarquias, como um ano que abrange meses, que são compostos de semanas e dias, por exemplo. Além da consideração de controle de anos fiscais ou ainda períodos de meses.

As medidas de tempo podem ser armazenadas na periodicidade original e exibidas em qualquer periodicidade desejada, incluindo diária, semanal, mensal, trimestral, anual, bimestral, etc. (Pilot, 2003; Mendelzon e Vaisman, 2003).

Em um modelo relacional de dados, para propósitos de normalização, não são fundidos dados de ano, mês, semana ou dia em uma única tabela. Muitas vezes tais informações são resumidamente armazenadas em uma única data de referência. Em um modelo de dados multidimensional, estas referências são fundidas como uma única tabela, denominada Dimensão Tempo.

2.5.3.3 Medidas

Uma medida é um atributo numérico de um fato, representando o desempenho ou comportamento do negócio relativo àquela dimensão. As medidas são determinadas pela combinação de membros das dimensões e são localizadas nos fatos. As medidas são definidas como grupos de dimensões que derivam da granularidade escolhida para representar os fatos (Trujillo et al., 2001).

2.5.3.4 Relacionamentos

Um relacionamento é representado com linhas que interligam entidades. O relacionamento entre duas entidades pode ser definido em termos de cardinalidade, que pode ser: um-para-um (1:1), um-para-muitos (1:n) e muitos-para-muitos (n:n). Quando a cardinalidade de uma entidade é 1:n, freqüentemente o relacionamento

representa dependência de uma para outra entidade. Neste caso, a chave primária da entidade pai é herdada na entidade dependente como parte da sua chave primária.

Uma tabela de fatos, por sua chave primária ser composta de duas ou mais chaves estrangeiras (Fks - *Foreign Keys*), pode expressar um relacionamento muitos-para-muitos (Kimball, 1998).

A agregação é o relacionamento “parte de“. É o processo pelo qual os dados de nível baixo de detalhe são previamente sumarizados e incluídos em tabelas que armazenam informações sumarizadas. Estas tabelas permitem que as aplicações antecipem consultas do usuário e eliminem a repetição de cálculos.

2.5.3.5 Atributos

Atributos descrevem as características das propriedades de uma entidade. Os atributos de uma tabela de fatos usualmente são numéricos e aditivos, enquanto aqueles das tabelas de Dimensão freqüentemente contêm informação textual descritiva. Os atributos de uma dimensão são usados para identificar quais fatos serão analisados.

Os atributos da dimensão são a fonte da maioria das restrições interessantes nas consultas do data warehouse e são sempre a fonte das linhas de cabeçalho de uma saída no SQL (*Structured Query Language* - linguagem de consulta estruturada) (Kimball, 1998, p.145).

“As tabelas multidimensionais contêm múltiplas colunas de atributos (normalmente baseadas em caracteres), relativas ao mesmo atributo atômico” (Singh, 2001, p. 89).

Se o campo é uma medida que adota vários valores e participa dos cálculos, torna-se um fato. Se é uma descrição com valor discreto que é relativamente constante e participa de restrições, é um atributo multidimensional. Resumidamente, a tabela de fatos formada por medidas numéricas é associada a um conjunto de tabelas de dimensão preenchidas com atributos descritivos.

Os atributos, normalmente chamados atributos de dimensão, fornecem as particularidades que caracterizam dimensões (Trujillo et al., 2001).

2.5.4 Modelo normalizado x Modelo multidimensional

O modelo ER, que é o modelo normalizado, fornece um modelo de dados de uma área específica de interesse, usando dois conceitos básicos: entidades e relacionamentos entre entidades. Os modelos multidimensionais usam três conceitos básicos: medidas, fatos e dimensões.

A modelagem ER é uma técnica de projeto lógico que busca eliminar a redundância dos dados. Este processo é extremamente benéfico para processar transações deixando-as simples e determinísticas. Em contrapartida, a modelagem multidimensional é uma técnica de projeto lógico que busca apresentar os dados em um *framework* padrão que é intuitivo e permite um acesso de alto desempenho (Kimball, 1998).

Em um ambiente de *data warehousing* a modelagem é radicalmente diferente do ambiente operacional, esquecendo-se tudo o que se sabe sobre a modelagem ER (Kimball, 1998). Porém, Moody e Kortink (2000) apresentam a modelagem ER como sendo igualmente aplicável no contexto de data warehouse tanto quanto no contexto operacional e que, segundo os autores, fornece uma importante base para projetos de data warehouse.

Singh (2001, p. 151) elabora uma comparação entre o modelo relacional e o modelo multidimensional, enfatizando que o modelo relacional tem sua origem no suporte aos processos operacionais, enquanto o modelo multidimensional destina-se a atender as necessidades dos profissionais do conhecimento e do suporte a decisão.

A modelagem multidimensional é bem diferente da modelagem normalizada, pois esta busca remover redundâncias de dados, gerando várias tabelas em um banco de dados. A diferença substancial entre as duas modelagens consiste, portanto, no nível de normalização de cada uma. Porém, tanto a modelagem normalizada quanto a multidimensional podem ser representadas através do diagrama ER (Kimball, 2002).

Aspectos relevantes para a comparação entre os dois modelos (Singh, 2001, p.151):

- no modelo multidimensional, a perspectiva da visualização das informações é de um período no tempo, enquanto no relacional, as transações são atômicas;

- os sistemas relacionais registram eventos atuais ou transações, enquanto o modelo multidimensional não se preocupa com os eventos em si, mas com o resultado quantitativo em um intervalo de tempo;
- a modelagem dos relacionamentos de entidades é a base do modelo relacional, que os registra explicitamente, enquanto no modelo multidimensional, estes relacionamentos estão implícitos na interseção das dimensões com a tabela de fatos.

Comparando-se o modelo ER (normalizado) e o modelo Estrela (multidimensional), são apontadas algumas das diferenças entre os dois modelos e a principal diferença entre estes modelos é a complexidade, considerando que a normalização do modelo ER gera inúmeras tabelas conectadas entre si, tornando-o confuso e de difícil compreensão para o usuário. O modelo multidimensional do tipo Estrela apresenta, porém, uma estrutura mais simples, em que uma tabela central, a Tabela Fatos, é ligada a várias Tabelas Dimensão de uma única vez, tornando o modelo mais facilmente compreendido (Oliveira, 1998).

O modelo ER está baseado no relacionamento entre os dados, de acordo com o procedimento real, enquanto que o modelo multidimensional está projetado com base nas necessidades do usuário, de acordo com sua visão de dados.

No modelo multidimensional as informações são armazenadas e analisadas sob uma perspectiva histórica, oferecendo uma visão global consistente e integrada para toda a empresa. No modelo ER as transações são atômicas, consistentes em um determinado escopo.

Os relacionamentos no modelo ER são modelados explicitamente, enquanto no modelo multidimensional tais relacionamentos são representados pela existência de fatos na intersecção entre as dimensões (Oliveira, 1998).

A maioria dos processos parte da suposição de que todos os elementos chaves podem ser agrupados em relacionamentos hierárquicos normalizados. Este conceito é comumente usado na modelagem ER, que busca assegurar a integridade do modelo. Porém, um modelo ER com estruturas normalizadas reflete relacionamentos e entidades como um instantâneo no tempo, não capturando mudanças nos relacionamentos com o passar do tempo. Como resultado, usar o modelo ER com tais objetivos pode gerar problemas de integridade de dados em uma análise histórica.

A volatilidade dos dados envolvidos gera problemas na análise operacional, pois a única hierarquia que não é volátil é a própria dimensão tempo. As demais hierarquias são voláteis no tempo (Wan, 2004).

A chave para entender a relação entre o diagrama multidimensional - DM e o diagrama entidade relacionamento – DER, é que um único diagrama ER pode ser decomposto em múltiplos diagramas multidimensionais. De certo modo, o DER pode representar em um único diagrama, múltiplos processos que nunca coexistem em um único conjunto de dados e em um único ponto consistente no tempo (Kimball, 1997).

Considera-se que o modelo ER e o modelo multidimensional, embora relacionados, são diferenciados um do outro. O quadro 3 mostra um resumo das diferenças entre os dois modelos.

Característica	Modelo ER	Modelo multidimensional
Estrutura	• Confusa e de difícil compreensão para o usuário	• Simples e mais facilmente compreendida
Base do projeto	• Relacionamento entre os dados	• Base nas necessidades do usuário, de acordo com sua visão de dados
Armazenamento de dados	• Transações são atômicas	• Perspectiva histórica
Relacionamentos	• Modelados explicitamente	• Representados pela intersecção entre as dimensões
Redundância de dados	• Remove a redundância	• Permanece a redundância

Quadro 3 - Resumo das principais diferenças entre os modelos ER e multidimensional.

Fonte: Adaptado de Oliveira, 1998, p. 65.

Procedimentos de desnormalização podem ser adotados como projeto pré-físico de banco de dados e como um passo intermediário entre a modelagem lógica e física, e fornece uma visão adicional refinada do banco de dados lógico antes do projeto físico. O processo de refinamento requer um bom nível de conhecimento do projetista de banco de dados, bem como um conhecimento adequado dos requisitos da aplicação (Shin e Sanders, 2005).

2.6 Modelos de dados multidimensionais

A primeira abordagem formal que apresentou um modelo de dados multidimensional, propondo um conjunto mínimo de operações no hipercubo, foi publicada por Agrawal et al. (1997). Trabalhos mais recentes abordam variações do

tratamento com o cubo para a representação das operações OLAP (Maniatis et al., 2005).

O primeiro passo para gerar o modelo multidimensional é identificar o processo de negócio a ser modelado, seguido da declaração da granularidade e, finalmente, a seleção das dimensões e fatos (Kimball e Ross, 2004).

2.6.1 Modelo Estrela

Em seu modelo lógico, Kimball (1997) define a representação de um hipercubo através do Modelo Estrela. Sua composição típica possui uma tabela central denominada Fatos (*fact table*) e um conjunto de entidades denominadas Dimensões (*dimension tables*), na forma de uma estrela, conforme ilustra a figura 4. A denominação de Esquema Estrela aplica-se a um modelo multidimensional baseado em tabelas em um banco de dados relacional. Em contrapartida, se os dados estiverem baseados em tecnologia de banco de dados multidimensional, seu armazenamento procede-se em cubos (Kimball, 2002).

O modelo Estrela apresenta o relacionamento entre a tabela de fatos e as tabelas de dimensão através de uma ligação do tipo um-para-muitos no sentido da dimensão para o fato, ou seja, a tabela de fatos possui múltiplas junções de conexão com outras tabelas, que são as tabelas de dimensão, mas cada uma destas tabelas possui apenas uma junção com a tabela central.

O modelo Estrela consiste de um processo em que se deve separar todos os elementos de dados envolvidos no contexto analítico em dois grupos: elementos numéricos ou referenciados como métricas de desempenho, que são valores numéricos com várias regras de agregação que podem ser aplicadas, como sumarização, média, porcentagem, etc.; e outro grupo com elementos fundamentais, usados para fatiar e visualizar os dados. Na seqüência do processo, organizam-se os elementos fundamentais em estruturas hierárquicas. Estas hierarquias ou dimensões formam os vários caminhos de navegação que auxiliam o usuário final a ter acesso a informação armazenada em um sistema OLAP (Wan, 2004).

Muitos projetos de DW utilizam o modelo multidimensional através do esquema Estrela, em que um tabela central (de fatos) está ligada a várias tabelas multidimensionais. A tabela de fatos representa os relacionamentos muitos-para-

muitos entre as tabelas de dimensão, tendo como chave primária uma chave composta de todas as chaves estrangeiras das tabelas de dimensão.

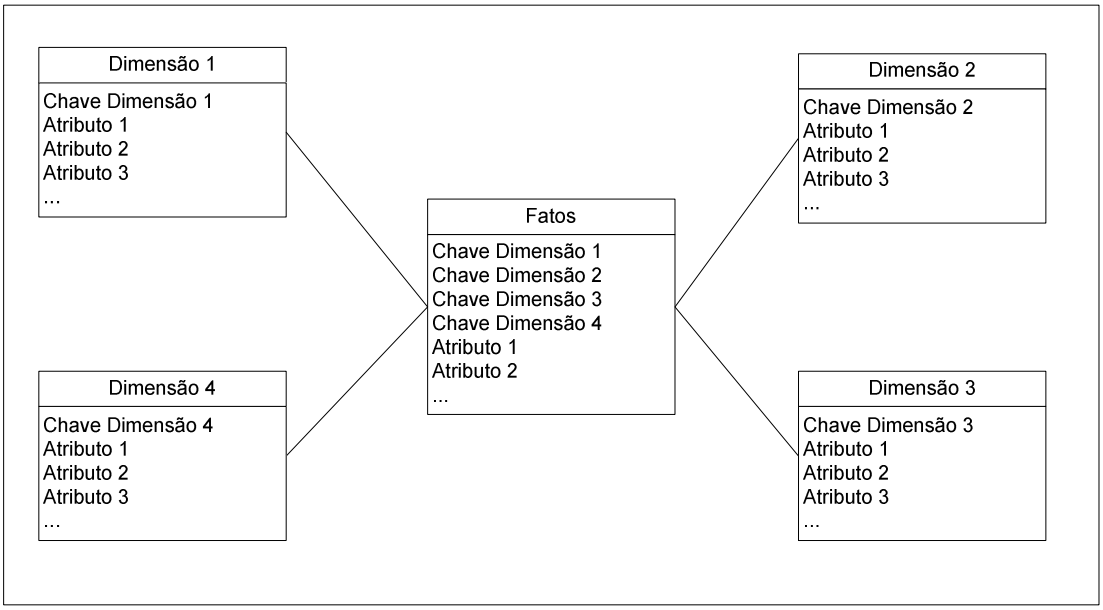

Figura 4 - Modelo Estrela.

2.6.2 Snowflake Schema

O *Snowflake Schema* ou modelo Floco de Neve é a decomposição de um modelo Estrela, de uma ou mais dimensões, que possuem hierarquias entre seus membros. Os relacionamentos neste modelo são de muitos-para-um entre os membros de uma dimensão, formando, através dos relacionamentos entre tabelas de dimensão, uma hierarquia.

Observando que as dimensões podem ser compostas de hierarquias de atributos, é comum os casos das tabelas de dimensão não serem normalizadas, resultando no esquema Floco de Neve. O modelo Floco de Neve consiste em se aplicar a terceira forma normal (3FN) sobre as tabelas de dimensão (Machado, 2000). Considerando que se trata de um modelo normalizado, diminui a redundância de dados, conforme mostra a figura 5.

Aspectos importantes do esquema Estrela e do esquema Floco de neve:

- em um esquema estrela cada dimensão terá uma chave primária;

- em um esquema estrela, uma tabela dimensão não terá nenhuma tabela pai, enquanto em um esquema floco de neve, poderá ter uma ou mais tabelas pai;
- no esquema estrela a própria tabela multidimensional armazena hierarquias das dimensões, enquanto em um esquema floco de neve as hierarquias são quebradas em tabelas separadas.

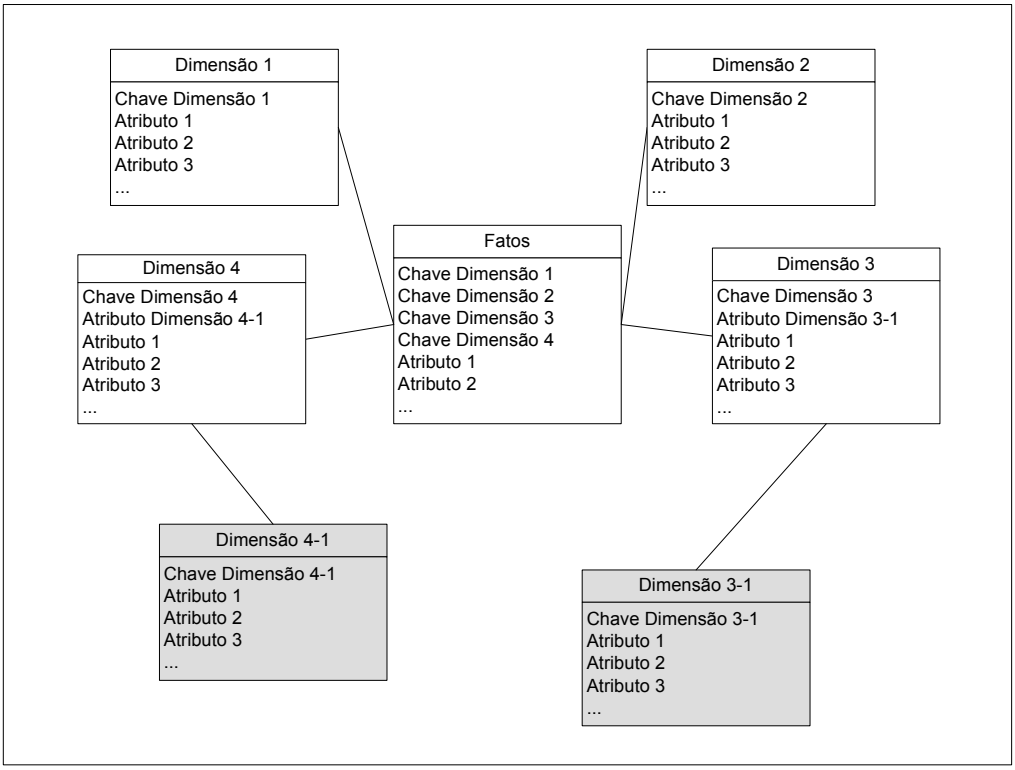

Figura 5 - Modelo Snowflake.

2.6.3 Outros modelos

O *Nested Multidimensional Data Model* (NMDM), definido por Lehner (1998), destaca a presença de dados em dois diferentes níveis de aninhamento, que fornecem flexibilidade para o processo de análise. Em cada hierarquia de classificação é colocado um atributo primário que são os elementos multidimensionais, como um nó folha de uma estrutura de árvore balanceada. O modelo representa uma única célula identificadora, chamada de Objeto Multidimensional Primário (PMO); um conjunto de atributos de classificação e atributos primários (um por dimensão), denotando a granularidade da célula. Todo o modelo é baseado em dimensões, que são definidas como uma hierarquia linear de níveis multidimensionais, os atributos de classificação.

O Modelo MD, de Cabibbo e Torlone (1998), foi considerado como um modelo lógico por seus autores, observando que independe de uma implementação específica e é obtido a partir de um modelo ER. Porém, Abelló et al. (2000a) classificam este modelo como conceitual, partindo da afirmação dos autores de que o modelo apresenta um nível de abstração superior ao modelo Estrela. O modelo apresenta dimensões e fatos, sendo que cada dimensão é organizada em uma hierarquia de níveis, que correspondem aos domínios dos dados de diferentes granularidades.

O trabalho de Hüsemann et al. (2000) propõe um esquema de DW a partir do esquema conceitual operacional. O trabalho estabelece diretrizes para distinguir se um atributo é um nível de dimensão ou um atributo de propriedade. Também propõe um formalismo gráfico para o projeto conceitual de DW, que captura esta distinção de um modo apropriado. Finalmente, o trabalho mostra como a forma normal multidimensional generalizada, originalmente proposta em Lehner et al. (1998), pode ser obtida para um esquema de DW a partir de um projeto.

Pesquisas em torno da modelagem multidimensional resultam, por vezes, na concepção de modelos de dados multidimensionais. Alguns destes modelos multidimensionais, como o esquema Estrela de Kimball (1997), que é caracterizado como um modelo de dados lógico multidimensional, provêem um alto nível de abstração por avaliar propriedades multidimensionais. Abelló et al. (2000a) e Trujillo et al. (2001) analisam e fazem um estudo comparativo das diferentes terminologias usadas nos modelos resultantes de algumas das pesquisas desenvolvidas, resumidas no quadro 4.

O *Dimensional Fact Model* (DFM) foi proposto por Golfarelli et al. (1998b, 1998c), através de um modelo gráfico e de uma metodologia para se obter o modelo multidimensional de um modelo operacional. O esquema multidimensional consiste de um conjunto de esquemas de fato e cada um destes contém um fato, medidas, dimensões e hierarquias. Um fato é o foco de interesse, juntamente com seus atributos e medidas. As dimensões são atributos discretos que determinam o nível mínimo de granularidade escolhido para representar o fato. Uma hierarquia é um conjunto de atributos de dimensão, ligados através de relacionamentos 1:1 ou 1:n.

O modelo *Multidimensional Entity Relationship* (M/ER), definido por Sapia et al. (1998b), surgiu com base no princípio de seus autores de que o modelo ER não era adequado para a modelagem conceitual multidimensional. Portanto, o modelo foi

definido com diretrizes para a especialização e extensão mínima de um modelo ER e a representação da semântica multidimensional. Considerando que o modelo é baseado no ER, níveis de dimensão e fatos podem ter atributos.

O modelo *starER*, elaborado por Tryfona et al. (1999) e baseado no modelo ER foi proposto a partir de uma lista de requisitos do usuário. Estes requisitos propõem-se a representar fatos e suas propriedades; ligar a dimensão temporal com fatos, capturando suas propriedades e associações; distinguir dimensões e categorizá-las em hierarquias. O modelo projeta um conjunto de fatos representando a realidade e suas propriedades (níveis multidimensionais e células de fatos), um conjunto de entidades que representa objetos reais e suas propriedades (dimensão), um conjunto de relacionamentos que representa um conjunto de associações entre conjuntos de entidades e conjuntos de fatos, e atributos que representam as propriedades estáticas de conjuntos de entidades, relacionamentos ou fatos (medidas e atributos de classificação).

Portanto, os modelos classificados por Abelló et al. (2000a, 2005) identificam no nível mais alto a representação de dimensões; no nível intermediário, os níveis multidimensionais e os relacionamentos das células de fato e/ou níveis multidimensionais; no nível mais baixo, os atributos de classificação e medidas e/ou atributos de classificação.

O quadro 4 mostra um resumo da comparação elaborada por Trujillo et al. (2001), identificando algumas propriedades multidimensionais dos três tipos de modelos conceituais. Só o Modelo *StarEr* considera os relacionamentos muitos-para-muitos entre fatos e dimensões indicando a cardinalidade exata entre eles. Nenhum dos modelos inclui medidas derivadas como parte do seu esquema conceitual.

	DFM	**M/ER**	**StarER**
Nível estrutural			
Fatos Relacionamentos muitos-para-muitos			X
Medidas atômicas	X	X	X
Medidas derivadas			
Aditividade	X		X
Dimensões Múltiplas hierarquias de classificação			
Nível dinâmico			
Especificação de requisitos	X	X	
Modelagem comportamental		X	

Quadro 4 - Comparação dos modelos conceituais multidimensionais.

Fonte: Resumo adaptado de Trujillo et al., 2001.

Com referência a dimensões, todos os três modelos usam diagramas acíclicos direcionados (DAGs) para definir atributos de dimensão para múltiplos e alternativos caminhos de hierarquias de classificação. Como os modelos *M/ER* e *StarEr* derivam do modelo Entidade-Relacionamento, eles usam relacionamentos *"IS-A"* para categorizar dimensões. O modelo *StarEr* não apresenta um mecanismo explícito para representar requisitos de usuário para modelagem dinâmica multidimensional.

2.7 Níveis de modelagem multidimensional

Durante o ciclo de desenvolvimento de projetos de sistemas, os modelos de dados passam por níveis distintos. A origem desta estratégia foi junto ao grupo ANSI-X3-SPARK, na década de 1970, pelas dificuldades quanto à implementação das bases de dados e a busca de padrões para gerenciar este ambiente (Cougo, 1997, p. 25). A figura 6 mostra este padrão, representando os níveis de projeto integrando a abordagem dos modelos conceitual, lógico e físico.

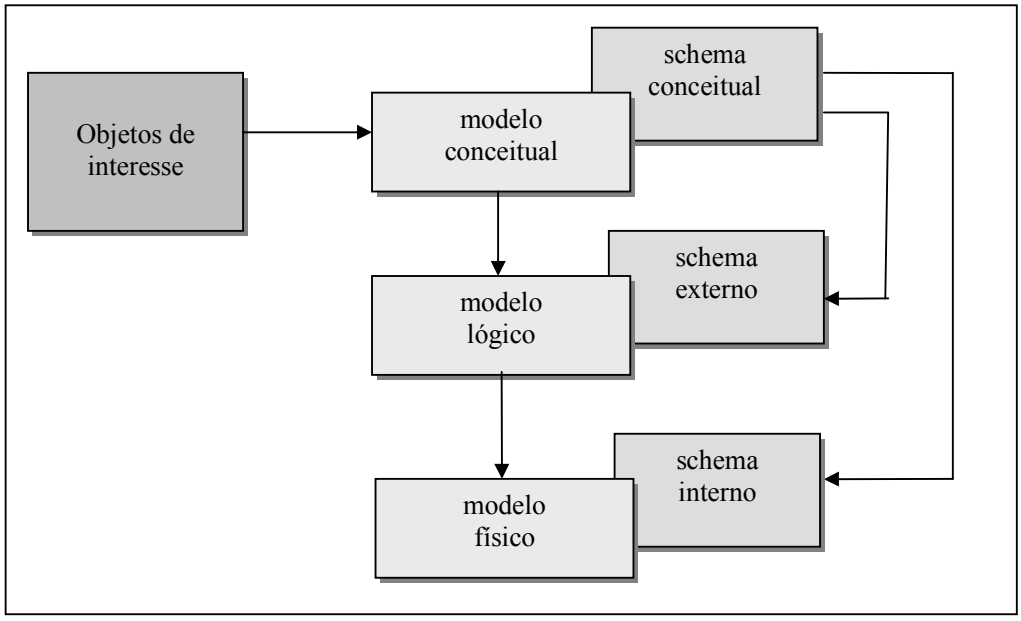

Figura 6 - Abordagem proposta pelo grupo ANSI-X3-SPARK.
Fonte: Cougo, 1997, p.31.

O padrão básico de definição e especificação dos elementos componentes dos bancos de dados incluía os aspectos conceituais, lógicos e físicos, estabelecendo o

uso de esquemas (*schemas*). Estes esquemas serviam de mapeamento para as estruturas do banco de dados.

O modelo ER é adequado para os sistemas de ambientes operacionais, porém, não possui aplicação adequada no modelo multidimensional. Enquanto os modelos dos ambientes operacionais utilizam as técnicas de normalização para gerar os dados em várias tabelas e seus relacionamentos, com o objetivo de reduzir a redundância, o modelo multidimensional busca utilizar estes dados com o menor número possível de tabelas e relacionamentos (Kimball, 1998; Oliveira, 1998; Machado, 2000; Torlone, 2003; Cavero et al., 2003).

A modelagem de dados para DW é completamente diferente da modelagem de sistemas OLTP, devido à complexidade do ambiente operacional, com a inclusão de dados históricos, além do baixo desempenho em razão das junções de tabelas (Machado, 2000).

Os modelos de dados tradicionais descrevem entidades e relacionamentos, focando a subdivisão das informações em várias tabelas, cada uma descrevendo uma entidade. As tabelas são inter-relacionadas através de junções (Singh, 2001).

Na modelagem do DW, as estruturas de dados são organizadas para descrever medidas e dimensões. As medidas fornecem os dados numéricos que ficam armazenados na tabela de fatos e as dimensões são os parâmetros que definem cada transação, armazenadas em tabelas vinculadas à tabela de fatos.

O nível conceitual é o mais próximo do usuário e independe da implementação. O nível lógico depende do tipo de SGBD, mas o usuário ainda consegue entendê-lo. O modelo físico depende diretamente do SGBD, considerando como os dados são armazenados. Similarmente ao ambiente operacional, no ambiente OLAP, no nível conceitual aparece o modelo de dados multidimensional (*Multidimensional Data Model*) e que reflete diretamente a abordagem do nível lógico, que pode ser: ROLAP (Relacional), O3LAP (Orientado a Objeto) ou MOLAP (muldidimensional puro) (Abelló et al., 2001a).

Um modelo de dados é composto por três níveis de modelagem: a modelagem de alto nível, de nível intermediário e de baixo nível (Oliveira, 1998, p.71):

- A modelagem de alto nível apresenta as entidades e seus relacionamentos, considerando o mais alto nível de abstração e seguindo um escopo de integração, que define os limites do modelo. Para o modelo

de dados do ambiente operacional no nível conceitual, pode-se utilizar o modelo ER.

- A partir do modelo de alto nível, gera-se o modelo no nível intermediário, que é formado por quatro elementos: um agrupamento primário de dados, composto pelos atributos que aparecem uma única vez; um agrupamento secundário de dados, abrangendo os atributos que podem aparecer mais de uma vez; um conector representando os relacionamentos dos dados; e o tipo dos dados. Os elementos do nível intermediário identificam os atributos de dados de um modelo e os relacionamentos entre seus atributos.
- O modelo de baixo nível ou modelo físico de dados é gerado a partir do modelo de nível intermediário, apresentando suas chaves e características físicas. O modelo físico necessita ser alterado para receber as características de desempenho. No modelo físico, efetua-se a implementação em um SGBD específico.

Considerando a modelagem multidimensional conceitual, a aplicação dos três diferentes níveis de detalhes, que podem ser equiparados à modelagem de dados tradicional, são assim resumidos (Abelló et al., 2001a):

- nível mais alto: onde se encontram as dimensões e os fatos. As dimensões são usadas para caracterizar os fatos e mostrar as óticas sob as quais os fatos serão analisados. Vários cubos representam o mesmo tipo de fato em um diferente nível de agregação e são agrupados em um fato.
- nível intermediário: as dimensões e os fatos são decompostos em níveis multidimensionais e células de fato, respectivamente. Os diferentes níveis multidimensionais em uma dimensão formam uma hierarquia. Agrupam-se células em diferentes classes que podem ser representadas como cubos n-dimensionais.
- nível mais baixo: mostra o grupo de atributos de um nível multidimensional e as células de fato. São os atributos de classificação e as medidas, respectivamente. As medidas são agrupadas em células, quando se referem ao mesmo fato.

O quadro 5 mostra os níveis de detalhe dos modelos de dados, segundo a abordagem tradicional (Cougo, 1997) e a abordagem multidimensional (Abelló et al., 2001a), situando no mesmo nível, as características de cada uma das abordagens.

O quadro 6 mostra o resumo dos diferentes elementos da modelagem multidimensional em níveis de detalhe, observando o assunto e a dimensão de análise.

Nível de detalhe	Modelagem tradicional	Modelagem multidimensional
Alto nível	• Apresenta as entidades e seus relacionamentos	• Apresenta as dimensões e os fatos
Nível intermediário	• Identifica os atributos de dados de um modelo e os relacionamentos entre seus atributos	• Identifica os diferentes níveis multidimensionais em uma dimensão, formando uma hierarquia
Baixo nível	• Chaves e características físicas • Implementação em um SGBD específico	• Mostra o grupo de atributos de um nível multidimensional e as células de fato • São os atributos de classificação e as medidas, respectivamente

Quadro 5 - Níveis de detalhe dos modelos de dados tradicional e multidimensional.

Nível de detalhe	Assunto da análise	Dimensão de análise
Mais alto	• Fatos (representando os tipos de fatos)	• Dimensões
Intermediário	• Cubos (representando a classe da célula)	• Níveis de agregação
Mais baixo	• Medidas (agrupadas nas células que correspondem aos Fatos)	• Instâncias dos níveis de agregação

Quadro 6 - Resumo dos diferentes elementos da modelagem multidimensional.

Fonte: Resumo adaptado de Abelló et al., 2001b.

2.8 Persistência de dados

Um sistema de banco de dados é comumente projetado para gerir grandes volumes de informações, considerando que o gerenciamento implica na definição das estruturas de armazenamento e nos mecanismos para manipulação de tais informações (Silberschatz et al., 1999).

Os dados em um banco de dados são comumente referenciados como persistentes, ou seja, não são transitórios, pois uma vez aceitos pelo SGBD para entrada no banco de dados, eles só podem ser removidos por alguma requisição explícita ao SGBD (Date, 2003).

Os modelos de banco de dados utilizados para a implementação de um DW podem ser baseados no modelo relacional, multidimensional ou orientado a objetos, conceituados a seguir.

2.8.1 Banco de dados relacional

O modelo de dados relacional estabeleceu-se como o primeiro modelo de dados para aplicações comerciais (Silberschatz, 1999, p.61). Um banco de dados relacional consiste em uma coleção de tabelas relacionadas entre si.

A visão relacional de dados oferece um meio de descrever dados apenas com sua estrutura natural, fornecendo uma base para uma linguagem de dados de alto nível com um rendimento máximo de independência de programas ou equipamentos (Cood, 1970). E segundo uma conceituação alternativa (Machado, 2004, p.41):

> A abordagem relacional está baseada no princípio de que as informações em uma base de dados podem ser consideradas relações matemáticas e que estão representadas de maneira uniforme com o uso de tabelas bidimensionais.

Na maioria das implementações relacionais do modelo Estrela, hierarquias são tabelas externas de referência além da tabela de fatos. Isto ocorre porque existe uma separação de hierarquias e tabelas de fato no esquema Estrela, e os sistemas geralmente têm *interfaces* e mecanismos internos diferentes para administrar as hierarquias e a tabela de fatos. Isto determina uma quantia significante de junções de tabelas consultadas em tempo de execução (Wan, 2004).

2.8.2 Banco de dados multidimensional

O banco de dados multidimensional é um BD que dá suporte e otimiza manipulações matemáticas, financeiras, estatísticas e de tempo, além de somatórios de valores referentes aos níveis de uma hierarquia de dados. Sua estrutura de dados é baseada nas hierarquias das dimensões e na hierarquia das medidas (Machado, 2000, p. 129).

A tecnologia multidimensional não é suficiente para suporte à decisão, pois apresenta limitações para que possa vir a substituir a tecnologia relacional, como por exemplo (Singh, 2001, p. 109):

- as ferramentas de gerenciamento de banco de dados devem oferecer, além de recursos para garantir a integridade referencial, utilitários para o gerenciamento do desempenho e segurança;
- como o detalhe da transação não é armazenado em bancos de dados multidimensionais, é difícil acessar detalhe em nível de linha, pois é necessária uma API diferente;
- um banco de dados multidimensional deve permitir atualização incremental de dados e recursos de inserção e não restringir o uso dos dados existentes enquanto novos dados são carregados, como a tecnologia relacional;
- os BDs multidimensionais não suportam múltiplos *arrays* e *joins* de vários *arrays* relacionados em uma única estrutura de banco de dados.

Alguns problemas dos BD multidimensionais podem ser observados (Trujillo et al., 2001):

- a modelagem depende estritamente da implementação correspondente (ROLAP ou MOLAP) e inexiste um modelo conceitual padrão (independente dos detalhes da implementação) para modelar banco de dados multidimensionais;
- os requisitos propostos para uma subseqüente análise de dados necessitam considerar os detalhes da organização física dos dados, mais que seus aspectos lógicos.

Tais problemas poderiam ser resolvidos com a proposta de um modelo conceitual genérico, independente dos detalhes de implementação.

A tecnologia de armazenamento multidimensional pode ser atraente, visto que oferece uma visualização multidimensional dos dados, entretanto, os dados podem ser armazenados de forma relacional (ou em outro formato) e ainda assim serem visualizados multidimensionalmente (Singh, 2001).

2.8.3 Banco de dados Orientado a Objetos

Sistemas de bancos de dados orientados a objeto (SBDOO) podem armazenar dados na forma de objetos permanentes, sendo utilizados para armazenar tanto os atributos como os métodos dos objetos.

O BDOO tem sua aplicação adequada (Singh, 2001, p. 234):

> Bancos de objetos são adequados para ferramentas que usam o modelo de desenvolvimento OO em sua forma pura, e também são o local adequado para estruturas de dados que incluem o armazenamento de repositório de informações binárias.

Quando Atkinson et al. (1989) escreveram o primeiro manifesto, expuseram que não existiam regras para a definição de um banco de dados orientado a objetos. Assim, foram elaboradas estas regras e se um sistema de banco de dados obedece a elas, pode ser chamado de SBDOO. Neste manifesto seus autores consideraram que os bancos de dados relacionais, como sistemas legados, não são adaptáveis às novas aplicações.

As regras para a definição de um BDOO foram separadas em três grupos: obrigatório, opcional e aberto, enfatizando principalmente as propriedades da orientação a objetos e, incluindo entre outros conceitos, o suporte a objetos complexos, identidade de objetos e encapsulamento.

O segundo manifesto, apresentado pelo *Committee for Advanced DBMS Function* (1991) apresentou seu próprio conjunto de regras, considerado uma oposição ao primeiro manifesto. Enfatizou os sistemas relacionais que tinham incluído dois desenvolvimentos principais (acesso não processual e independência de dados), não concordando em se abandonar tais sistemas.

O terceiro manifesto, de Darwen e Date (1995), mais formal e técnico que os dois anteriores, apresentou um sistema fortemente baseado no modelo relacional, não concordando com a posição de Atkinson et al. (1989) de tentar ignorar o modelo relacional. O sistema propôs estender o modelo relacional permitindo a definição de novos tipos de dados como domínios.

O objetivo inicial dos bancos de dados de objetos consistia no armazenamento e manipulação de objetos. Esta meta originou-se pela adoção difundida de técnicas de modelagem e linguagens orientadas a objeto. A primeira força de um banco de dados de objetos é sua habilidade *in-built* para administrar modelos complexos (em termos de tipos) com relacionamentos complexos. Os objetos consistem em atributos simples valorados (*integers, strings*), atributos multivalorados (*arrays* dinâmicos) e estruturas complexas, e seu gerenciamento é fundamental. Mas é a habilidade para controlar relacionamentos que o diferencia, não apenas

relacionamentos um-para-um ou um-para-muitos, mas aqueles que incluem semântica, como grupos, listas e mapeamentos (Versant, 2001).

Assim como o modelo ROLAP, os bancos de dados orientados a objetos já possuem alguns padrões definidos pelo *Object-Oriented Database Management Group* - ODMG, seguindo os trabalhos realizados pelo *Object Management Group* – OMG (OMG, 2005).

2.8.4 Vantagens na utilização de BDOO

Provavelmente a vantagem mais importante conceitual (multidimensional ou não) de um modelo OO é que o resultado é mais próximo à concepção do usuário pensar. Todo objeto ou classe modelada terá uma correspondência com alguma entidade real, tornando-o facilmente compreendido. Pode-se encontrar outros benefícios, além dos abstratos, no uso do paradigma OO, como o uso de Identificadores de Objeto (OIDs) que resolvem o problema de identificação pelo uso de chaves; a permissão para não usar a Primeira Forma normal (1FN), implicando na utilização de objetos que contêm valores não atômicos.

Uma das principais diferenças e, conseqüentemente, vantagens da orientação a objetos é que os dados e a funcionalidade estão associados na forma de um objeto (Ambler, 1998). Além da implementação de heranças múltiplas, a manutenção de relacionamentos inversos apresenta-se como uma característica de bancos de dados orientados a objetos (Urban e Dietrich, 2003).

O uso de um SBDOO ou o uso de persistência de objetos como a implementação física de um sistema O3LAP, permite que se consiga atingir as vantagens do modelo ROLAP e MOLAP com poucas de suas desvantagens (Buzydlowski et al., 1998).

Considerando que BDOO são bem adaptados para armazenar dados semi-estruturados e não estruturados, um DW OO é considerado um meio interessante para integrar dados de heterogêneas fontes de dados, observando que os BDOOs têm muitas características únicas como identidade de objeto, atributos complexos, referência inter-objeto e herança de classe (Chao, 2004).

O modelo de dados OO representa uma entidade do mundo real como uma unidade de objeto única, que tem ambas as propriedades: estrutural e comportamental. O estado do objeto deste modelo apenas se altera através de métodos (Lee et al., 2005), enquanto o modelo relacional gerencia todos os tipos de

dados como atributos ou associações, relegando o aspecto comportamental ao aplicativo.

Bancos de dados relacionais são basicamente conjuntos de tabelas, que são *arrays* de células. As colunas definem a estrutura da tabela e as linhas contêm os dados. Bancos de dados relacionais apenas suportam tabelas como listas não ordenadas e podem recuperar uma lista ordenada apenas se um índice for especialmente definido. Portanto, o projeto do modelo de dados deve ser descrito considerando tais conceitos (Versant, 2004a). Um banco de dados orientado a objetos não apresenta problemas com uma lista ordenada e não precisa de índices - os índices são artifícios gerados por causa dos limites de estruturas de dados do modelo relacional (Bloor e Bloor, 2004).

Além disso, uma outra vantagem é observada na identificação dos objetos armazenados no banco de dados. Enquanto no modelo orientado a objetos, é gerado automaticamente um identificador único para cada objeto (OID), que não pode ser mudado, o modelo relacional necessita da geração de uma chave primária, que consiste de um ou mais campos da relação. Enquanto o OID é um identificador, a Pk pode conter informações específicas da aplicação e, portanto, ser alterada pelo usuário.

Nos modelos conceituais orientados a objetos e representados através da UML é comum o diagrama de classes apresentar a existência de classes abstratas. No modelo relacional faz-se necessária uma avaliação do mapeamento para uma ou mais tabelas, gerando uma reengenharia do processo, enquanto no modelo orientado a objetos, as classes abstratas são mapeadas e representadas diretamente.

No modelo relacional, os esquemas com classes e superclasses devem ser mapeados para atributos específicos da classe e da superclasse. Uma solução para a implementação é colocar na classe todos os atributos da superclasse, ou ainda, armazenar na tabela da superclasse apenas atributos da superclasse e na classe serão armazenados os atributos específicos da classe, sendo necessária a utilização de junções. No modelo OO a passagem é direta do modelo representado em UML para o banco de dados OO (Badia, 2005).

Na prática, banco de dados de objetos tem vantagens significativas sobre os bancos de dados relacionais, pois tipicamente (Bloor e Bloor, 2004):

- executam mais rapidamente aplicações transacionais;

- trabalham com objetos complexos mais efetivamente;
- oferecem maior produtividade ao desenvolvedor;
- são mais facilmente gerenciáveis.

Os SBDOOs oferecem um paradigma para suportar objetos de forma completa na camada de persistência (Versant, 2004a). Em alguns casos, bancos de dados de objeto estão substituindo bancos de dados relacionais por razões de desempenho. A principal vantagem de desempenho dos bancos de dados de objeto é que eles normalmente não têm que agrupar os dados antes de usá-los, como o banco de dados relacional faz. Eles tendem a armazenar dados em sua forma mais usada que tipicamente auxilia no desempenho (Bloor e Bloor, 2004).

Uma comparação das técnicas do modelo relacional e de objetos é mostrada no quadro 7, com os conceitos de abstração de dados, herança e encapsulamento, apontando os benefícios do modelo de objetos (Barry, 2004).

Conceitos do modelo de objetos	Modelo relacional	Modelo de objetos	Benefícios do modelo de objetos
Abstração de dados	Entidades de intersecção e indexação para representar referências entre tuplas	OIDs para representar diretamente as referências entre objetos	Esquemas mais simples para representar dados complexos
Herança	Tipo codificado e programado	Hierarquias de classe	Representação direta das referências entre tipos e subtipos e suporte para a especialização de cada subtipo
Encapsulamento	Tipo codificado e programado, usualmente com bibliotecas	Fornece encapsulamento embutido para assegurar a execução do código correto nos dados corretos	Reduz o código de aplicação e a chance de erro de execução do código errado em dados corretos

Quadro 7 - Comparação do modelo relacional e de objetos.

Fonte: Adaptado de Barry, 2004.

2.8.5 Persistência de objetos

A idéia básica de um SBDOO é armazenar diretamente qualquer modelo complexo de dados em um banco de dados, sem limitações para os conceitos mantidos pela orientação a objetos (Versant, 2004a).

Quando se aborda a questão de persistência de dados, os problemas de concorrência, desempenho e escalabilidade devem ser observados durante as fases de análise e projeto, pois são exigências do mundo real que precisam ser refletidas na implementação. As aplicações atuais deixam de ser puramente aplicações de dados, e, especificamente, no ambiente de DW, a análise e o projeto do modelo devem ser cuidadosamente definidos.

Considerando que parte da força de um banco de dados de objeto vem de sua habilidade para apoiar relacionamentos complexos, permitindo a navegação e execução de manipulações complexas de dados, é essencial que se assegure que o modelo de objeto inclua caminhos de navegação ditados pelas transações do sistema.

Um armazenamento persistente coloca grande importância na definição de relacionamentos, desde que a navegação é o único meio de acesso. Se um objeto não pode ser obtido pela navegação, então deve ser desconsiderado. Esta abordagem dá uma grande ênfase em manter coleções de objetos para suportar os padrões de acesso exigidos.

No processo OO falta um detalhe importante: persistência dos dados. As estruturas de dados são, por definição, orientadas a objeto. Se o SGBD determinado também é orientado a objeto, o processo de mapear as classes persistentes da aplicação para classes persistentes de um SBDOO é direto. Porém, se o SGBD é relacional, o que mais comumente ocorre, são necessárias conversões para o mapeamento. Portanto, as técnicas tradicionais de modelagem de dados para gerar modelo lógico e modelo físico de dados são empregadas freqüentemente. Mas, em muitas situações, são apenas mapeadas classes OO para tabelas (Becker, 2001; Shah e Slaughter, 2003).

Diante disso, sistemas tradicionais de banco de dados não são apropriados para a análise multidimensional, analisando que são concebidos para aplicações OLTP, que possuem um grande número de transações concorrentes que envolvem um número pequeno de registros (Trujillo et al., 2001).

2.9 Mapeamento de dados

Na concepção de um sistema de informação, um modelo conceitual é desenvolvido como parte da especificação de requisitos da qual um modelo de

dados será derivado mais tarde, na fase de projeto. Assim, mapeamento entre modelos conceituais e modelos de dados é uma transformação vital no desenvolvimento de um sistema de informação (Badia, 2005).

O processo de mapeamento de dados valida, define e expande o modelo de dados da empresa. Algumas questões que devem ser tratadas durante o mapeamento de dados (Singh, 2001, p.100):

- integração de dados ao longo de aplicações: a maioria das organizações possui redundância de dados, portanto, estes elementos devem ser pesquisados e ajustados;
- resolução de idiossincrasias de aplicações legadas: faz-se necessária a participação de especialistas em aplicações no processo de identificação e desenvolvimento do projeto, auxiliando no mapeamento de dados nos vários tipos de registros, instruções e valores padrão.

2.9.1 Etapas de um mapeamento de dados

Para projetar o modelo de dados do DW, recomenda-se a utilização do modelo corporativo de dados, pois este modelo possui todos os atributos necessários do registro dos dados operacionais da empresa, com as alterações necessárias (Oliveira, 1998, p.74):

- remoção dos dados meramente operacionais: aqueles atributos que não são utilizados no processo de tomada de decisão, ou seja, usados apenas no ambiente operacional, não devem ser incluídos no DW;
- adição de um elemento de tempo na estrutura da chave: se os atributos chaves ainda não possuírem um elemento temporal, este deve ser incluído;
- introdução de dados derivados: os dados derivados que serão usados constantemente devem ser incluídos no DW, de forma que sejam calculados uma única vez;
- transformação de relacionamento entre dados em artefatos dos dados: estes artefatos podem incluir chaves estrangeiras ou atributos de tabelas associadas, considerando a existência de muitos valores para um dado relacionamento entre tabelas, pelo armazenamento de dados históricos;

- acomodação dos diferentes níveis de granularidade: o nível de granularidade do modelo operacional pode ou não ser o mesmo do DW, porém, quando este nível se altera, o DW deve representar esta mudança;
- união dos dados comuns de diferentes tabelas: considerar a possibilidade de se unir duas ou mais tabelas do modelo operacional em uma única tabela do DW, observando que estas tabelas devam compartilhar uma chave ou parte dela, que os dados das tabelas normalmente seja utilizados conjuntamente e que seu padrão de inserção seja o mesmo;
- criação de *arrays* de dados: gerar grupos repetitivos de dados, observando que o seu número de ocorrências seja previsível, as ocorrências sejam utilizadas juntas e que os padrões de inserção e remoção dos dados sejam estáveis.
- separação dos atributos de dados de acordo com sua estabilidade: agrupando atributos de dados segundo sua propensão a alterações, gerando grupos que apresentam características semelhantes;
- modelo de estrutura de dados: cada DW é uma estrutura de dados denominada “instantâneo”, que é gerado a partir de um evento, normalmente aleatório, ou de períodos regulares de tempo.

Alguns autores sugerem que se utilize o modelo operacional de dados para se chegar ao modelo multidimensional (Phipps e Davis, 2002; Freitas et al., 2002; Peralta et al., 2003). Porém, observa-se que o modelo mais utilizado para se projetar o DW é o modelo Estrela, que possui uma estrutura diferente do modelo ER (Oliveira, 1998).

Considerando que os dois modelos representam uma visão dos dados da empresa, sob perspectivas diferentes, mapeamentos entre os dois modelos são propostos, de forma que os modelos ER corporativos possam ser aproveitados para gerar o modelo multidimensional. Os trabalhos de Hitchman (2004a, 2004b, 2004c) examinam a relevância do modelo ER para a modelagem de dados prática.

2.10 Considerações finais

O presente capítulo abordou os principais conceitos da modelagem de dados, relacionando o modelo ER e o modelo multidimensional.

Uma das principais razões para a complexidade dos bancos de dados operacionais é o uso da normalização, pois tal processo gera muitas tabelas, e até mesmo consultas simples irão requerer múltiplas junções. Uma das maiores diferenças entre o Modelo ER e o Modelo multidimensional recai sobre esta questão. A modelagem ER implementa tabelas utilizando o processo de normalização, enquanto na modelagem multidimensional, este processo é revertido.

No presente capítulo foram apresentados os principais modelos de representação do modelo multidimensional, caracterizado basicamente pelo modelo Estrela e Floco de Neve. Outras propostas foram apresentadas como variações destes modelos.

Complementando as diferenças entre o ambiente de DW e o ambiente operacional, observa-se que a atualização dos dados, que ocorre no ambiente operacional e não no ambiente de DW, não implica em redundância de dados, pela estrutura proposta pelo modelo Estrela.

Durante a fase de modelagem de dados no ciclo de vida do DW, projeta-se o resultado da análise contextualizada do negócio que se pretende modelar. São identificados os fatos e as dimensões, com respectivas medidas e granularidade. Este processo assemelha-se a definição das entidades, relacionamentos e atributos do modelo ER.

Observa-se, portanto, que o modelo ER é uma ferramenta que pode auxiliar na análise de requisitos do negócio e no projeto da estrutura de dados resultante. Em contrapartida, o modelo multidimensional fornece uma melhor visualização das questões abstratas para atender os usuários finais, proposta do ambiente de DW.

No próximo capítulo são discutidos aspectos da modelagem multidimensional voltada ao paradigma da orientação a objetos.

3 MODELO MULTIDIMENSIONAL ORIENTADO A OBJETOS

3.1 Introdução

Para que os objetivos propostos neste trabalho possam ser alcançados, faz-se necessário conhecer os componentes do paradigma orientado a objetos e sua representação na UML, aplicados à modelagem multidimensional, como propõe a segunda fase da metodologia da pesquisa.

O paradigma OO oferece um modelo conceitual genérico com total independência dos aspectos físicos para a modelagem conceitual multidimensional, de modo mais simples e natural que os modelos anteriormente propostos (Trujillo e Palomar, 1998). Portanto, este capítulo busca registrar este paradigma através dos aspectos implícitos desta abordagem, discutindo e comparando os modelos até então propostos.

O modelo orientado a objetos mantém a abstração de dados estática usando classes de objetos e modela o comportamento do sistema através de métodos. A abordagem orientada a objetos além de ser utilizada para a modelagem conceitual de dados, também é aplicada em banco de dados orientado a objetos, pois existe uma transição direta de um modelo de dados OO para sua implementação em um BDOO. Portanto, existe uma tendência que se utilize a orientação a objetos em substituição ao modelo ER como modelo de modelagem de dados (Balaban e Shoval, 2002).

A segunda seção apresenta os conceitos básicos da orientação a objetos, que são complementados na terceira seção, com seus principais componentes, que serão utilizados no estudo dos modelos existentes e também na proposta deste trabalho.

Na quarta seção é apresentada a linguagem UML como um padrão para a representação do paradigma orientado a objetos, referenciando especificamente os diagramas de classe e de estrutura composta.

A multidimensionalidade através da UML é abordada na quinta seção, em que são relacionadas as características do nível estrutural desta linguagem para a representação no modelo multidimensional.

3.2 Conceitos básicos da orientação a objetos

A orientação a objetos é um paradigma de modelagem de software, abrangendo análise, projeto e programação. Na programação orientada a objetos as aplicações são compostas por várias classes, que contêm procedimentos (métodos) e tipos. Os dados dos programas são armazenados em objetos, criados durante a execução das aplicações, e não durante a declaração das classes. Os objetos executam ações e se comunicam através de mensagens e são compostos por variáveis de instância e métodos. As variáveis de instância armazenam os dados dos objetos, definindo sua estrutura, enquanto os métodos definem o comportamento do objeto (Nassu e Setzer, 1999).

Os conceitos básicos da orientação a objetos para banco de dados são assim definidos (Nassu e Setzer, 1999):

- Herança: mecanismo no qual uma classe é definida a partir de uma outra classe, herdando suas variáveis de instância e métodos. A classe gerada é chamada de subclasse. A herança pode ser única, herdando características de apenas uma classe, ou múltipla, herdando características de várias classes.
- Polimorfismo: pode ser definido como um mecanismo que permite que o nome de um mesmo método possa ser definido em várias classes, com implementações diferentes em cada uma delas.
- Encapsulamento: proteção da estrutura interna do objeto pelos métodos, gerando uma maior independência de dados, considerando que sua implementação não é conhecida por quem utiliza os objetos.
- Mensagens: os objetos de uma aplicação comunicam-se através de mensagens. Um objeto envia uma mensagem para modificar o estado ou conseguir uma informação de um outro objeto, que executa um método, de acordo com a mensagem recebida.

Na maioria das linguagens de objetos, atributos e métodos podem ser privados, públicos ou protegidos (*private, public, protected*). Porém, usualmente, em um projeto de objetos, cada atributo deve ser protegido para forçar o encapsulamento. Métodos públicos definem a *interface* de uma classe (seu protocolo ou o conjunto de métodos que outros objetos podem chamar) (Versant, 2004a).

A utilização da abordagem orientada a objetos, comparada a abordagens tradicionais, proporciona alguns benefícios (Becker, 2001):

- maior reusabilidade;
- especificação mais detalhada das restrições de sistema e processos;
- distinção mais clara entre “o que é o problema” (análise) e “como o problema será resolvido” (projeto e desenvolvimento).

3.3 Componentes da orientação a objetos

Os conceitos da orientação a objetos são implementados pelas classes e objetos, que são os principais componentes da orientação a objetos.

3.3.1 Classe

Classe é “uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamento e semântica” (Booch et al., 2000, p. 49).

Uma classe abstrata é uma classe que não tem instâncias. Ela existe principalmente para representar uma abstração comum que diversas subclasses compartilham.

A característica de hierarquias de classe, introduzindo relacionamentos entre entidades “*IS-A*" (classe/subclasse), permite ao projetista reconhecer membros de associação entre entidades; normalmente significando que existem atributos compartilhados. Atributos compartilhados são removidos e agrupados em uma nova entidade (classe) que é uma generalização das outras, e um relacionamento classe-subclasse é criado. Com uma abordagem OO, assume-se a herança de atributos (Badia, 2005).

Uma superclasse é uma classificação de objetos contendo outros objetos. A representação gráfica da superclasse é dividida em três partes: a superior nomeia a superclasse, a intermediária especifica os atributos da superclasse, considerando que todas as classes subordinadas a uma superclasse herdam estes atributos e, finalmente, a última parte, que especifica os comportamentos que podem ser executados pelas classes subordinadas a esta superclasse, que são os métodos.

As características dos objetos são consideradas seus atributos. Quando os objetos são agrupados considerando estes atributos, estão sendo criadas classes de

objetos, ou seja, uma classe é a descrição de um conjunto de objetos com propriedades comuns. Enquanto uma classe é uma abstração, um objeto é uma manifestação concreta desta abstração (Booch et al. 2000, p. 29).

As divisões da classe são as mesmas encontradas na superclasse. Os atributos e comportamentos especificados em uma determinada classe objeto estão relacionados apenas àquele objeto e não afetam outros objetos na superclasse. Se a classe especifica um atributo ou comportamento que tem o mesmo nome de um atributo ou comportamento na superclasse, as especificações da subclasse anulam aqueles da superclasse. Para modelar uma classe como subordinada à superclasse, utiliza-se o objeto subclasse.

Existem duas distinções importantes entre classes e superclasses. Uma superclasse é um nível mais elevado de abstração. Objetos similares são agrupados em uma classe definindo um grupo de atributos e comportamentos que são compartilhados por todas as classes subordinadas. As classes subordinadas herdam os atributos e comportamentos da superclasse. Outra diferença entre classes e superclasses é que podem existir instâncias reais de classes, enquanto não existem para as superclasses.

Em algumas situações, é necessária a geração de uma classe base da qual outras classes possam ser derivadas futuramente, caracterizando uma classe abstrata, cuja existência é particularmente útil para exibir um modelo de derivação (Matos, 2002).

3.3.2 Objetos

Objeto é “alguma coisa geralmente estruturada a partir do vocabulário do espaço do problema ou do espaço da solução”. Os objetos têm uma identidade, um estado e um comportamento, são instâncias das classes (Booch et al., 2000, p. 11).

Um objeto pode ser definido como qualquer coisa; o mundo é constituído por objetos e não são limitados pelo mundo tangível, pois um objeto pode também ser um conceito (Giovinazzo, 2000, p. 49). Considerando as organizações como o foco da visão dos objetos, um serviço é um objeto. Portanto, pode-se ver como a organização funciona quando associamos os procedimentos aos objetos.

Um objeto possui um estado interno e um comportamento. O estado interno é composto por variáveis que podem armazenar dados e modificado ao longo da vida

do objeto. O comportamento é um conjunto de ações predefinidas, que são os métodos, com as quais os objetos respondem às mensagens enviadas por outros objetos (Nassu e Setzer, 1999).

Todo objeto pode ter um estado, que é composto dos valores dos atributos e o status das operações que o objeto executou. Um método que o objeto executa ou a mudança do valor de algum atributo automaticamente muda o seu estado anterior (Matos, 2002). Um método tem sua assinatura, que consiste dos tipos de parâmetros e os próprios parâmetros que são passados ao método, além do nome e do tipo de retorno do método (Cardoso, 2003).

3.3.3 Atributos

Um atributo é um valor de dados guardado pelos objetos de uma classe. Cada atributo possui um valor para cada instância dos objetos. Diferentes instâncias de objetos podem ter valores iguais ou diferentes para um dado atributo. Cada nome de atributo é único dentro de uma classe (Rumbaugh et al., 1994).

Uma classe pode ter vários atributos ou até mesmo nenhum atributo. Um atributo representa alguma propriedade do item que está sendo modelado, que é compartilhado por todos os objetos da classe. Em um determinado momento, um objeto de uma classe terá valores específicos para cada um dos atributos de sua classe (Booch et al., 2000).

Os atributos são mostrados na segunda parte do quadro de uma classe. Cada nome de atributo pode ser seguido de detalhes opcionais, como tipo e valores *default*, que definem o nível de detalhamento desejado.

Na definição de um atributo pode-se incluir, além de seu nome e tipo de dados, seu valor inicial, visibilidade e outras características, sendo que o nome do atributo é o único requisito obrigatório. A visibilidade do atributo pode ser representada pelas palavras-chaves *public, protected, private* ou *package* (ou por seus ícones +, #, -, ~) (Melo, 2004; OMG, 2005).

3.3.3.1 Identificador de objeto

Quando um objeto é criado, ele recebe um identificador (OID), gerando a característica de unicidade para aquele objeto, diferenciado-o de todos os demais

objetos. A identidade do objeto corresponde ao conceito de chave primária na modelagem ER (Muller, 2002).

3.3.4 Relacionamentos

O modelo de objetos, além de apresentar os objetos individuais também mostra como os objetos se relacionam entre si. Podem ser considerados três tipos principais de relacionamentos na UML: dependência, associação e generalização (Booch et al., 2000), e ainda a agregação e composição, como variação do relacionamento de associação (Melo, 2004).

As dependências representam relacionamentos de utilização entre as classes, as associações representam relacionamentos estruturais entre objetos, e as generalizações relacionam classes generalizadas a outras mais especializadas, conhecidos como relacionamentos subclasse/superclasse ou filha/mãe, conforme ilustra a figura 7. Estes relacionamentos são itens estáticos, que se apresentam, normalmente, em diagramas de classe da UML.

Mapeando os tipos de relacionamento para os operadores de relacionamento no modelo de objetos temos: exatamente-um, um-para-muitos, zero-ou-um e zero-para-muitos (Giovinazzo, 2000, p. 102).

Nos relacionamentos que são exatamente-um, uma instância do primeiro objeto deve se relacionar com uma e apenas uma instância de outro objeto. Em um relacionamento que é um-para-muitos uma instância de um objeto pode se relacionar com muitos objetos de outro tipo, mas deve se relacionar ao menos com um. Estes relacionamentos não são opcionais. Relacionamentos de objeto opcionais são aqueles: zero-ou-um e zero-para-muitos. A cardinalidade zero indica que o relacionamento entre duas instâncias destes objetos não é obrigatório.

3.3.4.1 Associação

A associação é um relacionamento em que se representa a conexão de um objeto a outro, ou seja, é um relacionamento que conecta duas (associação binária) ou mais classes (associação ternária ou de ordem-n), demonstrando a colaboração entre as instâncias de classe (Melo, 2004), conforme mostra a figura 7.

No entanto, outro tipo de associação é a agregação, que é uma pura associação entre duas classes, considerando que em alguns casos, a modelagem representa

um relacionamento "todo-parte", no qual uma classe representa um item maior, formado por itens menores (Booch et al., 2000; Barbier et al., 2001, Abelló et al., 2000b; Giovinazzo, 2000; Fowler, 2000). Uma agregação é um objeto que é composto de outros objetos. O símbolo de agregação mostra quais objetos são combinados para compor a agregação. Os objetos agregados são características de um novo, que podem ter uma existência dependente também, mas não é uma implicação da existência da própria ligação, como os vários departamentos de uma empresa da figura 8a.

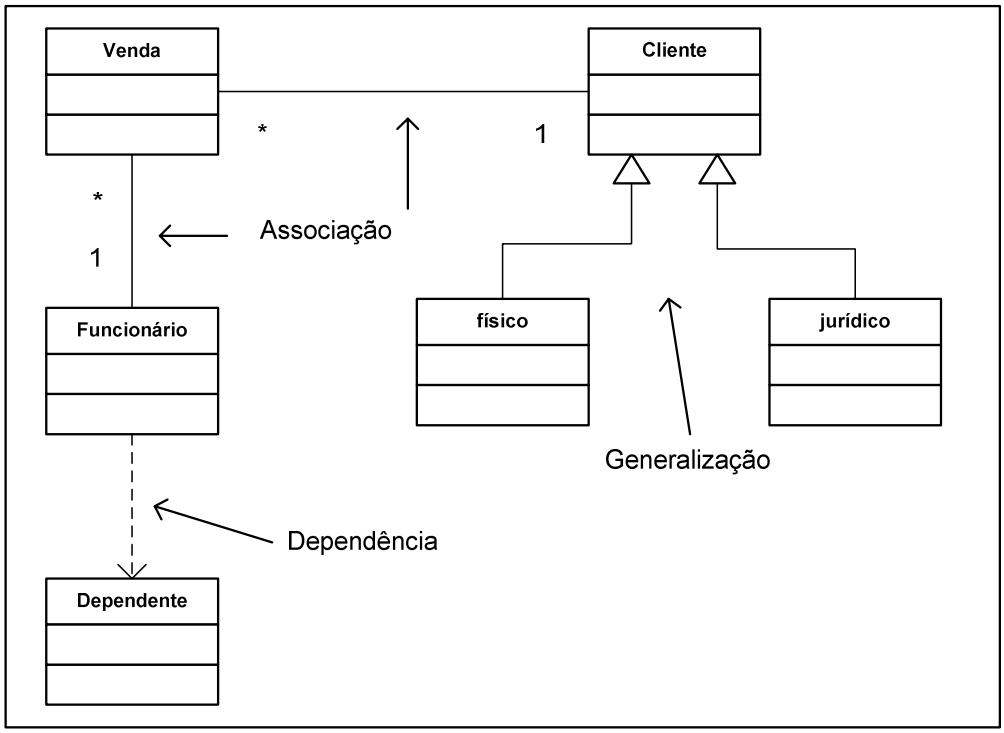

Figura 7 - Relacionamentos de generalização, associação e dependência.

Porém, o conceito de agregação na UML não é suficientemente suportado. A agregação pode desprezar a complexidade do esquema por considerar vários objetos como uma única unidade. Na UML, a agregação apenas define a semântica abstrata, não comportamentos (Lee et al., 2005).

As agregações podem ser com a existência de dependência e com a existência de independência. No primeiro caso, a existência do objeto “parte” é completamente dependente da existência de um objeto “todo” e caso o “todo” seja excluído, todas as “partes” também serão excluídas. No segundo caso, a existência do objeto “parte” é

independente, ou seja, se o objeto “todo” for removido, ainda assim, as partes podem existir (Rahayu et al., 2001).

Além da agregação, a UML oferece uma variação, chamada de composição. Na composição, se o novo objeto é concebido como composto por outros que são suas partes, isso é chamado de relação “todo-parte” e implica na existência de uma dependência entre ambos os lados do relacionamento. O objeto “parte” pode pertencer a somente um “todo”; além disso, espera-se que as “partes”, geralmente, tenham sua existência totalmente relacionada com o “todo”. Normalmente, qualquer remoção do “todo” é considerada como tendo um efeito cascata nas “partes”, como mostra a relação entre Pedido e Item_Pedido da figura 8b.

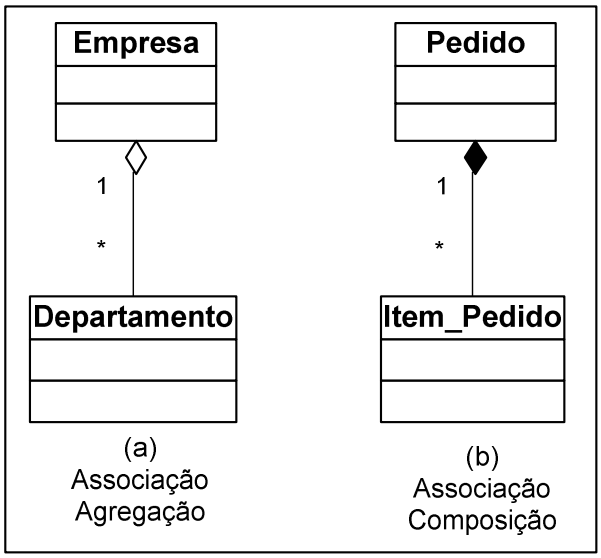

Figura 8 - Associação agregação e associação composição.
Fonte: Adaptado de Booch et al. (2000, p.66).

3.3.4.2 Classificação

Toda instância é relacionada a pelo menos uma classe no esquema através de relacionamento. Todas as instâncias compartilham alguns atributos e representam conceitos relacionados que se agrupam em uma determinada classe. Da mesma maneira todos os elementos em um esquema (classes, relacionamentos) que representam conceitos relacionados em um modelo de dados são agrupados em uma metaclasse. Portanto, todas as metaclasses podem se agrupar em exatamente uma metaclasse, que é sua própria instância (Abelló et al., 2000b).

3.3.4.3 Generalização / Especialização

A ligação da Generalização/Especialização relaciona duas classes (ou metaclasses). Uma destas classes tem um significado mais específico que a outra. A classe mais geral é chamada superclasse e a mais específica, referenciada como subclasse. Como conseqüência deste tipo de ligação, obtém-se a herança, conforme mostra a figura 7. A subclasse herda as propriedades e métodos de sua superclasse (ou superclasses). Se for permitido ter mais de uma superclasse, pode-se ganhar heranças múltiplas. Cada classe terá além de seus próprios atributos, os atributos e relacionamentos de cada uma de suas superclasses.

Uma classe generalizada herda operações e atributos da classe superior, não sendo necessário refazer métodos. Porém, há situações em que métodos ou atributos podem ser redefinidos, caracterizando o polimorfismo.

A generalização significa um relaciomento "*IS-A*". Se um objeto A1 tem um relacionamento "*IS-A*" com um objeto A2, todos os membros do objeto A2 também se tornam membros do objeto A1. Usualmente o conceito de generalização é suportado diretamente pela maioria das linguagens e sistemas de banco de dados orientados a objetos (Lee et al., 2005).

3.3.5 Operações

Uma outra forma de interação entre os objetos é quando, através de um estímulo, um determinado objeto provoca uma reação em outro objeto. Quando um objeto reage a um estímulo, a reação é definida como o comportamento do objeto. Um comportamento é uma resposta previsível a um estímulo (Giovinazzo, 2000).

Uma operação é a implementação de um serviço que pode ser solicitado por algum objeto da classe para modificar o comportamento, ou seja, é uma abstração de algo que pode ser feito com o objeto e compartilhado por todos os objetos da classe. Uma classe pode ter várias operações ou nenhuma (Booch et al., 2000).

No modelo de implementação pode-se identificar a visibilidade da operação, sendo pública, protegida ou privada. Com modelos conceituais, não se deve usar operações para especificar a *interface* de uma classe (Fowler, 2000).

Uma classe aceita certos tipos de mensagens das instâncias de outras classes, que ativam a execução de métodos (consultas, atualizações, cálculos, etc.).

Entidades relacionais representam tabelas passivas que contêm dados e, considerando que não são objetos reais, são independentes de comportamento. A inclusão de métodos no modelo de dados ajuda modelar o comportamento juntamente com os dados. Métodos facilitam a implementação de funções de agregação complexas (Abelló et al., 2000b). Entende-se que o modelo de DW está completo quando o modelo de objetos contém todos os atributos e comportamentos que são relevantes para o projeto (Giovinazzo, 2000).

A UML faz uma diferença entre operação e método: uma operação especifica um serviço que pode ser solicitado por qualquer objeto da classe, enquanto um método é a implementação de uma operação (Fowler, 2000).

3.4 A Linguagem UML

A UML (*Unified Modeling Language*) é uma linguagem padrão para visualizar, especificar, construir e documentar projetos de software, desenvolvida pela associação de três outras metodologias: Booch, OOSE de Jacobson e OMT de Rumbaugh, com o objetivo de criar uma linguagem unificada de modelagem, buscando oferecer um padrão para a metodologia orientada a objetos (Booch et al., 2000). No ano de 1997 a OMG - *Object Management Group* - tornou a UML uma linguagem de modelagem padrão para a orientação a objetos (Matos, 2002).

Apesar de ser considerada padrão, estudos abordam considerações sobre a complexidade e extensão da UML, além de sua imprecisão semântica, sua implementação de modo não padronizado, além de não apresentar suporte adequado para desenvolvimento baseado em componentes, e a dificuldade de intercâmbio entre modelos de diagramas (Halpin, 2002; Siau e Cao, 2002; Grossman et al., 2005; Siau et al., 2005).

A UML 2.0 apresenta treze diagramas, incluindo diagramas estruturais ou estáticos: de classes, de objetos, de componentes, de implantação e os novos diagramas de pacotes e de estrutura composta; e os diagramas dinâmicos: de caso de uso, de atividades, de seqüência, de estados, de comunicação (denominado anteriormente de colaboração), e os novos diagramas de visão geral e temporal (Ambler, 2005).

A UML abrange características da modelagem que não se encontram no modelo ER, como a representação de classes abstratas, agregações e comportamento

(Urban e Dietrich, 2003). Enquanto há vinte anos o modelo ER apresentou-se como o padrão para a modelagem conceitual de banco de dados, nos últimos anos a UML vem sendo disseminada e tornando-se um padrão para a modelagem em várias aplicações e domínios, incluindo banco de dados (Rizzi, 2004).

Considerando o objetivo do trabalho proposto e suas etapas, a abordagem utiliza-se do diagrama de classes e do diagrama de estrutura composta para representação da etapa da modelagem multidimensional utilizando a UML.

3.4.1 Diagrama de Classes

Um diagrama de classes é um diagrama estrutural ou estático com o qual se modela a estrutura de um sistema de classes e que sob vários aspectos assemelha-se a um diagrama ER. As diferenças surgem principalmente na modelagem de operações e nos relacionamentos (Muller, 2002).

As notações OO sobre as quais a UML foi fundamentada foram desenvolvidas a partir de notações ER. O diagrama de classes da UML oferece efetivamente uma notação ER, considerando sua complementação de restrições definidas pelo usuário (Halpin, 2002).

O diagrama de classes pode representar um conjunto de classes, *interfaces* e colaborações e seus relacionamentos, portanto, são os mais encontrados em sistemas modelados segundo a orientação a objetos, considerando sua representatividade da visão de projeto (Booch et al, 2000). É usado para modelar a visão estática de um projeto, modelando o vocabulário do sistema e o esquema lógico de banco de dados (Hiremath e Skibniewski, 2004).

O nome da classe é único dentro da classe, mas podem existir classes com o mesmo nome em pacotes diferentes, considerando pacotes a representação de um espaço de nomes, um escopo para os nomes dos elementos do pacote (Muller, 2002).

A representação de uma classe em um diagrama de classes, segundo o item 3.3.1, anteriormente apresentado, consiste de três divisões básicas. Os diagramas de classe da UML 2.0 mostram as classes do sistema, os relacionamentos (inclusive herança, agregação e associação), operações e atributos das classes (Ambler, 2005).

De uma perspectiva puramente orientada a dados, diagramas de classes persistentes são equivalentes a diagramas ER, com métodos e atributos multivalorados. Mapeando diagramas de classe persistentes para estruturas ER, faz-se necessário considerar outros tipos de relacionamentos, como a composição, a agregação e a dependência, que diferem entre as duas abordagens. Portanto, diante de tais considerações, diagramas de classe raramente são detalhados na fase de análise. Assim, a especificação detalhada do diagrama de classes é gerada durante a fase de projeto (Becker, 2001).

Na primeira divisão do diagrama, além do nome da classe, indica-se através de um estereótipo (<<persistente>>) se o estado de uma instância permanece quando o sistema destrói a instância, indicando que esta classe é persistente e que é convertida para a implementação.

Atributos e associações são considerados propriedades na UML 2.0, e são tratados da mesma maneira. As associações são implementadas como uma combinação de atributos e operações que em modelos simples pode-se assumir que os atributos e operações existem para implementar as associações (Ambler, 2005).

3.4.2 Diagrama de Estrutura Composta

O Diagrama de Estrutura Composta (*Composite Structure Diagram*) surge na proposta 2.0 da UML (OMG, 2004) com o objetivo de permitir a exibição de um diagrama de classes dentro de uma classe, que pode ser utilizado para a representação de um relacionamento de composição (Melo, 2004).

Na UML 2.0, componentes e conectores foram incorporados na linguagem no pacote da estrutura composta, com a finalidade de representar hierarquias de composição, aparecendo pela primeira vez o conceito de conector, dando condições de se gerar uma descrição arquitetônica (Rodríguez et. al, 2004).

O diagrama pode representar os relacionamentos entre as classes “todo” e suas “partes” e entre as próprias partes, como mostra a figura 9.

As classes “partes” podem ser ligadas por conectores, que são linhas indicando *links* que habilitam a comunicação entre duas ou mais instâncias. No caso da composição, indicam que uma classe “parte” pode se comunicar com outra. Para estes *links* a UML 2.0 define dois tipos de conectores (Melo, 2004):

- *Assembly*: permite que uma classe “parte” supra serviços que outra classe “parte” necessita, conectando duas partes como uma associação;
- *Delegation*: conecta o “todo” com uma de suas “partes”, sendo exibido com uma linha saindo da extremidade da classe composta para uma de suas “partes” dentro da classe composta.

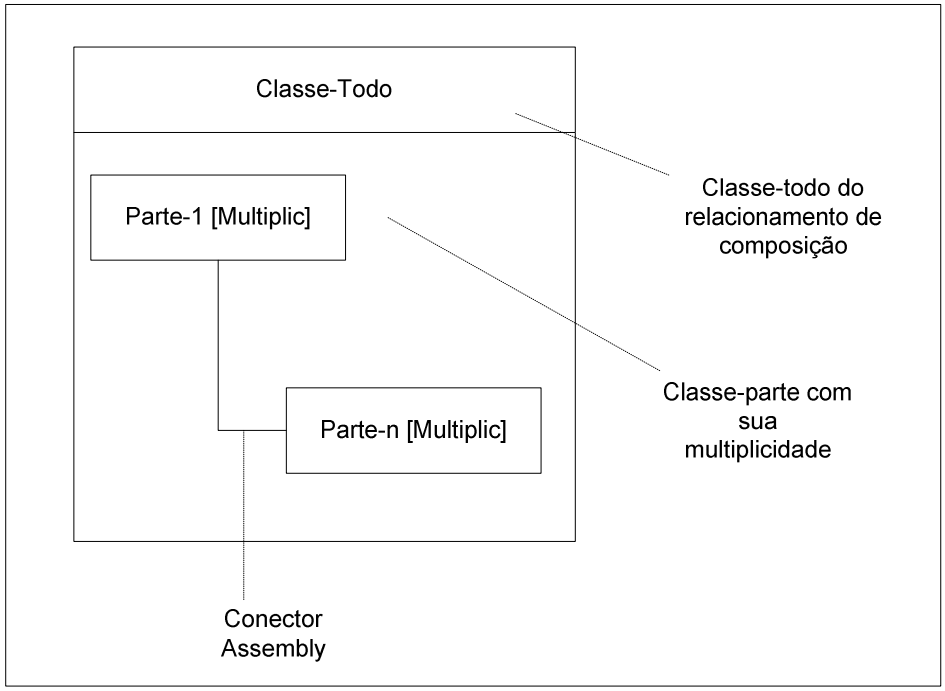

Figura 9 - Diagrama de estrutura composta.
Fonte: Melo (2004, p.125).

Na UML 2.0, o conector foi concebido como uma conexão primitiva, que é definido com um tipo de característica, no mesmo nível de complexidade que uma operação ou uma propriedade. O conector não apresenta um nome, definido apenas como um conjunto de extremidades (metaclasse *ConnectorEnd*) que unem os elementos conectáveis (metaclasse *ConnectableElement*). O conector pode estar relacionado ou não a uma associação, fornecendo um significado implícito (Rodríguez et. al, 2004).

Um relacionamento composto é um relacionamento entre estruturas ou entre entidades e estruturas, ou seja, é uma agregação de um ou mais relacionamentos específicos entre entidades que não pertencem a mesma estrutura (Danoch et al., 2005). Uma estrutura de alto nível inclui também estruturas de baixo nível e

relacionamentos compostos, em que um relacionamento composto é uma agregação de relacionamentos entre estruturas e entidades. Ou seja, cada estrutura também mostra relacionamentos externos, isto é, relacionamento de entidades na estrutura com entidades pertencentes a outras estruturas (Danoch et al., 2005).

3.5 A OO no modelo multidimensional

A orientação a objeto com UML pode fornecer uma anotação adequada para modelar todos os aspectos de um sistema de DW, desde os requisitos de usuário até a implementação (Trujillo et al., 2001). Esta linguagem possui uma estrutura de sistema de informação e propriedades dinâmicas no nível conceitual, de forma mais natural que outras abordagens clássicas, como o modelo Entidade-Relacionamento.

As ferramentas OLAP implementam o modelo multidimensional em dois níveis diferentes (Trujillo at al., 2001):

- estrutural - estruturas que formam o esquema de banco de dados para armazenar dados multidimensionais e o modelo multidimensional subjacente conhecido como metadados - que provê a chave do modelo semântico, como fatos, medidas, e dimensões;
- dinâmico - refere-se a definição dos requisitos do usuário final e operações OLAP para mais adiante analisar dados.

Considerando a abordagem da pesquisa, serão contempladas as características do nível estrutural dos trabalhos de Trujillo et al (Trujillo et al., 2001; Trujillo et al., 2003). O pacote da linguagem UML provê mecanismos, criando níveis diferentes de abstração e simplificando o modelo final. As propriedades dinâmicas podem ser representadas através da utilização dos diagramas de estado e de interação da UML para modelar o comportamento dos cubos de dados baseados nas operações das aplicações OLAP (Trujillo et al., 2003). A abordagem também permite definir atributos identificadores que podem ser definidos na classe Fatos, colocando a restrição {OID} próxima a uma medida nomeada (figura 10).

3.5.1 Fatos e dimensões

As dimensões e fatos são representados pelas classes Dimensão e Fatos, respectivamente, sendo que a classe Fatos representa os fatos e suas medidas,

definidas como atributos nestas classes (Luján-Mora et al., 2002). Um nível representa um grupo de instâncias da mesma granularidade em uma dimensão de análise (Abelló et al, 2005).

Classes Fatos são especificadas como classes compostas em relacionamentos de agregação compartilhados de várias classes Dimensão, conforme ilustra a figura 10. Estas classes consistem de dois tipos de atributos: atributos fatos, que representam medidas (as transações ou valores que estão sendo analisados), e dimensões degeneradas, que permitem ao projetista representar outras características dos fatos, além das medidas para análise (Luján-Mora et al., 2004a).

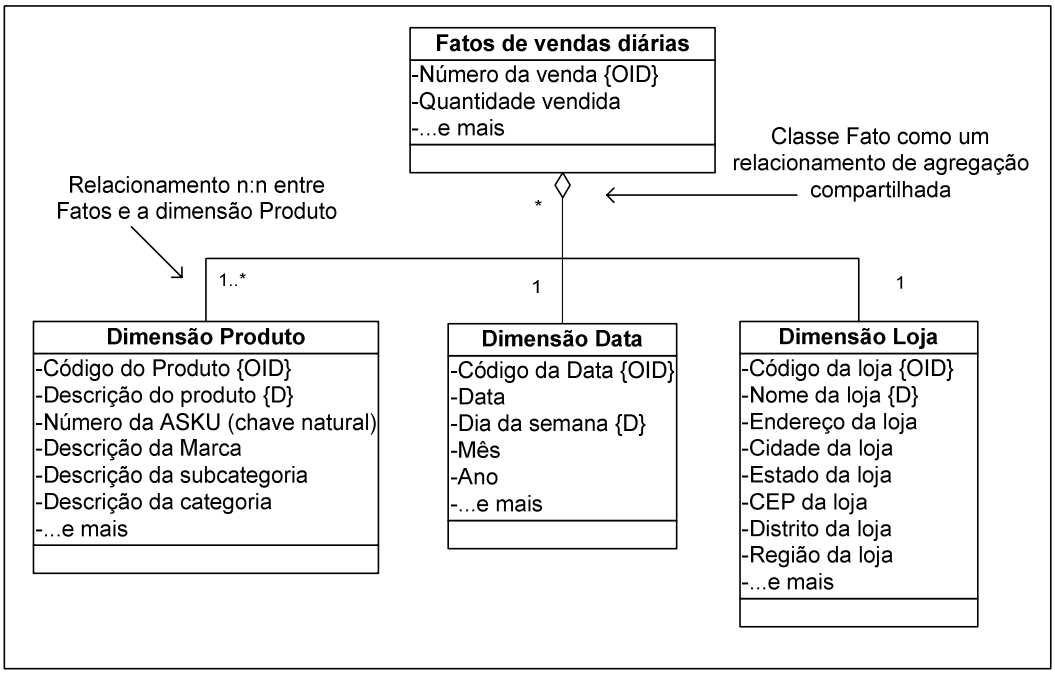

Figura 10 - Fatos e Dimensões.

Fonte: Adaptado de Kimball (2002) e Trujillo et al. (2001).

As regras da cardinalidade mínima da classe dimensão são definidas como 1 para indicar que uma instância de objeto fatos sempre é relacionada a instâncias de objeto de todas as dimensões. As regras de cardinalidade da classe Fatos são definidas como * para indicar que um objeto de dimensão pode ser parte de um, zero, ou mais instâncias de objetos de Fatos (Trujillo et al., 2001).

Graças à flexibilidade de relacionamentos de agregação compartilhada que a UML provê, pode-se considerar relacionamentos “muitos-para-muitos” entre fatos e

dimensões particulares. Isso é representado pela indicação da regra de cardinalidade 1.. * na classe de dimensão, para mostrar que uma instância de objeto Fatos pode ser relacionada a uma ou mais instâncias de objetos Dimensão. Na figura 10 esta flexibilidade é representada através do relacionamento entre a classe Fatos e a Dimensão Produto.

3.5.2 Medidas de derivação e aditividade

O conceito de aplicar aditividade ou sumarização para medidas de dimensões é importante para a modelagem de dados multidimensional. Uma medida é aditiva em uma dimensão se for possível usar o operador *SUM* para agregar valores de atributo em todas as hierarquias definidas naquela dimensão. Uma medida é semi-aditiva se o operador *SUM* puder ser aplicado a algumas dimensões, mas não a todas as dimensões. Uma medida é não-aditiva se o operador *SUM* não puder ser aplicado a nenhuma dimensão (Trujillo, 2001; Trujillo et al., 2003).

Por padrão, consideram-se todas as medidas como aditivas: o operador *SUM* pode ser aplicado para agregar valores de medidas em todas as dimensões, que possuem fórmulas subjacentes formais e contém os operadores permitidos, conforme representa a figura 11.

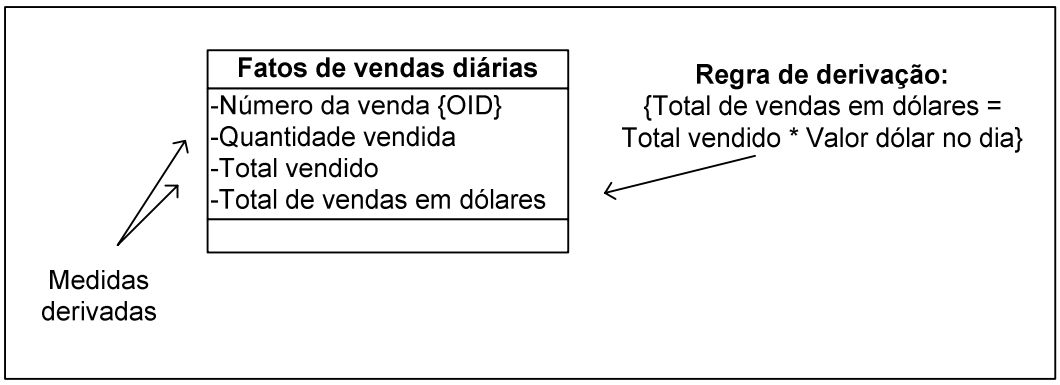

Figura 11 - Medidas derivadas e regras de derivação.

Fonte: Adaptado de Trujillo et al. (2001).

Para simplificar, o diagrama de classes pode conter as regras de aditividade e regras de derivação para os atributos derivados. A inclusão da definição de atributos derivados na fase de projeto conceitual evita sua definição incorreta nas fases

seguintes. Além disso, as regras de derivação podem ser usadas em uma fase posterior de implementação (Luján-Mora et al, 2004a).

3.5.3 Hierarquias de classificação

Para dimensões, uma classe básica representa todos os níveis de hierarquia de classificação. Uma associação de classes especifica os relacionamentos entre dois níveis de uma hierarquia de classificação. A única restrição é que as classes usadas para definir uma hierarquia de classificação ao longo de uma dimensão devem definir um DAG (*Directed Acyclic Graphs*) na classe dimensão. A estrutura DAG pode representar um caminho alternativo ou múltiplas hierarquias de classificação. Assim, uma classe B de uma hierarquia é considerada uma associação de uma classe A. Colocar a restrição {DAG} próxima de cada classe dimensão no diagrama de classes UML especifica que qualquer classe em uma hierarquia de classificação tem que definir um DAG.

As classes básicas, inclusive a classe dimensão, que pertencem à hierarquia de classificação devem conter um atributo identificador explicitamente definido, através da restrição {OID} próximo a um atributo em cada classe, como representado na figura 10.

Pode-se também, definir um descritor em cada classe que representa um nível de hierarquia de classificação, para que possa ser usado como atributo *default* para operações. Representa-se a restrição {D} próxima a um atributo, qualificando o atributo descritor para cada nível de hierarquia no diagrama de classes UML. Finalmente, pode-se definir os atributos descritores e identificadores simultaneamente (figura 10).

É importante definir as hierarquias de classificação de certos atributos de dimensão porque estas hierarquias provêem a base para a subseqüente análise de dados, considerando que um atributo de dimensão também pode ser agregado a mais de um atributo ou pertencer a múltiplas hierarquias. Por isto, DAGs são comumente utilizados na representação e análise de dimensões com suas hierarquias de classificação, como mostra a figura 12, com as diferentes hierarquias de classificação definidas pelas dimensões produto, loja e tempo. Na dimensão Produto foi definida uma hierarquia múltipla de classificação de forma a se agregar valores de dados ao longo de dois caminhos de hierarquia diferentes: Produto → tipo

→ família → grupo; e Produto → marca. A figura 13 mostra a hierarquia de classificação representada através do diagrama de classes para a dimensão Loja.

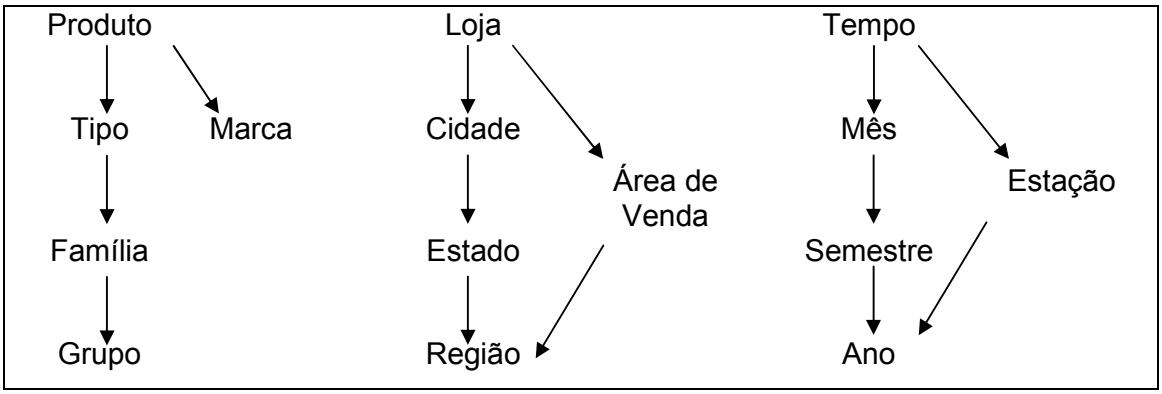

Figura 12 - Hierarquias de Classificação.

Fonte: Adaptado de Trujillo et al. (2003)

3.5.4 Exatidão e completeza

Os conceitos de exatidão e completeza são significantes para o propósito conceitual e para a modelagem multidimensional, definindo alguns passos. Considera-se a exatidão quando um objeto no mais baixo nível de uma hierarquia pertence a apenas um objeto de nível mais alto (Tryfona et al., 1999; Trujillo, 2001; Trujillo et al., 2003).

No exemplo da figura 12, uma região só se relaciona a um estado. Por completeza considera-se que todos os membros pertencem a um objeto de classe mais alta, e aquele objeto consiste apenas destes membros. Nas figuras 12 e 13, uma Região é formada por todos os Estados armazenados e todos os Estados que formam a Região são armazenados.

A multiplicidade 1 e 1..*, definida na parte da classe associada, envia os conceitos de exatidão e não exatidão, respectivamente. A exatidão significa que um objeto em um mais baixo nível de hierarquia pertence a apenas um objeto de mais alto nível, enquanto a não exatidão compreende as relações em que um objeto pode pertencer a mais de um objeto. Além disso, definir a restrição de completeza em uma parte da classe associada, envia a completeza para uma hierarquia de classificação. A completeza significa que todos os membros pertencem a um objeto da classe mais alta e que o objeto consiste apenas daqueles membros, conforme

mostra a figura 13. Por padrão, as hierarquias de classificação não possuem a característica de completeza (Luján-Mora et al., 2002).

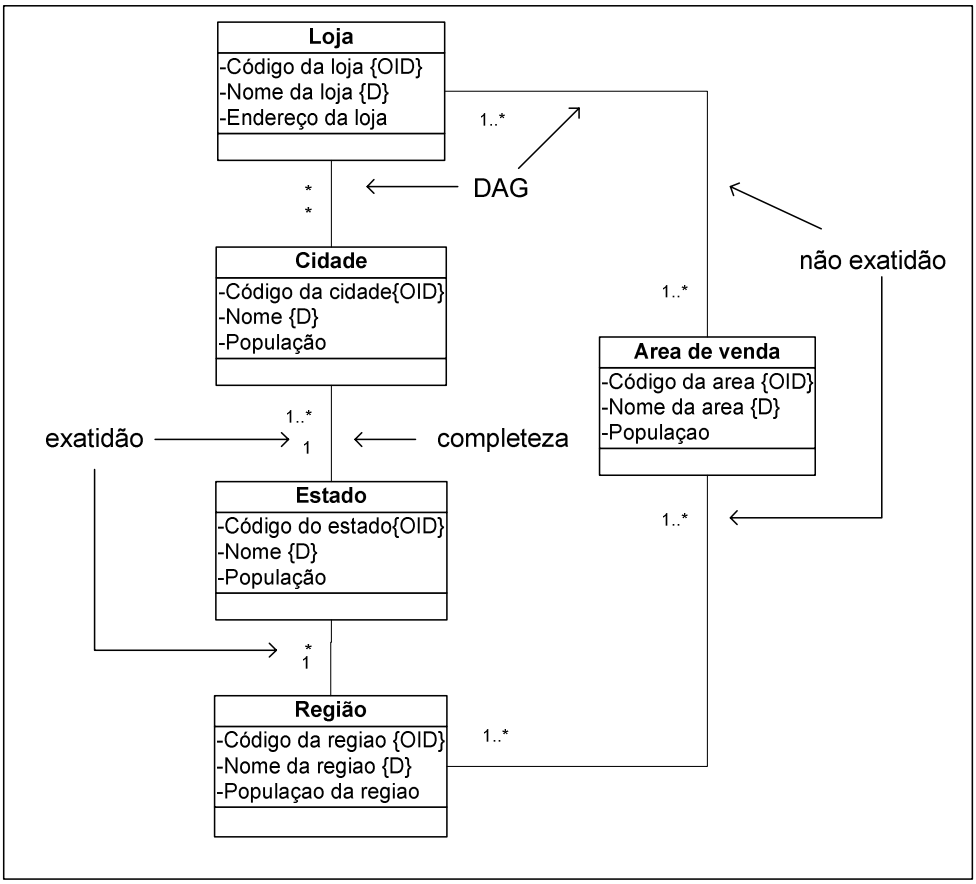

Figura 13 - Conceitos de exatidão e completeza.

Fonte: Adaptado de Trujillo et al. (2001)

3.6 Considerações finais

O presente capítulo abordou os principais conceitos do modelo multidimensional orientado a objetos, utilizando a UML como referência para a representação deste paradigma.

Dentre os conceitos básicos e componentes abordados são relacionados: classes, objetos, atributos e operações. Para conceituar os relacionamentos no paradigma OO, foram definidos a associação, classificação e a generalização.

Através do diagrama de classes, que é o diagrama estático mais utilizado na representação da modelagem de dados, foram representados conceitos da

modelagem multidimensional aplicados ao paradigma orientado a objetos, como as classes Fatos e Dimensão, além de seus respectivos relacionamentos.

Atentando para a utilização da proposta do modelo multidimensional representado pela UML, o capítulo abordou o nível estrutural da linguagem UML na representação deste modelo, explicando como os conceitos de hierarquias de classificação e de exatidão e completeza da modelagem multidimensional são utilizados através deste padrão. Também foram abordados os fundamentos da geração de DAGs e medidas de derivação.

Todos os conceitos e componentes apresentados neste capítulo foram incorporados à proposta metodológica de representação do modelo multidimensional sob o paradigma da orientação a objetos.

Após a introdução dos conceitos e componentes da orientação a objetos, da UML e de sua aplicação na modelagem multidimensional, o próximo capítulo apresenta metodologias existentes do modelo multidimensional e uma comparação destas propostas.

4 METODOLOGIAS DE DESENVOLVIMENTO DO MODELO MULTIDIMENSIONAL

4.1 Introdução

O projeto lógico de um DW freqüentemente se inicia com um esquema conceitual gerado a partir de estruturas relacionais operacionais. Aplicar esta abordagem implica em seguir as etapas de mapear a estrutura do modelo conceitual para um modelo lógico, e considerar sua implementação, que não é observada no esquema conceitual. Após sua geração, faz-se necessária sua aplicação (Singh, 2001, p.100):

> Desenvolver um modelo de dados envolve analisar e definir as entidades identificadas durante o processo de definição. Após ser construído, o modelo deve ser validado e os dados mapeados a partir de aplicações legadas para o modelo.

Considerando o objetivo do trabalho proposto, faz-se necessário verificar as metodologias de desenvolvimento do modelo multidimensional existentes e suas respectivas etapas, analisando suas características e aplicabilidades no modelo orientado a objetos.

Cada seção apresenta uma proposta de metodologia de desenvolvimento do modelo multidimensional. Considerando que as primeiras aplicações comerciais baseiam-se no modelo proposto por Kimball (1997), esta é a primeira metodologia apresentada, lembrando que as demais propostas consideram seus conceitos e fundamentos, baseados no esquema Estrela. Alguns trabalhos abordam metodologias para o desenvolvimento do DW, seguindo etapas do ciclo de desenvolvimento do próprio DW (Kimball, 1997), ou definindo um método integrado com a metodologia tradicional de desenvolvimento de software (Cavero et al., 2002).

As seções 2, 3 e 4 abordam metodologias baseadas no modelo ER para gerar o modelo multidimensional, relacionando etapas de geração de Fatos e Dimensões a partir de tabelas operacionais.

A modelagem OO, segundo Booch et al. (2000) é uma técnica adotada para o desenvolvimento de software, que tem como elemento principal o objeto. Considerando esta modelagem e o objetivo deste trabalho, as seções 5 e 6 abordam metodologias mais recentes, que utilizam o paradigma da orientação a objetos para gerar o modelo multidimensional.

A última seção faz uma análise comparativa das propostas, relacionando as principais características de cada metodologia.

4.2 Metodologia segundo Kimball

A metodologia de Kimball (1997, 2002) relaciona os conceitos fundamentais da modelagem multidimensional e propõe etapas a serem observadas na geração de um modelo multidimensional. A preocupação no modelo de Kimball (1997) volta-se para a modelagem do negócio a partir do modelo ER existente. Sua proposta baseia-se na utilização da MD para a geração do modelo no DW.

A metodologia é baseada em dois pontos: *Datawarehouse Bus Architecture* que mostra como construir uma série de *data marts*[1] para criar um DW corporativo; e *Business Dimensional Lifecycle* que, a partir dos requisitos do negócio, gera os *data marts* baseados no esquema multidimensional Estrela.

Um diagrama de um modelo ER deve ser representado em vários diagramas multidimensionais, considerando que o modelo ER é mais complexo que o multidimensional. Portanto, para a geração do modelo multidimensional a partir do modelo ER, a metodologia relaciona algumas etapas a serem observadas (Kimball, 1997, 2002):

Etapa 1: selecionar o processo de negócio a ser modelado

Inicialmente, deve-se decidir quais processos de negócio serão modelados, a origem do modelo multidimensional que será a base da tabela de fatos.

Etapa 2: declarar a granularidade

O segundo passo da metodologia busca definir o nível de detalhes que estarão associados às medidas da tabela de fatos do modelo. Para que os detalhes possam ser verificados de forma precisa, os dados devem ser expressos na menor granularidade possível.

Etapa 3: escolher as dimensões

As dimensões devem representar todas as descrições possíveis que utilizam valores únicos no contexto de cada medida. Através do processo de desnormalização de tabelas que se conectam diretamente com o negócio a ser modelado. Estas tabelas tornam-se tabelas Dimensão. A metodologia recomenda

[1] *Data mart* – dados de um único processo de negócio, DW departamental.

que quando uma tabela Dimensão conecta-se a mais de uma tabela Fatos, deve-se representá-la nos vários esquemas, referenciando-as como tabelas *conformed* entre os modelos multidimensionais.

As tabelas de dimensão normalizadas geram o modelo Floco de Neve. Os atributos redundantes (normalmente descritores) são removidos da tabela de dimensão desnormalizada e alocados em tabelas de dimensão secundárias normalizadas.

Etapa 4: identificar os fatos

Finalmente, são identificados os fatos numéricos, freqüentemente aditivos. A metodologia recomenda que fatos calculados devem ser armazenados fisicamente no banco de dados. Os fatos com diferentes granularidades devem ser registrados também em tabelas de fatos diferentes.

Portanto, pode-se observar que a proposta da metodologia de Kimball (1997, 2002) volta-se para o negócio a ser modelado e sua granularidade, adotando como padrão a geração de Tabelas de Fatos e de Dimensão, com regras básicas e simples de desenvolvimento de modelos de dados. Deve-se lembrar, porém, que o Modelo Estrela é amplamente utilizado para a modelagem do modelo multidimensional, adotado em propostas de outros autores.

As hierarquias de classificação são sutilmente tratadas por Kimball (2002), observando que são consideradas exclusivamente na representação do esquema Floco de Neve, que devem ser utilizadas apenas em situações que incluam a utilização de *outriggers*[2] .

Um projeto multidimensional deve permitir apenas fatos de uma granularidade padrão (a mesma dimensionalidade) para coexistir em uma única tabela de fatos. Esta uniformidade garante que todas as dimensões sejam usadas com todos os registros de fatos, reduzindo a possibilidade de erros de aplicação para combinar dados de granularidade diferente. Para propósitos práticos, a escolha mais comum para a aditividade é a soma. Aplicações são mais simples se armazenarem fatos em um formato aditivo (Kimball, 2002).

---

[2] Uma tabela de dimensão secundária ligada a uma tabela de dimensão. A tabela *outrigger* é uma interpretação do projeto físico de uma única tabela de dimensão lógica, que ocorre quando uma tabela de dimensão for normalizada.

Na modelagem multidimensional os projetos lógico e físico são muito semelhantes (Kimbal, 2002, pg.407). Estes modelos são diferenciados apenas nos detalhes específicos do banco de dados físico, como tipos de dados e índices.

A figura 14 representa o modelo proposto por Kimball, com as tabelas Fatos e Dimensão.

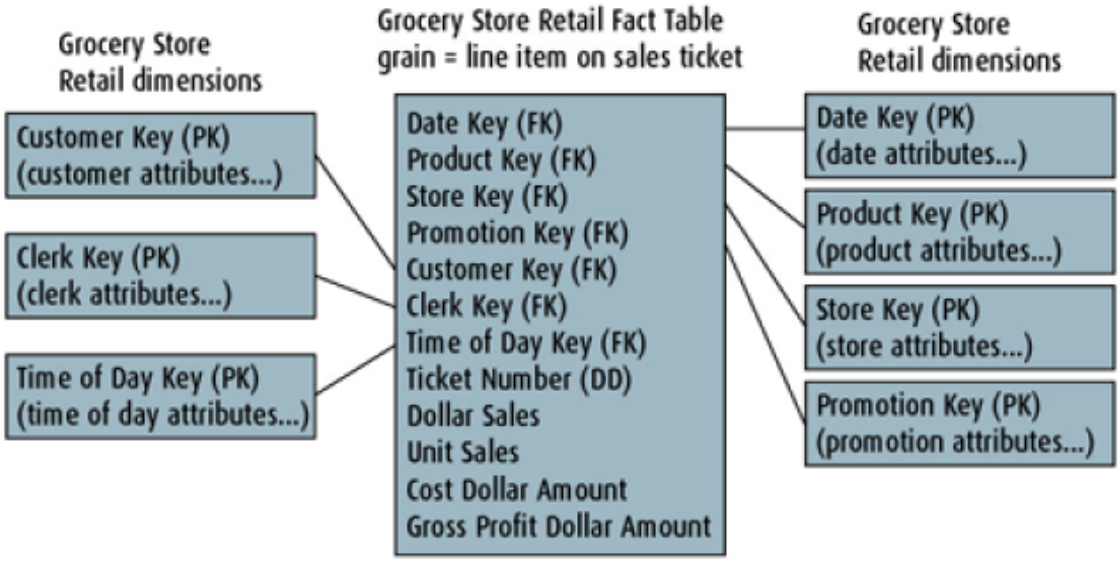

Figura 14 - Modelo dimensional da metodologia de Kimball.
Fonte: Kimabll (2003, p.23).

4.3 Metodologia segundo Moody e Kortink

A metodologia proposta por Moody e Kortink (2000) também aborda o desenvolvimento de modelos multidimensionais a partir dos tradicionais modelos ER. O primeiro passo do método envolve classificar entidades no modelo de dados dentre várias categorias. O segundo passo envolve identificar hierarquias que existam no modelo. O terceiro passo envolve separar estas hierarquias e agregar dados de transação para formar modelos multidimensionais. Finalmente, para concluir a proposta, são abordadas as questões de avaliação e refinamento da metodologia.

O autor apresenta várias alternativas de esquemas, tais como: *flat schema, terraced schema, star schema* e *snowflake schema*, a partir dos quais define um novo tipo de esquema, chamado *star cluster schema*, que é uma forma restrita do

esquema *snowflake*, que minimiza o número de tabelas enquanto evita a sobreposição entre diferentes hierarquias multidimensionais.

A seguir são descritas as etapas da metodologia proposta por Moody e Kortink (2000):

Etapa 1: classificar entidades

O primeiro passo para gerar um modelo multidimensional de um modelo ER é classificar as entidades em três categorias: de transação, de componentes e de classificação:

- Entidades de transação: descreve um evento que acontece em um determinado espaço de tempo e contém medidas que podem ser resumidas. Entidades de transação são as mais importantes entidades em um DW e formam a base para a construção das tabelas de fatos em uma esquema estrela.
- Entidades componentes: estão diretamente relacionadas a uma entidade de transação por um relacionamento 1:n. Estas entidades definem os detalhes ou componentes de cada transação do negócio, respondendo “quem”, “o que”, “quando”, “onde”, “como” e “por que” de um evento de negócio. Um componente importante de qualquer transação é o tempo. Entidades componentes formam a base para a construção de tabelas de dimensão em esquemas estrela.
- Entidades de classificação: são funcionalmente dependentes de uma entidade componente (diretamente ou transitivamente), representando hierarquias embutidas nos modelos de dados, que podem ser desmontadas em entidades componentes para formar tabelas de dimensão em um esquema estrela.

Em alguns casos, entidades podem ser ajustadas em categorias múltiplas. Define-se, então, uma hierarquia de precedência para solucionar tais ambigüidades:

- entidade de transação (precedência mais alta);
- entidade de classificação;
- entidade componente (mais baixa precedência).

Na prática algumas entidades não se ajustarão em nenhuma destas categorias. Tais entidades não se ajustam a estrutura hierárquica de um modelo multidimensional e não podem ser incluídas em esquemas Estrela.

Etapa 2: identificar hierarquias

Hierarquia é um conceito extremamente importante na modelagem multidimensional, e forma a base primária para modelos multidimensionais derivados de modelos ER. Como previamente discutido, a maioria das tabelas de dimensão em esquemas Estrela contém hierarquias embutidas. Uma hierarquia em um modelo ER é qualquer seqüência de entidades unidas por relacionamentos um-para-muitos, todos alinhados na mesma direção.

Etapa 3: gerar modelos multidimensionais

A metodologia proposta por Moody e Kortink (2000) utiliza dois operadores para produzir modelos multidimensionais a partir de modelos ER.

Operador 1: desmontar hierarquias

Entidades de nível mais alto podem ser “desmontadas” em entidades de nível mais baixo dentro de hierarquias. Isto gera redundância na forma de uma dependência transitiva que é uma violação da 3FN (Codd, 1970). Desmontar uma hierarquia é então uma forma de desnormalização.

Operador 2: agregação

O operador de agregação pode ser aplicado a uma entidade de transação para criar uma nova entidade contendo dados sumarizados. Um subconjunto de atributos é escolhido da entidade fonte para agregar (os atributos de agregação) e outro subconjunto de atributos escolhido para agregar por (o agrupamento de atributos). Atributos de agregação devem ser valores numéricos. A agregação de dados reduz a granularidade dos dados no DW, que limita os tipos de análises.

Etapa 4: avaliação e refinamento

Na prática, a modelagem multidimensional é um processo interativo. A escolha do esquema produz um primeiro projeto, mas através de um refinamento, produz-se o modelo de dados final. Muitas destas modificações são elaboradas para simplificar o modelo e lidar com padrões não hierárquicos nos dados, através da combinação de tabelas:

- Combinar Tabelas Fatos: tabelas de fatos com as mesmas chaves primárias, ou seja, com as mesmas dimensões, devem ser combinadas. Isto reduz o número de esquemas estrela e facilita a comparação entre fatos relacionados.

- Combinar Tabelas Dimensão: criar tabelas de dimensão para cada entidade componente freqüentemente resulta em um número grande de tabelas de dimensão. Para simplificar a estrutura do DW, dimensões relacionadas devem ser consolidadas em uma única tabela de dimensão.

Portanto, a proposta da metodologia de Moody e Kortink (2000) é uma variação da proposta de Kimball (1997), estabelecendo parâmetros para modelos multidimensionais variantes do modelo estrela. Porém, as etapas da metodologia são similares à proposta anterior. A figura 15 mostra um exemplo da proposta após concluídas todas as suas etapas.

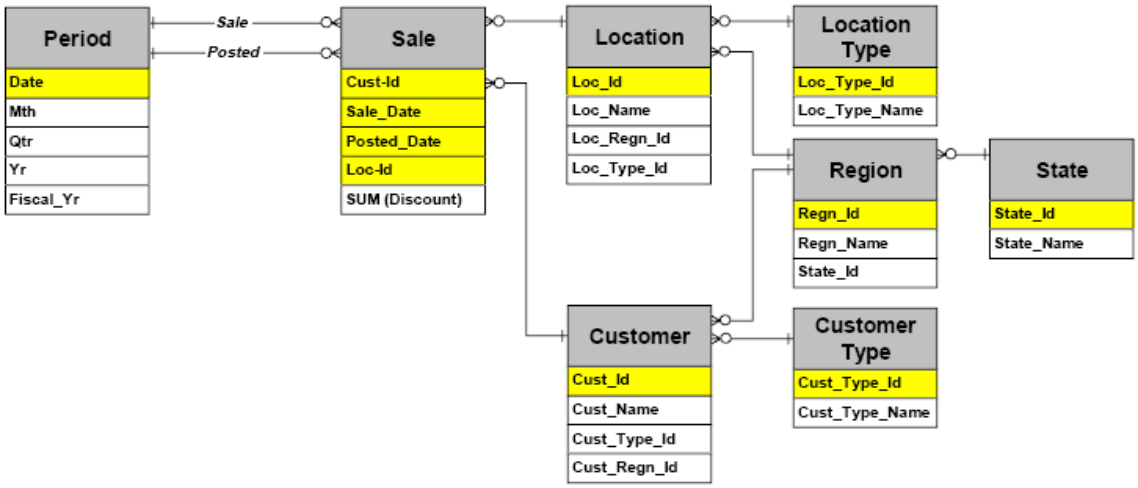

Figura 15 - Exemplo da proposta de Moody e Kortink.
Fonte: Moody e Kortink (2000, p. 4).

4.4 Metodologia segundo Cavero

O trabalho desenvolvido por Cavero et al. (2002, 2003) propõe uma metodologia para o projeto de DW através da modelagem conceitual, propondo a utilização do paradigma multidimensional durante a fase de projeto e manutenção, integrado com a metodologia de desenvolvimento de software. A proposta baseia-se no modelo conceitual ER e nos requisitos do usuário.

O modelo aborda sete etapas no desenvolvimento do processo, descritas a seguir:

Etapa 1: obter a estrutura preliminar de subcélulas

Inicialmente a metodologia visa obter um esquema preliminar, que corresponde à definição do negócio a ser modelado. Nesta etapa podem ser analisadas diferentes fontes de dados e os requisitos dos usuários. Se existe um esquema conceitual ER, podem ser usadas entidades ou atributos dos relacionamentos n:n para gerar esta estrutura preliminar de subcélulas.

Cada estrutura de células é composta de subestruturas, chamadas estruturas de subcélulas e seus métodos. Cada estrutura de subcélula consiste de um atributo de síntese e um conjunto de funções de síntese que representam como os dados operacionais serão processados para obter os dados sumarizados.

Etapa 2: obter as dimensões preliminares

O próximo passo é detectar as dimensões que irão fazer parte do esquema, ou seja, os valores que podem ser agregados. Nesta etapa, os usuários devem considerar dimensões de modo geral, sem detalhar os atributos multidimensionais hierárquicos. Uma dimensão é constituída de atributos multidimensionais.

Etapa 3: obter as hierarquias preliminares

Neste ponto já existe um esquema preliminar, com um conjunto de estruturas de subcélulas definidas em algumas dimensões. Portanto, agora é necessário identificar de maneira mais precisa as dimensões e suas hierarquias. Deve-ser identificar cada dimensão, descrevendo suas hierarquias de agregação. Porém, ainda não é necessário detalhar domínios das dimensões para cada agregação. Nesta etapa novas dimensões podem ser detectadas, como a dimensão Tempo.

As hierarquias são definidas a partir dos esquemas conceituais do banco de dados operacional, observando os atributos, entidades e seus relacionamentos.

Etapa 4: refinar as hierarquias

A próxima etapa compõe-se do refinamento das hierarquias obtidas na etapa anterior. Este refinamento consiste em um detalhamento dos atributos multidimensionais das hierarquias. Novos atributos podem ser detectados, assim como convertidos em atributos de descrição. Devem ser definidos os domínios para cada atributo, assim como os domínios das agregações de hierarquias devem ser detalhados, com suas respectivas funções.

Os domínios de descrição são usados para representar os atributos de descrição que representam informação complementar sobre os atributos multidimensionais.

Uma agregação consiste de uma função agregação e dois domínios de dimensão, sendo um deles o original e o outro o destino. Uma hierarquia é um grupo de domínios de agregação, representada por um gráfico onde cada nó representa um domínio de dimensão, e cada arco representa uma função de agregação.

Etapa 5: obter a estrutura detalhada de subcélulas

Neste instante já se observa que estão completamente definidas as dimensões e suas hierarquias, portanto, deve-se detalhar as estruturas das subcélulas. Para cada estrutura de subcélula especificam-se seus atributos e funções de sumarização, em que usualmente são utilizadas a função *SUM*.

Atributos sumarizados da tabela Fatos são definidos como domínios quantitativos nos quais podem ser aplicadas operações matemáticas e domínios booleanos, usados para indicar a existência ou não de informação.

Etapa 6: obter o esquema Fatos

Nesta etapa agrupam-se as estruturas de subcélulas em estruturas de células para formar esquemas Fatos. Cada estrutura célula pertence a um esquema Fatos e cada esquema Fatos possui dimensões associadas. Portanto, devem ser detectadas estruturas de subcélulas que puderem ser agrupadas em um único esquema Fatos.

A chave primária da tabela Fatos é composta das chaves estrangeiras das tabelas de dimensão. Todo atributo sem função de sumarização é transformado em colunas da tabela Fatos, caso contrário, cada função é transformada em uma coluna que irá armazenar o resultado do dado operacional sumarizado.

Etapa 7: verificar e validar o esquema multidimensional

Nesta etapa assegura-se que o esquema conceitual esteja completo, ajustado aos requisitos do usuário e critérios pré-definidos de qualidade. Outras verificações devem ser feitas, como a disponibilidade de dados nas fontes de dados.

A utilização de uma metodologia tradicional de desenvolvimento de projetos proposta por Cavero et al. (2003) também é proposta por Prakash e Gosain (2003), que representam as etapas e resultados esperados para cada uma delas no quadro 8, adaptando os resultados para o ambiente da modelagem multidimensional.

Etapa	Resultado
Levantamento de requisitos	Objetivos e hierarquias de decisão.
Projeto conceitual	Fatos, dimensões, agregações, metadados, dados históricos.
Projeto lógico	Representação em DDL do pacote DW a ser usado.
Projeto físico	*Layout* físico do DW.

Quadro 8 - Etapas da metodologia de desenvolvimento de DW.

Fonte: Adaptado de Prakash e Gosain (2003).

A metodologia utiliza o modelo de dados multidimensional no projeto conceitual. Para o projeto lógico e físico, podem ser usados o modelo de dados multidimensional ou o relacional. No projeto conceitual são representados os requisitos do usuário de maneira similar ao modelo ER. Sistemas OLTP e requisitos obtidos dos usuários são as principais entradas do esquema conceitual.

A metodologia é fundamentalmente focada na modelagem de dados e não considera os aspectos funcionais do desenvolvimento.

No trabalho apresentado no ano seguinte (Cavero et al., 2003), os autores colocam sua proposta baseada nas metodologias tradicionais de desenvolvimento de software, com as etapas de análise, projeto e implementação. Esta metodologia segue as etapas anteriormente estabelecidas para definir o projeto conceitual, utilizando os conceitos de Fatos, Dimensões e Hierarquias. No projeto lógico e físico, pode-se utilizar o modelo relacional ou multidimensional.

A figura 16 mostra um exemplo do esquema gráfico final proposto na metodologia, representando respectivamente as fases de análise, projeto e implementação: *Information System Analysis* (ASI), *Design* (DSI) e *Construction* (CSI).

Esta metodologia assemelha-se ao trabalho desenvolvido por Hüsemann et al. (2000), que propõe um modelo de processo do projeto de um DW similar ao processo tradicional de projeto de banco de dados, colocando o modelo em quatro fases seqüenciais:

- Fase de análise de requisitos e especificação: o modelo operacional ER envia informações básicas para determinar o potencial da análise multidimensional. Nesta fase, os responsáveis pelo negócio selecionam estrategicamente atributos relevantes do banco de dados operacional e especificam a finalidade para usá-los como dimensões e/ou medidas.
- Fase de projeto conceitual: transforma a especificação de requisitos de negócios semiformal em um esquema conceitual multidimensional formalizado. A formalização resulta em um esquema multidimensional gráfico, que abrange o esquema Fatos com suas Medidas e as hierarquias de Dimensão.

- Fase de projeto lógico: o esquema conceitual é convertido em um esquema lógico, gerado de acordo com as regras de transformação que se referem apenas aos diagramas conceituais desenvolvidos e restrições resumidas.
- Fase de projeto físico: implementação do esquema lógico com as respectivas propriedades individuais do objetivo do sistema de banco de dados, incluindo as técnicas de otimização física, bem como os ajustes específicos para as aplicações OLAP, como a desnormalização relacional das tabelas Dimensão.

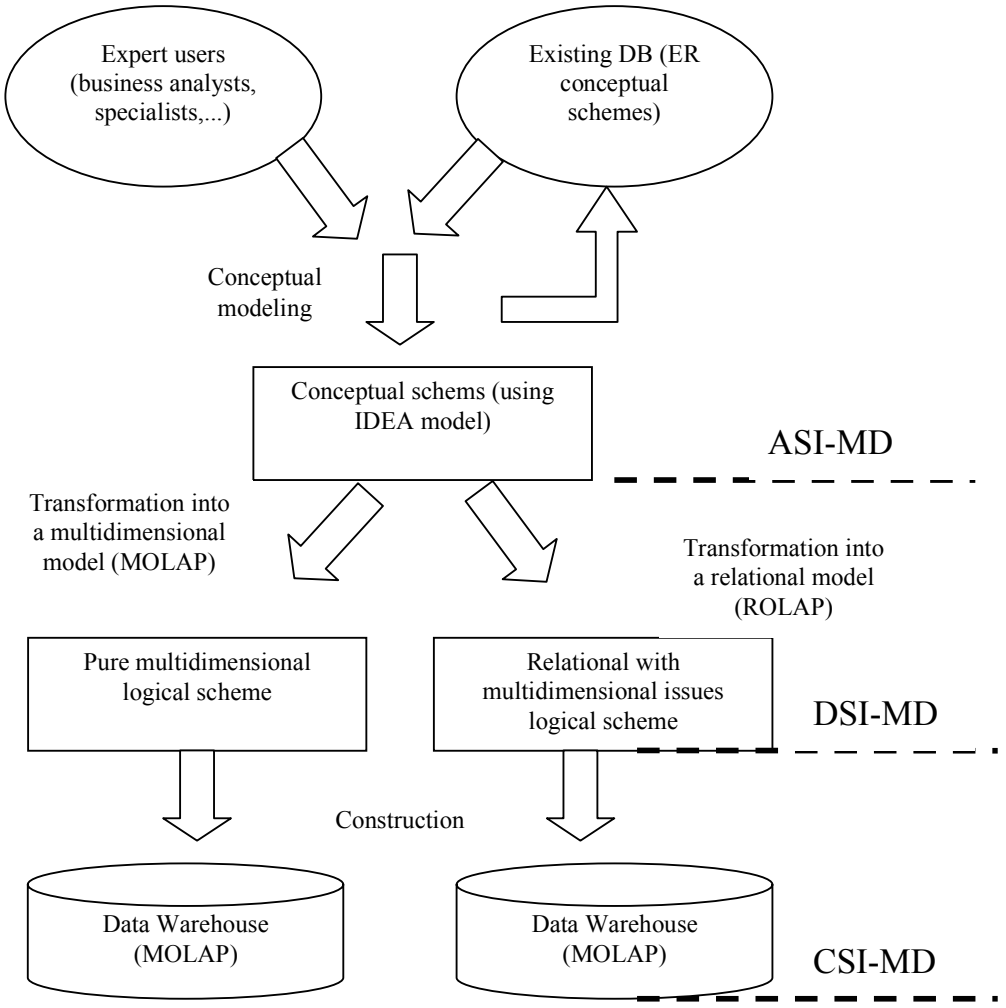

Figura 16 - Exemplo da metodologia proposta.

Fonte: Cavero et al. (2003, p. 196).

4.5 Metodologia segundo Trujillo

A metodologia proposta por Trujillo et al. (2000, 2003) nomeada de modelo GOLD, é um aprimoramento do modelo OOMD (*Object Oriented Multidimensional Database*), proposto anteriormente por Trujillo e Palomar (1998), que definia objetos abstratos sem referência a tabelas ou a sua subseqüente implementação. É o primeiro modelo que apresenta os conceitos de classe para o encapsulamento dos dados e suas operações, e que oferece um alto nível de abstração.

O modelo GOLD foi proposto com a inclusão dos conceitos da UML, como classes e associações, e propondo um modelo conceitual abordando as operações OLAP. O modelo inicial aborda os conceitos do modelo multidimensional clássico (dimensões e fatos) para propor uma nova abordagem baseada no paradigma OO. Os elementos básicos do modelo multidimensional OO são as classes Dimensão e Fatos. A vantagem de usar o paradigma OO refere-se a fornecer um modelo conceitual flexível, natural e mais simples que os modelos anteriores. Os autores também acreditam que esta proposta fornece um modelo conceitual genérico com total independência dos aspectos físicos.

Segundo a visão dos autores, o modelo Estrela considera apenas tabelas relacionais e, portanto, elementos básicos dos bancos de dados multidimensionais, como os atributos das hierarquias de classificação das dimensões não podem ser representados. A metodologia aborda os aspectos da multidimensionalidade e o paradigma da orientação a objetos, descritos a seguir.

Fatos:

No modelo GOLD, as classes Fatos (FC) são especificadas como classes compostas de uma relação agregação em que as classes Dimensão (DC) são os componentes. Um fato é um item de interesse da empresa e é descrito através de um grupo de atributos chamados medidas ou atributos fatos, que estão contidos em células ou pontos em um cubo de dados. Além disso, este grupo de medidas é baseado em um grupo de dimensões, definidos como a granularidade adotada para representar os fatos.

No modelo, um fato denota um relacionamento muitos-para-muitos entre várias classes dimensão. O modelo GOLD define uma classe Fatos como uma classe

composta em uma relação agregação na qual várias classes dimensão são os componentes.

Dimensões:

As classes Dimensão contêm objetos Dimensão que fornecem características dos dados reais. Uma classe Dimensão é um conjunto de eventos permitidos nos objetos da classe. As classes Fatos são construídas das classes Dimensão.

Atributos:

Atributos são n-tuplas em que cada elemento é uma característica que tem o objeto da classe específica, ou seja, esta tupla caracteriza o objeto da classe.

Agregação:

O conceito de agregação de classes é tomado verificando as linguagens de especificação formais OO, fornecendo um operador específico para construir classes complexas de classes básicas como uma relação de agregação.

Aditividade:

A metodologia proposta considera medidas derivadas definindo predicados complexos através de operações matemáticas e funções relacionais de agrupamento.

Um atributo dimensão pode também ser agregado a mais de uma hierarquia. Múltiplas hierarquias de classificação e caminhos alternativos de hierarquias podem ser representados através de DAGs.

Hierarquia de classificação:

Outra característica deste modelo é a hierarquia de classificação definida em atributos nas dimensões, que determinam como instâncias fatos podem ser agregadas e selecionadas significativamente para o processo de decisão.

Cada nível da hierarquia de classificação é especificado por uma classe (chamada classe básica). Uma associação de classes especifica os relacionamentos entre dois níveis de uma hierarquia de classificação. O único pré-requisito é que estas classes devem definir um DAG na classe dimensão.

A metodologia adota o diagrama de classes para a representação do modelo de dados do DW e o uso de pacotes para a representação do mecanismo de agrupamento de classes visando representar diferentes níveis de abstração. Os autores observam que o diagrama de classes melhora e simplifica a especificação do sistema, gerada pela semântica clássica do modelo de dados ER.

O modelo propõe uma abordagem orientada a objetos para a modelagem multidimensional de bancos de dados relacionais, definindo os elementos básicos: classes Dimensão e Fatos, além de outros conceitos do paradigma orientado a objetos aplicados na modelagem multidimensional. A metodologia, porém, não apresenta um conjunto de regras de transformação ou geração do modelo multidimensional a partir de um modelo operacional. Apenas aplica os conceitos da orientação a objetos no modelo multidimensional.

A figura 17 mostra os diagramas da metodologia. Na figura 17a é mostrado um exemplo de regras de derivaçao. A figura 17b representa as hierarquias de classificação e na figura 17c a representacao da categorização das dimensões.

A continuidade dos trabalhos dos autores voltou-se para a aplicação dos pacotes da UML na representação de modelos mais complexos e com um grande número de classes e relacionamentos e, posteriormente, na representação dos aspectos dinâmicos, através da utilização dos diagramas de estado e de interação (Trujillo et al., 2003). Atualmente, o modelo é complementado através da geração de documentos em XML para a representação do modelo multidimensional (Trujillo et al., 2004).

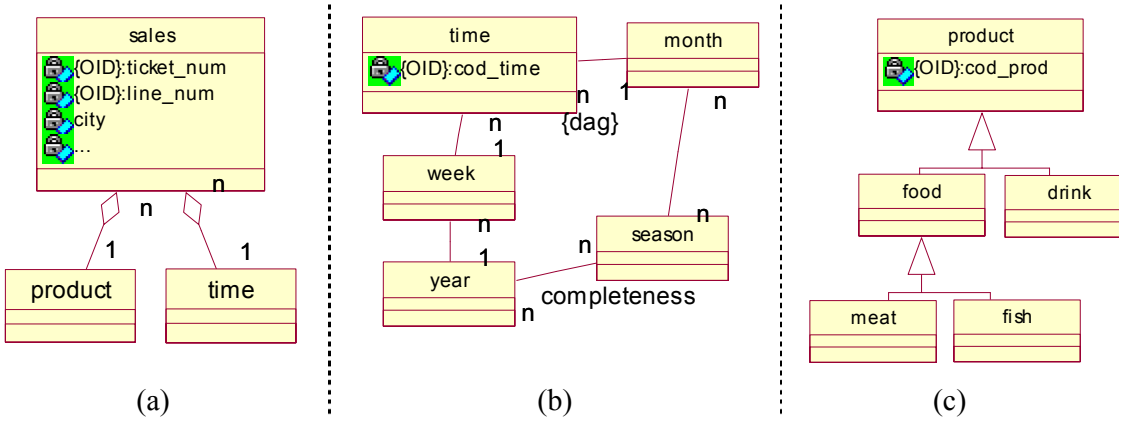

Figura 17 - Modelo da metodologia de Trujillo.

Fonte: Trujillo et al. (2003, p.22).

4.6 Metodologia segundo Abelló

Abelló et al. (2000b) relacionam as principais vantagens da utilização do modelo multidimensional orientado a objetos, no nível conceitual. Os trabalhos desenvolvidos pelos autores (Abelló et al. 2000b, 2002, 2005) apresentam os

benefícios do modelo de dados OO para integrar as diferentes visões multidimensionais e manter a semântica dos dados no nível conceitual.

A metodologia proposta pelos autores, denominada modelo YAM^2 (Abelló et al., 2002, 2005) baseia-se na dualidade Fatos-Dimensões, considerando que os Fatos representam os dados a serem analisados e as Dimensões mostram as diferentes visões destes dados. Tais características são definidas e aplicadas em níveis compostos de nós e arcos.

Os nós são compostos de seis tipos, descritos a seguir:

- Fatos: representam o assunto de análise. Medidas agrupadas em células correspondem a Fatos.
- Dimensão: usada para caracterizar os Fatos e representam um ponto de vista dos dados a serem analisados.
- Célula: representa um conjunto de instâncias de um determinado tipo de fato mensurado na mesma granularidade para suas dimensões de análise.
- Nível: representa um conjunto de instâncias da mesma granularidade em uma análise multidimensional.
- Descritor: um atributo de um nível, usado para selecionar suas instâncias.
- Medida: um atributo de uma célula representando dados mensurados a serem analisados. Assim, cada instância de células contém um (possivelmente vazio) grupo de medidas.

Uma vez definidos os nós, os arcos encontrados entre eles, ou seja, os relacionamentos de um elemento com outro, são a classificação/ instanciação, generalização/ especialização e agregação/ composição.

- Na classificação ou instanciação cada instância é relacionada a pelo menos uma classe no esquema. Todas as instâncias compartilham atributos e representam conceitos relacionados que são agrupados em uma determinada classe.
- A especialização/generalização relaciona duas classes, em que uma delas tem um significado mais específico que a outra. A especialização divide as instâncias de uma superclasse em diferentes subclasses e a generalização considera o processo inverso. Neste relacionamento surge o conceito de herança.

- Pelo relacionamento de agregação/composição é possível construir novos objetos. Na composição um novo objeto é concebido como composto por outros, que são suas partes, chamado de “todo-parte”. Na agregação os objetos agregados são apenas características de um novo.

A metodologia divide o desenvolvimento do modelo multidimensional em três níveis, em que são agrupados os nós:

- Nível mais alto (*Upper Level* – UL): neste nível são encontrados os fatos e dimensões, compondo um esquema Estrela.
- Nível intermediário (*Intermediate Level* – IL): dimensões e fatos são decompostos em Níveis de Dimensão e Células Fatos, respectivamente. Os diferentes níveis da dimensão formam uma hierarquia. Cada célula fatos contém dados que são grupos de medidas.
- Nível mais baixo (*Lower Level* – LL): este nível mostra os atributos dos níveis de dimensão e das células fatos, que são os descritores e as medidas, respectivamente. Também são definidos os tipos de medida para mostrar que várias medidas em diferentes células correspondem ao mesmo conceito de medida em diferentes níveis de agregação.

O modelo mostra como os dados podem ser classificados e agrupados de modo apropriado para uma subseqüente sumarização. Os dados sumarizados podem ser refletidos no esquema, assim como as operações de agregação para obtê-los. A figura 18 mostra a representação do modelo no nível mais baixo (*Lower Level* – LL).

Na modelagem multidimensional é essencial saber como um determinado tipo de medida pode ser agregado para se obter a maior granularidade. O modelo identifica três condições necessárias para a sumarização:

- disjunção: subgrupos de objetos a serem agregados devem ser disjuntos;
- completeza: a união de subgrupos deve constituir o grupo inteiro;
- compatibilidade: atributo categoria (nível), atributo sumarizado (tipo de medida) e função estatística (sumarização) devem ser compatíveis.

Diferentes funções de agregação (*SUM, AVG, MIN*, etc) podem ser usadas em dimensões para se obter diferentes medidas em uma célula complexa. A aditividade

pode ser refletida no modelo como um caso particular de agregabilidade em que a função *SUM* é aplicada.

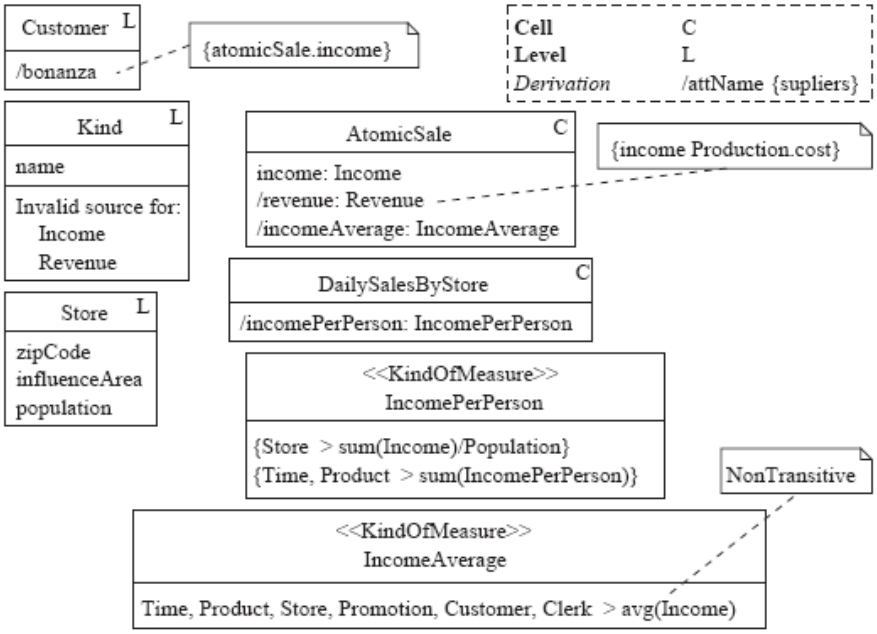

Figura 18 - Modelo da metodologia de Abelló, representando o *Lower Level* – LL.
Fonte: Abelló et al. (2005, p. 10).

Os autores utilizam um estudo de caso para validar a proposta, baseado no modelo de Kimball (2002), que é modelado em cada um dos três níveis propostos e seus respectivos elementos. A metodologia inicia a modelagem no mais alto nível, em que são representados os Fatos e Dimensões e respectivos relacionamentos, que compõem o esquema Estrela. Na seqüência, o nível intermediário define as hierarquias de agregação das dimensões, e os caminhos alternativos para estas hierarquias. Finalmente, o nível mais baixo define os atributos e suas medidas derivadas.

4.7 Análise comparativa dos modelos

As metodologias estudadas nas seções anteriores podem ser classificadas em dois grupos, diferenciados pelo paradigma relacional e orientado a objetos. Enquanto as metodologias de Kimball (1997, 2002), Moody e Kortink (2000) e Cavero et al. (2002) voltam-se para a modelagem multidimensional a partir do modelo ER e dos requisitos do usuário, as metodologias de Trujillo et al. (2000, 2003) e Abelló et al. (2002, 2005) buscam associar o modelo multidimensional com

os conceitos da orientação a objetos, utilizando a UML para representar seus diagramas. Todas as metodologias partem do esquema Estrela ou Floco de neve como padrão para a modelagem multidimensional.

Considerando as metodologias do primeiro grupo, observa-se que a proposta de Kimball (1997, 2002) restringe-se a etapas gerais de um mapeamento do modelo ER para o modelo multidimensional, não abrangendo especificidades, considerando o tratamento de hierarquias de classificação como exceção. A proposta de Kimball (1997, 2002) oferece indicativos do processo de mapeamento, que são efetivamente aplicados nos exemplos do livro "*The Data Warehouse lifecycle toolkit*" (Kimball, 1998).

Moody e Kortink (2000), complementando a metodologia proposta por Kimball (1997, 2002), detalham as etapas de um mapeamento do modelo ER para um modelo multidimensional, tratando separadamente entidades, hierarquias e relacionamentos.

A metodologia proposta por Cavero et al. (2003) segue as propostas anteriores, diferenciando-se por especificar separadamente as fases tradicionais de desenvolvimento de software: análise, projeto e implementação e utilizando o modelo relacional e multidimensional para exemplificar sua proposta.

Resumidamente, as metodologias consistem das seguintes etapas:

- inicialmente, deve-se identificar qual é o processo de negócio a ser modelado, determinando o que será armazenado no data warehouse; este processo é a fonte das medidas do modelo de dados;
- o segundo passo é identificar a granularidade, determinando o que significa uma coluna da tabela de fatos; esta granularidade consiste no aspecto básico a ser analisado;
- na seqüência, são identificadas as Dimensões, que devem ser descritivas buscando a confirmação da granularidade definida;
- finalmente, os fatos são identificados através da definição das medidas.

Observando os trabalhos do segundo grupo, que consideram o paradigma da orientação a objetos em suas metodologias, o modelo GOLD, proposto por Trujillo et al. (2000, 2003), apresenta o mais alto nível de abstração, representando as classes Dimensão, Fatos e posteriormente a classe Cubo para encapsular dados e operações, enquanto os demais modelos definem separadamente suas operações. A classe Fatos é definida como sendo composta de uma relação agregação em que

as classes Dimensão são os componentes. A metodologia propõe preceitos para a classificação da classe Fatos de acordo com o tipo de agregação. Também são consideradas as múltiplas hierarquias de classificação que são caracterizadas por atributos nos modelos OOMD e GOLD.

A metodologia de Abelló et al. (2002, 2005) relaciona as principais vantagens da utilização do modelo multidimensional orientado a objetos, no nível conceitual. O trabalho desenvolvido pelos autores apresenta os benefícios do modelo de dados OO para integrar as diferentes visões multidimensionais e manter a semântica dos dados no nível conceitual. A metodologia propõe-se a ser construída em níveis, com o detalhamento do modelo e seus componentes, utilizando a UML para sua representação.

Observa-se uma proximidade nos conceitos básicos da modelagem multidimensional aplicada ao paradigma orientado a objetos, com pequenas variações de nomenclatura adotada pelos autores. O quadro 9 mostra um resumo das diferenças entre as características de cada modelo, observando os conceitos da modelagem multidimensional.

4.8 Considerações finais

Nas metodologias estudadas os autores colocam genericamente as mesmas etapas para a geração do modelo multidimensional, definindo a identificação das entidades como Dimensão ou Fatos, a partir dos relacionamentos existentes no modelo ER. Ainda observando os relacionamentos, são identificados como relacionamentos ou hierarquias, em que atributos são agregados às dimensões relacionadas ou continuam relacionando-se a elas, como definido no modelo Floco de Neve.

Uma análise do modelo gerado e a necessidade de um refinamento das propostas faz com que a dimensão Tempo inclua-se na referência dos modelos multidimensionais, considerando um de seus objetivos, que é a análise de dados em determinado período de tempo.

O presente capítulo analisou metodologias existentes para a geração do modelo multidimensional. A partir destas análises, o próximo capítulo propõe uma metodologia que aborde a modelagem multidimensional sob o paradigma da

orientação a objetos, indicando as etapas de elaboração do modelo e sua representação.

Característica	KIMBALL	MOODY e KORTINK	CAVERO	TRUJILLO	ABELLÓ
Fatos	Grupos de atributos que geram tabelas relacionais, gerados dos relacionamentos n:n	Entidade de transação, composta dos atributos que representam eventos do negócio	Formados por agrupamentos de estruturas de subcélulas em estruturas de células	Grupo de atributos chamados de medidas ou fatos, que estão contidos em células	Medidas agrupadas em células
Dimensões	Grupos de atributos gerados da desnormalização dos relacionamentos 1:n	Entidades componentes, geradas da desnormalização dos relacionamentos 1:n	Constituídas de atributos multidimensionais	Atributos ou componentes de uma relação agregação	Representam um ponto de vista dos dados a serem analisados
Granularidade	Definida pela lista de dimensões, informando o escopo da medição	--	--	Definida pelas dimensões	Definida pelas dimensões de análise
Função de agregação	Fatos numéricos, sendo a soma a escolha mais comum	Aplica operador de agregação a uma entidade de transação para criar dados sumarizados	Domínios quantitativos em que podem ser aplicadas funções de sumarização, usualmente a soma	Predicados complexos definidos através de operações matemáticas e funções relacionais de agrupamento	Aplica função de agregação (*sum, avg, min*, etc) em dimensões
Hierarquia de classificação	Devem ser tratadas como tabelas outrigger, no esquema Snowflake	Entidades de classificação que representam hierarquias embutidas podendo ser desmontadas em entidades componentes	Grupo de domínios de agregação, representado por um gráfico em que cada nó representa um domínio de dimensão, e cada arco representa uma função de agregação	Atributos nas dimensões que determinam como instâncias fatos podem ser agregadas. Representadas através de DAGs	O nível intermediário define as hierarquias de agregação das dimensões, e os caminhos alternativos para estas hierarquias
Abordagem	Relacional	Relacional	Relacional e multidimensional	Orientada a objetos	Orientada a objetos

Quadro 9 - Resumo das características dos modelos analisados.

5 PROPOSTA DE METODOLOGIA DE IMPLANTAÇÃO DE MODELOS MULTIDIMENSIONAIS EM BANCO DE DADOS ORIENTADO A OBJETOS

5.1 Introdução

Considerando o objetivo do presente trabalho, que busca propor uma metodologia para a modelagem multidimensional representada graficamente na UML e seu mapeamento em banco de dados orientados a objetos, o capítulo cinco apresenta as etapas no desenvolvimento desta proposta.

A partir da análise e comparação das metodologias para a implementação do modelo multidimensional (Kimball, 1997, 2002; Moody e Kortink, 2000; Cavero et al., 2002; Trujillo et al., 2000, 2003; Abelló et al., 2002, 2005), a proposta aqui apresentada busca atender a todos os requisitos considerando a modelagem multidimensional.

Um modelo de dados é composto por três níveis de modelagem: a modelagem de alto nível, de nível intermediário e de baixo nível. Estes níveis podem ser adaptados do ambiente operacional (Oliveira, 1998), agregando características do modelo multidimensional conceitual (Abelló et al., 2001a):

- nível mais alto: seguindo um escopo de integração, que define os limites do modelo, onde se encontram as dimensões e os fatos; as dimensões são usadas para caracterizar os fatos e mostrar as óticas sob as quais os fatos serão analisados;
- nível intermediário: as dimensões e os fatos são decompostos em níveis multidimensionais e células de fato, respectivamente; os diferentes níveis multidimensionais em uma dimensão formam uma hierarquia;
- nível mais baixo: mostra o grupo de atributos de um nível multidimensional e as células de fato, representando suas características físicas; são os atributos de classificação e as medidas, respectivamente. As medidas são agrupadas em células, quando se referem ao mesmo fato.

A metodologia proposta envolve cinco etapas, descritas seqüencialmente nas seções 2, 3, 4, 5 e 6:

- definição do modelo de negócio;
- geração do modelo multidimensional;

- representação do modelo multidimensional através do diagrama de classes e de estrutura composta;
- mapeamento do modelo gerado para banco de dados orientado a objetos;
- implementação do modelo.

A seção 7 resume a metodologia proposta, representando seu esquema gráfico.

Na seção 8 faz-se uma análise comparativa da metodologia proposta e das anteriormente estudadas.

5.2 Etapa 1: definir o modelo de negócio

As metodologias estudadas anteriormente apontam como primeira tarefa a definição do negócio a ser modelado e, posteriormente, implementado. Observa-se, portanto, que esta etapa é cumprida no âmbito organizacional, através da decisão das áreas a serem atendidas na modelagem multidimensional e, conseqüentemente, do DW.

Nesta fase são avaliados os detalhes a respeito do modelo de negócio sobre o qual se dará o desenvolvimento do projeto, analisando-se os princípios básicos de quais requisitos de negócio serão traduzidos em objetos físicos na base de dados.

Ainda nesta fase procura-se determinar quais fontes de dados serão utilizadas na implementação do projeto, além de se prever a utilização de aplicativos para se obter tais fontes.

Durante todo o processo procede-se à documentação das tarefas realizadas. O resultado desta etapa consiste em uma documentação que relacione:

- finalidade do projeto;
- limite do escopo do negócio a ser modelado;
- ferramentas e aplicativos a serem utilizados.

Observa-se, portanto, que a Etapa 1 da metodologia abrange o processo decisório gerencial que estabelece as diretrizes do modelo a ser gerado e, subseqüentemente, implementado. Neste momento, não é necessário o detalhamento do nível de atributos ou funções. Nesta fase, o interesse volta-se para a observação apenas das estruturas preliminares, genéricas, verificadas a partir dos requisitos dos usuários e nos esquemas conceituais ER existentes.

5.3 Etapa 2: gerar o modelo multidimensional

Todos os procedimentos devem ser seqüencialmente concluídos para que uma nova etapa seja iniciada. Na etapa 2 da metodologia, após definido o escopo do negócio a ser modelado, são identificadas as bases transacionais com suas respectivas funcionalidades no meio operacional, que originam o modelo multidimensional.

Partindo do modelo de negócios existente na organização, projetado através do modelo ER, analisam-se as dimensões que podem vir a fazer parte do processo e como podem ser agregados os valores detectados. Inicialmente, as dimensões são consideradas de maneira geral, para posterior detalhamento de atributos e suas hierarquias.

Deve existir a correlação entre os esquemas conceituais do banco de dados operacional, observando os atributos das entidades e relacionamentos que identificam fatos em um esquema ER. Esta análise pode indicar dimensões implícitas no negócio a ser modelado.

Para gerar o modelo multidimensional, portanto, faz-se necessário desmontar hierarquias, que são os relacionamentos normalizados do modelo ER, anexando as informações relevantes às tabelas Fatos ou Dimensão correspondentes.

Esta fase da metodologia caracteriza-se pela definição dos aspectos iniciais do modelo de negócio, que correspondem à definição da granularidade, dimensões e fatos. Quando são estabelecidas as dimensões e fatos, são declarados seus respectivos atributos. Considerando a granularidade fixada neste estágio, são estabelecidas as medidas de derivação. Estes procedimentos são descritos subseqüentemente, compondo a etapa 2 da metodologia.

5.3.1 Granularidade

A granularidade do fato é o nível de detalhe com que são gravados os dados. Para que possam ser efetivamente analisados, os dados devem estar no mesmo nível de granularidade. Como uma regra geral, os dados podem ser mantidos com o maior nível de detalhamento e, posteriormente, sumarizados, gerando um nível mais baixo de granularidade.

Modelos multidimensionais que expressam dados no mais baixo nível de detalhe geram flexibilidade e extensabilidade máxima. Os dados atômicos podem ser

resumidos de qualquer modo. Igualmente, os dados atômicos podem ser estendidos com atributos adicionais, medidas ou dimensões sem romper com processos existentes (Kimball e Ross, 2004).

Observando a questão da granularidade, pode-se considerar, também, a aditividade, que é a habilidade das medidas serem resumidas. A aditividade torna-se importante quando ocorre a possibilidade de sumarização em tabelas de fato. Geralmente é desejável que as medidas possam ser completamente aditivas, pois quando não são, deve-se considerar rompê-las em seus elementos atômicos.

O modelo multidimensional deve ser povoado com dados atômicos para que os usuários possam elaborar questões precisas de consulta (Kimball e Ross, 2004). Com a definição de uma granularidade baixa é possível responder a praticamente qualquer solicitação do usuário, porém, aumenta a utilização de recursos computacionais para atender tais solicitações. Inversamente, com uma alta granularidade, ocorre uma significativa redução da possibilidade de utilização dos dados para atender consultas detalhadas, porém, reduz-se o armazenamento em disco e os índices necessários.

Definidas a granularidade e aditividade existentes nos fatos, pode-se considerar a possibilidade de se fazer combinação de fatos e, se for o caso, conseqüentemente, podem ocorrer mudanças na granularidade. Portanto, conhecer o negócio a ser modelado e seus objetivos (Etapa 1), é imprescindível para a definição correta da granularidade do modelo.

5.3.2 Dimensões e Fatos

Considerando que em uma análise do modelo multidimensional, as medidas são critérios de avaliação e as dimensões o que está sendo analisado, deve-se identificar medidas e dimensões de acordo com os requisitos estabelecidos pelo usuário. A definição de quais atributos serão utilizados deve assegurar que as necessidades do usuário serão satisfeitas e que as respostas serão obtidas através destes atributos.

Considerando que um grupo de dimensões e suas medidas associadas compõem os fatos, deve-se organizar as dimensões e medidas nos fatos do modelo proposto. Para toda medida que descreve exatamente o mesmo grupo de dimensões, pode-se criar apenas um fato.

Ter disciplina para criar tabelas de fatos com um único nível de detalhe assegura que medidas não serão inapropriadamente duplicadas. Uma tabela de fatos de granularidade mista só pode ser examinada por uma aplicação específica nos níveis variados de detalhe, validando efetivamente consultas *ad hoc*. Se as medidas existem naturalmente em diferentes granularidades, então se pode estabelecer uma tabela de fatos para cada nível, protegendo aplicações existentes das mudanças que venham a ocorrer (Kimball e Ross, 2004).

Devem ser analisados os elementos de cada dimensão e considerada a distinção entre as dimensões simples, que não apresentam informações adicionais e as dimensões mais elaboradas, que apresentam informações adicionais ou definições de hierarquias. Portanto, são classificadas as dimensões de acordo com sua complexidade (Buzydlowski et al., 1998):

- dimensão não associativa: são as tabelas dimensão que não possuem informação adicional sobre os elementos multidimensionais, como em um esquema Estrela simples;
- dimensão associativa: são aquelas tabelas dimensão que têm informações hierárquicas ou informações adicionais sobre os elementos multidimensionais, como em um esquema Floco de neve.

Observando a classificação proposta por Buzydlowski et al. (1998), são mapeadas as dimensões definidas como associativas e não associativas. Para as classes associativas, os elementos de cada nível são mapeados em uma dimensão, tornando-se uma classe separada e a hierarquia entre os níveis é representada como atributos adicionais (pais e filhos). A dimensão propriamente é mapeada para uma classe base para as hierarquias com ligações para o primeiro nível abaixo.

Se diferentes atributos em uma classe pertencem a diferentes níveis de dimensão, a classe pode ser separada. Atributos operacionais que não são importantes para a análise do negócio podem ser descartados.

5.3.2.1 Dimensão Tempo

Vale destacar a necessidade de se definir a dimensão Tempo para todo e qualquer DW a ser construído, considerando que o contexto de *data warehousing* baseia-se na equivalência de tempo, considerando a perspectiva de análise de negócio, agregando dados por dia, semana, mês ou ano, por exemplo.

Portanto, se o negócio que está sendo analisado no DW já possui este parâmetro, deve-se incluí-lo como atributo de tempo, caso contrário, faz-se necessária a inserção de um atributo com a característica de preservar o tempo da ação no contexto organizacional. A maioria dos negócios modelados exige que o tempo seja armazenado em dias, considerando o mais baixo nível de granularidade para um atributo de data. Porém, analisando os requisitos, pode-se observar a necessidade de se gerar dados por dia, mês e/ou ano.

5.3.2.2 Medidas de derivação

Neste momento, as informações já definidas são consideradas e inicia-se o processo de refinamento e detalhamento do modelo. Podem ser detectados atributos novos e atributos desnecessários podem ser eliminados.

Embora os usuários não se preocupem com detalhes de transações simples, as questões de consulta envolvem sumarização dos detalhes de forma nem sempre previsível. Este problema surge quando consultas freqüentes requerem dados altamente sumarizados. Consultas em tempo de execução geralmente não são satisfatórias e é necessária a materialização de dados agregados. Além disso, ao materializar uma agregação, tabelas de dimensão adicionais (com a granularidade apropriada) devem ser geradas para assegurar resultados corretos. Esta pode ser uma razão adicional para a normalização parcial de uma tabela de dimensão.

Portanto, propõe-se criar mecanismos para sumarizar informações, evitando a necessidade de execução do processo de sumarização em toda consulta, através de regras estabelecidas na metodologia proposta:

- para a sumarização dos dados devem ser utilizadas as funções *SUM, AVG* e *COUNT*, ou equivalentes;
- quando não existir a necessidade de análise atômica no dado, este deve ser sumarizado;
- quando existir a necessidade de se utilizar o dado em sua forma atômica, manter o mais baixo nível de granularidade ou manter na tabela Fatos os dados sumarizados e uma hierarquia (ou similar) com os dados atômicos.

5.3.3 Relacionamentos

O relacionamento entre as entidades no modelo multidimensional normalmente é de cardinalidade 1:n, considerando que os identificadores das Dimensões são atributos nos Fatos. Porém, relacionamentos n:n também são possíveis para a representação do negócio modelado. Quando a cardinalidade de uma dimensão com fatos é n:n, devem ser analisadas e consideradas as possíveis soluções para o problema. Uma análise comparativa de propostas para o problema de relacionamentos muitos-para-muitos entre tabelas Fatos e Dimensão no modelo multidimensional é proposto no trabalho de Rowen (2000), incluindo a solução proposta por Kimball (1998), de uma tabela *bridge* entre Dimensão e Fatos.

5.3.3.1 Hierarquias de classificação

O próximo passo é definir e consistir as hierarquias obtidas das dimensões previamente detectadas. Este refinamento estabelece as propriedades para os atributos das hierarquias, analisando a possibilidade da conversão em atributos descritores.

As associações de entidades são utilizadas para representar os relacionamentos entre níveis de uma hierarquia de classificação. Nesta etapa, são necessários os seguintes procedimentos:

- identificar as hierarquias de classificação;
- representar os DAGs das hierarquias;
- identificar atributos que descrevem a hierarquia e outros atributos utilizados na análise do negócio em cada nível da hierarquia;
- associar as hierarquias com dimensões que suportem os atributos identificados anteriormente.

5.3.3.2 Exatidão e completeza

A completeza caracteriza o grau no qual os elementos de um esquema estão presentes nas instâncias. No DW, os problemas de completeza podem ser causados pela carga incompleta na tabela de fatos, dimensões incompletas que podem acarretar problemas de integridade referencial da tabela de fatos em relação à dimensão, além do tratamento de valores nulos (Campos e Amaral, 2004).

Portanto, as características de exatidão e completeza são definidas nos procedimentos de associação das hierarquias de classificação, considerando o processo de limpeza dos dados operacionais.

Relacionamentos entre uma classe Fatos e classes Dimensão podem ser associações de exatidão, especificando que para cada objeto na classe Fatos existe um objeto nas classes Dimensão.

O resultado da etapa 2 consiste em documentar os seguintes elementos, gerando um esquema conceitual preliminar, a ser utilizado nas fases seguintes:

- determinar a granularidade do modelo, estabelecendo o nível atômico ou resumido para os elementos, ou ainda sua combinação;
- gerar a tabela de fatos, a partir do escopo do problema a ser modelado, com respectivos atributos;
- relacionar as dimensões não associativas, que estão ligadas aos fatos, com respectivos atributos;
- relacionar as dimensões associativas, se for o caso, optando pelo esquema Floco de Neve, com respectivos atributos;
- verificar os atributos existentes para a geração da dimensão Tempo, definindo sua granularidade ou, se for o caso, estabelecer parâmetros para sua concepção;
- definir as medidas de derivação e suas respectivas regras;
- verificar as hierarquias de classificação, definindo se estas serão agregadas às dimensões associativas ou estabelecer os DAGs, gerando o modelo Floco de Neve;
- verificar a existência de valores nulos, que não atendam aos requisitos de exatidão e completeza estabelecidos para o modelo;
- verificar os relacionamentos gerados no modelo multidimensional, comparando-os com o modelo operacional original, validando seus atributos e referências para atender o modelo proposto, sem mudar ou perder a regra de negócio.

5.4 Etapa 3: Representar o modelo através do diagrama de classes

Nesta etapa o modelo multidimensional está definido, representado pelo esquema Estrela, Floco de neve ou similar. Portanto, as entidades e hierarquias do modelo são classificadas e separadas e, posteriormente, são agregados os dados de transação.

Um diagrama Floco de Neve UML consiste de um diagrama de classes UML que atenda os seguintes requisitos (Jensen et al., 2003), conforme mostra a figura 19:

- as extremidades da classe origem (que corresponde aos fatos) para as classes dimensão representam o nível mais baixo nas dimensões e são associações com cardinalidade 0..1;
- as extremidades entre classes dimensão em uma dimensão são associações do tipo agregação com cardinalidade 1..1;
- uma classe conectada à classe origem por uma ligação do tipo de associação junto com todas as classes que têm um caminho de ligações do tipo agregação é chamado de subgráfico, não existindo extremidades entre subgráficos;
- nenhuma extremidade existe entre classes no mesmo subgráfico se o comprimento dos caminhos da classe origem para estas duas classes for igual.

As restrições no diagrama Floco de Neve são escolhidas com cuidado para assegurar a ação de sumarização dos dados, ou seja, assegurar que as consultas OLAP retornem resultados corretos.

5.4.1 Classes

A partir do modelo multidimensional projetado, as dimensões e fatos são representados pelas classes Dimensão e Fatos, respectivamente, sendo que a classe Fatos representa os fatos e suas medidas, definidas como atributos nestas classes.

Se uma entidade gera uma classe não abstrata contendo apenas atributos simples, pode-se representá-la através de uma diagramação de classes e seus atributos. Porém, neste processo, pode-se gerar tipos de dados complexos, através da identificação de soluções a serem implementadas na orientação a objetos, que diferem do paradigma relacional.

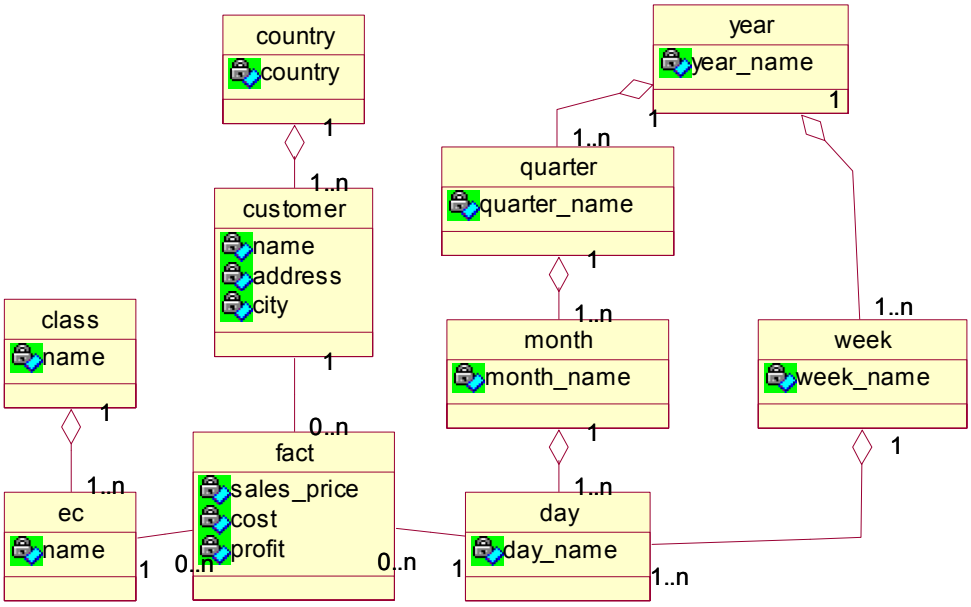

Figura 19 - Diagrama Floco de Neve UML.

Fonte: Jensen et al. (2003, p. 327).

5.4.2 Relacionamentos

A flexibilidade da agregação compartilhada na UML permite representar o relacionamento muitos-para-muitos entre a classe Fatos e classes Dimensão, registrando que uma instância de um objeto da classe Fatos pode ser relacionada a uma ou mais instâncias do objeto das Dimensões.

Classes Dimensão que possuem características semelhantes podem ser modeladas através do relacionamento do tipo generalização. As características adicionais das classes dimensão podem ser representadas através do relacionamento do tipo especialização, usando os conceitos de herança da orientação a objetos. As classes de dimensão podem pertencer a uma hierarquia de classificação e especialização ao mesmo tempo.

5.4.3 Identificadores e Descritores

Quando se cria um objeto, ele recebe um identificador (OID), que deve ser único para cada objeto (Khoshafian, 1994; Nassu e Setzer, 1999). Como os OIDs são identificadores únicos, são utilizados nos BDOO para estabelecer referências entre

objetos, oferecendo uma maneira de se recuperar os objetos do banco de dados. Este identificador é gerado quando o objeto é criado, tornando-se permanente.

Além dos identificadores de objeto, podem ser associados descritores aos objetos. Um descritor é um atributo de um nível, usado para selecionar suas instâncias (Abelló, 2005). Estes atributos descritores, com a restrição {D} especificada, além de utilizados na fase de análise de dados pelas ferramentas OLAP, definidos em cada um dos níveis da hierarquia de classificação no diagrama, também podem descrever cada instância do objeto.

5.4.4 Refinar o modelo

O modelo operacional pode dar origem a dimensões grandes ou complexas e o processo simples de desnormalização pode gerar grande redundância e, conseqüentemente, problemas de manutenção. A normalização parcial da dimensão pode ser uma contraposição entre desempenho e redundância, sendo necessária à validação dos dados.

Portanto, para finalizar o modelo multidimensional, deve-se rever o negócio a ser modelado, verificando se os requisitos foram devidamente cumpridos.

Características a serem observadas no modelo gerado:

- os atributos da classe Fatos atendem às especificações do negócio?
- os relacionamentos oferecem a real representação de ligação entre as dimensões e os fatos?
- classes semelhantes estão associadas através da herança?
- as hierarquias de classificação foram devidamente associadas e continuam representando caminhos de associação?

O resultado da etapa 3 consiste em documentar os seguintes elementos:

- definir o diagrama de classes que vai representar o modelo multidimensional, observando os procedimentos:
  - identificar as classes;
  - identificar os atributos das classes, considerando sua importância para os objetivos do DW;
  - verificar a existência de classes semelhantes, remodelando-as através de herança;

- verificar a possibilidade de se utilizar o diagrama de estrutura composta para a representação de composições;
- validar os relacionamentos do modelo multidimensional observando os conceitos do paradigma orientado a objetos, transformando os relacionamentos em associação, especialização, generalização e agregação;
- relacionar os identificadores para cada classe, através da restrição {OID};
- relacionar um único descritor para cada classe, através da restrição {D};
- refinar o modelo multidimensional.

5.5 Etapa 4: Mapear o diagrama de classes para Banco de Dados OO

O projeto de banco de dados geralmente inclui as fases: conceitual, lógica e física. A desnormalização pode ser separada destas fases porque envolve aspectos que não estão puramente relacionados nem ao projeto lógico nem ao físico. O processo de desnormalização deve ser implementado entre o mapeamento do modelo de dados e o projeto físico e, com isto, integrado com o projeto lógico e físico (Shin e Sanders, 2005).

Para implementar a metodologia apresentada, o modelo de dados é gerado a partir de diferentes níveis de abstração:

- nível conceitual - representação conceitual dos dados operacionais;
- nível lógico - representação do modelo de dados lógico a partir do modelo do DW;
- nível físico - especificação dos dados armazenados.

Os relacionamentos entre os níveis conceitual e lógico, e entre o lógico e o físico são representados explicitamente pela especificação do mapeamento entre os objetos correspondentes dos diferentes níveis. A figura 20a representa o fluxo de mapeamento da implementação tradicional do modelo multidimensional e a figura 20b representa o fluxo dos níveis para a implementação do modelo apresentado.

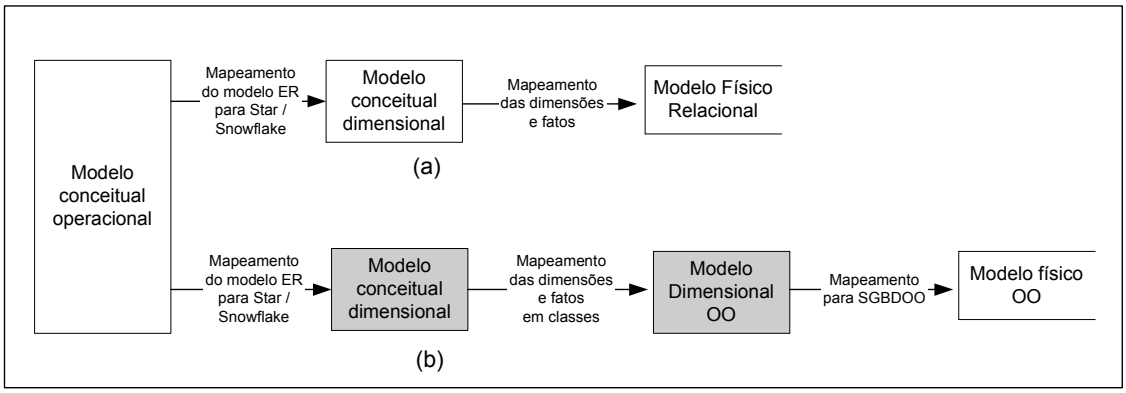

Figura 20 - Fluxo da metodologia.

O armazenamento físico do modelo relacional é composto de tuplas de valores, que são o único tipo de objeto do nível lógico. Porém, o modelo orientado a objetos é composto de representações de objetos com restrições, herança, polimorfismo e encapsulamento. O mapeamento é um processo em que cada elemento de dado deve ser mapeado e definido no repositório. O modelo de dados, incluindo suas classes e atributos, é transferido do estágio de modelagem para o repositório de dados.

O mapeamento de um modelo de dados UML para um esquema orientado a objetos poderia ser direto, se os fornecedores de SBDOO aderissem a um padrão de definição de esquemas. Questões surgem da natureza inerente do projeto de objetos persistentes, aparecendo nos produtos de SBDOO e no padrão ODMG. O padrão ODMG abrange a linguagem de consulta OQL (*Object Query Language* - linguagem de consulta de objetos) e os recursos básicos que um SBDOO deve apresentar, não oferecendo interoperabilidade. Porém, considerando-o como o atual padrão existente (Turowski, 2000; Muller, 2002), as especificações desta seção são definidas observando os padrões estabelecidos pela ODMG.

A ODMG definiu uma linguagem de especificação usada para representar objetos em sistemas de banco de dados. Esta linguagem de especificação independe de linguagem de programação e é usada para definir o esquema, operações e estado dos objetos em bancos de dados. A ODL - *Object Definition Language* - é uma linguagem para definir as especificações dos tipos de objetos de acordo com o modelo de objetos ODMG (ODMG, 2005; Cattell et al., 2000).

Os SGBDs usam a linguagem de definição de dados DDL (*Data Definition Language*) e a linguagem de manipulação de dados DML (*Data Manipulation*

Language). A DDL permite aos usuários definir os tipos de dados enquanto a DML permite criar, apagar, ler e alterar instâncias dos tipos de dados. A ODL é a DDL para objetos, definindo as características dos tipos, incluindo suas propriedades e operações (ODMG, 2005). Este padrão é utilizado na próxima seção para o mapeamento do modelo multidimensional para o modelo de banco de dados orientado a objetos.

5.5.1 Padrão ODMG

O modelo de objetos do ODMG 3.0 tem dois tipos de propriedades para representar o estado do objeto: o atributo, que define o estado de um tipo, e o relacionamento entre objetos, que devem possuir instâncias que podem ser referenciadas pelo identificador de objetos (Cattell et al., 2000).

A especificação do relacionamento nomeia e define um caminho através de um relacionamento. Porém, a definição do relacionamento é referenciada como a parte estática de cada objeto, não suportando a semântica comportamental que é essencial em relacionamentos "todo-parte" (Lee et al., 2005).

Em ODL, classes são declaradas através de uma *interface* (usando a palavra chave *interface*); cada classe é uma coleção de atributos primitivos (de um tipo de dados básico) ou relacionamento, ou seja, seu valor é um objeto ou conjunto de objetos de uma determinada classe. A *interface* inclui: um nome, uma chave (opcional), uma *extent* e as propriedades e operações. Uma chave é um conjunto de propriedades (atributos ou relacionamentos) cujos valores são únicos para cada objeto na extensão (chaves compostas são chamadas de combinação na ODMG). A extensão é o nome do conjunto de objetos abaixo da declaração (Badia, 2005).

Valores de atributos são tipicamente literais (atômico ou complexo), mas podem ser OIDs. Valores de relacionamentos são sempre nomes de objeto ou uma coleção aplicada para nomes de objetos. No modelo ODMG apenas relacionamentos binários são representados explicitamente, através de um par de referências inversas (usando a palavra chave *inverse*) (Badia, 2005; ODMG, 2005).

Valores de literais podem ser simples ou complexos. Há três tipos de literais: atômico, coleção e estruturado. Literais atômicos correspondem a tipos de dados básicos: *long, short, unsigned long, unsigned short*, *float, double, boolean*, *octet* e *char*. Literais estruturados têm uma estrutura de tupla, incluindo tipos como *date* e

time. O usuário pode definir literais estruturados (usando o construtor *Struct*). Literais de coleção especificam uma coleção de objetos ou literais. Tipos de coleções são: *set, bag, list array* e *dictionary*. Cada coleção tem um grupo de operadores embutidos (Cattell et al., 2000; Badia, 2005).

Finalmente, objetos têm identificadores de objeto (OID) e um valor (diferente das literais que têm valor, mas não OID). Objetos podem ter um nome, sendo atômicos ou tipo coleção. Para cada objeto, propriedades (atributos e relacionamentos) e operações são especificadas (Cattell et al., 2000; Badia, 2005).

Cada classe representada na UML é transposta para uma classe em ODL e cada associação identificada como um atributo de relacionamento com um inverso. Atributos complexos tornam-se atributos estruturados, e atributos multivalorados podem ser gerados com um construtor *Set* (descrito posteriormente). Classes de associação podem ser tratadas como classes regulares, considerando que em ODL não se pode tratar diretamente relacionamentos com atributos. A herança é modelada diretamente tanto em UML quanto em ODL (Badia, 2005).

5.5.1.1 Gerar classes

Uma classe define um conjunto de propriedades através das quais os usuários podem acessar o estado das instâncias destas classes. Dois tipos de propriedades são definidos no modelo de objetos da ODMG: atributos e relacionamentos (Cattell et al., 2000; ODMG, 2005).

A ODL versão 3.0 supõe que as classes são persistentes e a sintaxe que define esta persistência é específica de cada fabricante (Cattell et al., 2000). Nesta etapa são geradas as classes, como exemplificado na figura 21a.

A partir da especificação de uma superclasse, são geradas as classes especializadas, com as características herdadas da superclasse, e agregando suas próprias características. O padrão ODMG define a extensão de um tipo como sendo o conjunto de todas as instâncias do tipo em um determinado banco de dados. Em um BDOO pode-se manter automaticamente o conjunto de extensões independente da propriedade ou localização de objetos de um determinado tipo. Com a ODL, basta especificar a palavra-chave de extensão e um nome, como o trecho da figura 21b.

5.5.1.2 Gerar identificadores

A referência do objeto é analisada através do identificador de objetos, não de valores de atributos no objeto, não existindo os conceitos de chave primária ou chave estrangeira, mas os marcadores {OID}, que definidos, transformam-se em restrição à extensão da classe. As implementações não envolvem os OIDs, pois quando um objeto é instanciado no banco de dados, obtém um OID que é parte conceitual do modelo OO e a ODL não prevê a representação dos OIDs.

```
class Superclasse
{
attribute string supatributo1;
attribute date supatributo2;
attribute string supatributo3;
}
```

(a)

```
class Subclasse1 extends Superclasse
{
attribute string sub1atributo1;
attribute string sub1atributo2;
}
```

```
class Subclasse2 extends Superclasse
{
attribute string sub2atributo1;
attribute string sub2atributo2;
}
```

(b)

Figura 21 - Geração de superclasse e classe especializada em ODL.

5.5.1.3 Gerar descritores

As propriedades de identificação do objeto (OID) e descritor do objeto (D) são representadas através da utilização do diagrama de classes da UML, porém, ignorados no mapeamento, considerando que o OID é gerado automaticamente pelo SGBDOO e o descritor é uma propriedade lógica do modelo multidimensional proposto. As regras de derivação de atributos, aditividade, exatidão e completeza não são especificadas diretamente pela ODL, portanto, devem ser consideradas a partir dos conceitos definidos e implementados através das regras da OO.

5.5.1.4 Gerar relacionamentos

Os BDOOs não suportam associações "n-árias" como relacionamentos de primeira classe e, portanto, o modelo de objetos do padrão ODMG (ODMG, 2005) não oferece uma maneira de declarar tais associações, propondo como solução deste problema a criação de classes de associação.

Em ODL, um relacionamento é definido explicitamente pela declaração de caminhos transversos que permitem aplicações usando conexões lógicas entre os objetos participantes do relacionamento. Caminhos transversos são declarados em pares, um em cada direção do relacionamento. A geração de um relacionamento inclui a definição de um tipo destino, a cardinalidade do lado destino e informações sobre o relacionamento inverso, do lado origem.

Os relacionamentos 1:n e n:n podem ser representados explicitamente em um banco de dados orientado a objetos, que tem a capacidade de armazenar coleções.

Através da declaração *set* pode-se indicar uma coleção não-ordenada de elementos sem duplicatas, usado para definir relacionamentos entre classes. Para identificar a cardinalidade do relacionamento do lado destino, deve-se consultar o relacionamento inverso. O quadro 10 mostra a especificação dos relacionamentos em ODL da ODMG.

Tipo de relacionamento	**Especificação**	***Links***
um-para-muitos	Relationship X Y inverse Z	Tratados como atributos
muitos-para-muitos	Relationship set<X> Y inverse Z	Tratados como coleções

Quadro 10 - Relacionamentos definidos na ODL

O modelo de objetos estabelecido pela ODMG inclui os relacionamentos de herança baseados em tipos e subtipos, que são comumente representados por gráficos, em que cada nó é um tipo e cada arco conecta um tipo (chamado supertipo) a outro tipo (chamado subtipo). Esta ligação é conhecida por relacionamento *ISA* ou relacionamento de generalização / especialização. A figura 22 mostra o exemplo dos relacionamentos.

```
class ClasseExtend extends Superclasse : Relac_isa {
attribute string exatributo1;
attribute date exatributo2;
relationship Relac_ext estende inverse Relac_ext::isa;
}

class ClasseSet extends Classeextend : Relac_set {
relationship set<Relac_isa> isa
inverse Relac_isa:: estende;
}
```

Figura 22 - Relacionamentos *ISA* e *EXTENDS* em ODL..

Além do relacionamento *ISA*, a ODMG define um relacionamento denominado *EXTENDS* para a herança de estado e comportamento. É um relacionamento de herança simples entre duas classes pelo qual as classes subordinadas herdam todas as propriedades e comportamentos da classe mais

Um relacionamento 1:n entre uma Classe A qualquer e uma Classe B, pode ser representado em ODL segundo o *script* da figura 23.

```
class ClasseA                          class ClasseB
{                                      {
...                                    …
relationship set<ClasseB> rel_um       relationship ClasseA rel_n
inverse ClasseB::rel_n;                inverse ClasseA:: rel_um;
…                                      …
}                                      }
```

Figura 23 - Relacionamento 1:n em ODL.

O relacionamento de agregação não é representado diretamente em ODL, portanto, deve ser transformado em um relacionamento de associação. Se uma associação entre uma Classe A qualquer e uma Classe B tem uma multiplicidade muitos-para-muitos e esta associação tem suas propriedades ou restrições específicas, o padrão ODMG gera uma classe de associação entre os relacionamentos (Jones e Song, 2002; Philippi, 2004; Song et al., 2005; ODMG, 2005), conforme o *script* da figura 24a. Nas Classes A e B, respectivamente, são gerados os relacionamentos correspondentes em ODL, conforme mostra a figura 24b.

A vantagem em uma associação simples em UML advém do encapsulamento e da conseqüente limitação de acoplamento das classes relacionadas. Se a multiplicidade for 0..1 ou 1..1 a ODL provê a declaração de um atributo de um tipo de classe, ocultando a implementação da estrutura de dados. A literal estruturada é um tipo de literal suportada pelo modelo de objetos. Além dos tipos de estrutura suportados pela ODMG, que são *date, interval, time e timestamp*, pode-se definir outras estruturas como extensão dos modelos de objeto (Cattell et al., 2000). A figura 25 mostra um exemplo de tipo de dado estrutura definido pelo usuário.

```
class Associativa
{
attribute currency asatributo1;
attribute long asatributo2;
relationship set<ClasseA> ItemA inverse ClasseA::associaA;
relationship set<ClasseB> ItemB inverse ClasseB::associaB;
}
```

(a)

```
class ClasseA
{
attribute int Aatributo1;
attribute currency Aatributo2;
attribute long Aatributo3;
relationship set<Associativa> associaA inverse Associativa::ItemA;
}

class ClasseB
{
attribute string Batributo1;
attribute string Batributo2;
attribute string Batributo3;
relationship set<Associtativa> associaB inverse Associativa::ItemB;
}
```

(b)

Figura 24 - Relacionamentos de associação em ODL..

```
Struct TipoClasse
{
String atributo1;
String atributo2;
}
```

```
class ClasseC
{
attribute TipoClasse;
attribute unsigned long atributo1;
attribute string atributo2;
...
}
```

Figura 25 - Tipo estrutura em ODL.

Entre os tipos literais atômicos na ODL, encontra-se o *enum*, um tipo gerador que a partir de um valor digitado, pode assumir um dos valores específicos e enumerados relacionados em sua declaração. O *script* da figura 26 exemplifica a declaração *enum* para um atributo com quatro valores possíveis:

```
{
 ...
attribute enum atributoenum {valor1, valor2, valor3, valor4};
...
}
```

Figura 26 - Tipo literal *enum*.

O mapeamento do modelo multidimensional gerado através da representação do diagrama de classes da UML, utilizando-se dos padrões definidos pela ODMG deve ser implementado em um BDOO. O padrão ODMG define formas de utilização das características do paradigma OO através da persistência de dados. Portanto, na etapa 4 da metodologia são analisados os padrões da ODMG a serem aplicados no modelo multidimensional.

O resultado da etapa 4 consiste em documentar os seguintes elementos:

- transposição das classes representadas no diagrama de classes;
- transposição dos relacionamentos de especialização / generalização para o relacionamento *ISA*;
- transposição dos relacionamentos de herança de estado e comportamento para o relacionamento *EXTENDS*;
- transformação dos relacionamentos de agregação em relacionamentos de associação;
- definição dos tipos de dados dos atributos de cada classe;
- validação dos atributos descritores;
- definição dos relacionamentos de associação e seus inversos;
- transformação dos atributos complexos em atributos estruturados;
- definição das restrições de valores através da declaração *enum*.

5.6 Etapa 5: Implementar o modelo

Observando o paradigma da OO pode-se agrupar objetos em classes. Estas classes podem ser encapsuladas tanto com as propriedades estáticas como com as dinâmicas destes objetos. As propriedades dinâmicas são as operações que podem ser aplicadas nos objetos para mudar suas características. Porém, no contexto de bancos de dados multidimensionais, os objetos (dados) são estáticos no sentido de que, uma vez que existam no sistema, eles não alteram suas características (propriedades estáticas), até que sejam levados a um armazenamento auxiliar. Conseqüentemente, as duas primeiras ações a serem aplicadas nestes objetos devem ser a de criá-los e destruí-los (Trujillo e Palomar, 1998).

Na modelagem multidimensional, os projetos lógicos e físicos são muito semelhantes e as diferenças apresentam-se em termos de detalhes especificados

para o banco de dados físico, incluindo nomes de coluna físicos, tipos de dados, declarações de chave e a permissibilidade de nulos (Kimball, 2002). No projeto físico observam-se ainda elementos básicos como ajuste de desempenho, particionamento e *layout* do arquivo, porém, que não fazem parte do escopo deste trabalho.

Considerações para o desenvolvimento do modelo físico (Oliveira, 1998, p.70):

- chaves alfanuméricas: devem ser substituídas por chaves numéricas, considerando que o tempo de busca de índices numéricos é menor que o tempo de busca de índices alfanuméricos;
- dependências transitivas: o processo de normalização pode remover atributos de descrição das entidades fracas, deixando-os apenas nas entidades fortes; portanto, mecanismos são gerados para que os atributos de descrição possam ser recuperados pelas entidades fracas.

As considerações acima relacionadas devem ser observadas em projetos de migração de sistemas organizacionais para um modelo de DW. O trabalho aqui apresentado pressupõe que tais situações devem ser tratadas a parte, em um outro contexto.

A implementação de um modelo relacional permite que se mapeie apenas tipos de dados entendíveis pelo modelo de banco de dados. Portanto, um modelo orientado a objetos deve prever a implementação dos tipos de dados e características apontados pelo paradigma da orientação a objetos.

A etapa 5 da metodologia considera que os padrões da OO estabelecidos pela ODMG são mapeados para um BDOO, partindo do modelo multidimensional projetado.

O resultado da etapa 5 consiste em documentar os seguintes elementos:

- detalhamento dos tipos físicos de dados de cada uma das classes Dimensão e Fatos;
- definição de índices;
- definição de restrições (tipos de dados e relacionamentos);
- definição da utilização das estruturas da OO disponíveis no BD definido;
- geração dos *scripts* de criação de classes e métodos no BD definido.

5.7 Resumo da proposta metodológica

O presente trabalho propõe uma metodologia para implantação de modelos multidimensionais mapeados em banco de dados orientado a objetos representados pela UML. A proposta baseia-se em um processo desenvolvido em 5 etapas e suas respectivas documentações, que devem ser seqüencialmente definidas e elaboradas. Cada uma das etapas impõe uma série de passos, resumidas no quadro 11 e na figura 27.

Etapa	Objetivo	Resultado	Documentação gerada
1	• Definir o negócio a ser modelado; • Observar estruturas genéricas; • Verificar os esquemas ERs existentes.	• Descrição da finalidade e limite do projeto; • Definir ferramentas e aplicativos.	• Finalidade do projeto.
2	• Definir o esquema conceitual preliminar; • Gerar o esquema Estrela ou Floco de neve.	• Definição da granularidade, fatos, dimensões e medidas de derivação; • Definição de hierarquias de classificação e relacionamentos.	• Esquema conceitual preliminar, através da modelagem multidimensional através do esquema Estrela ou Floco de neve.
3	• Gerar o diagrama de classes do esquema conceitual.	• Classes, atributos e relacionamentos; • Definição de identificadores, descritores; • Refinamento do modelo.	• Diagrama de classes e diagrama de estrutura composta do esquema Estrela ou Floco de neve.
4	• Mapear o modelo conceitual multidimensional para o paradigma da OO.	• Transposição de classes e relacionamentos do diagrama de classes para o padrão ODMG; • Transformação dos relacionamento de agregação em relacionamentos de associação; • Definição, transformação e validação dos tipos de dados e suas restrições.	• *Scripts* de criação dos componentes do modelo multidimensional através da ODL.
5	• Implementar o modelo proposto em um BDOO.	• Nomeação de classes e respectivos atributos; • Definição de restrições; • Validação e adaptação dos padrões da ODMG no BDOO; • Criação de classes Dimensão e Fatos; • Instanciação os objetos.	• *Scripts* da criação do modelo multidimensional para um banco de dados específico.

Quadro 11 - Resumo das etapas da metodologia.

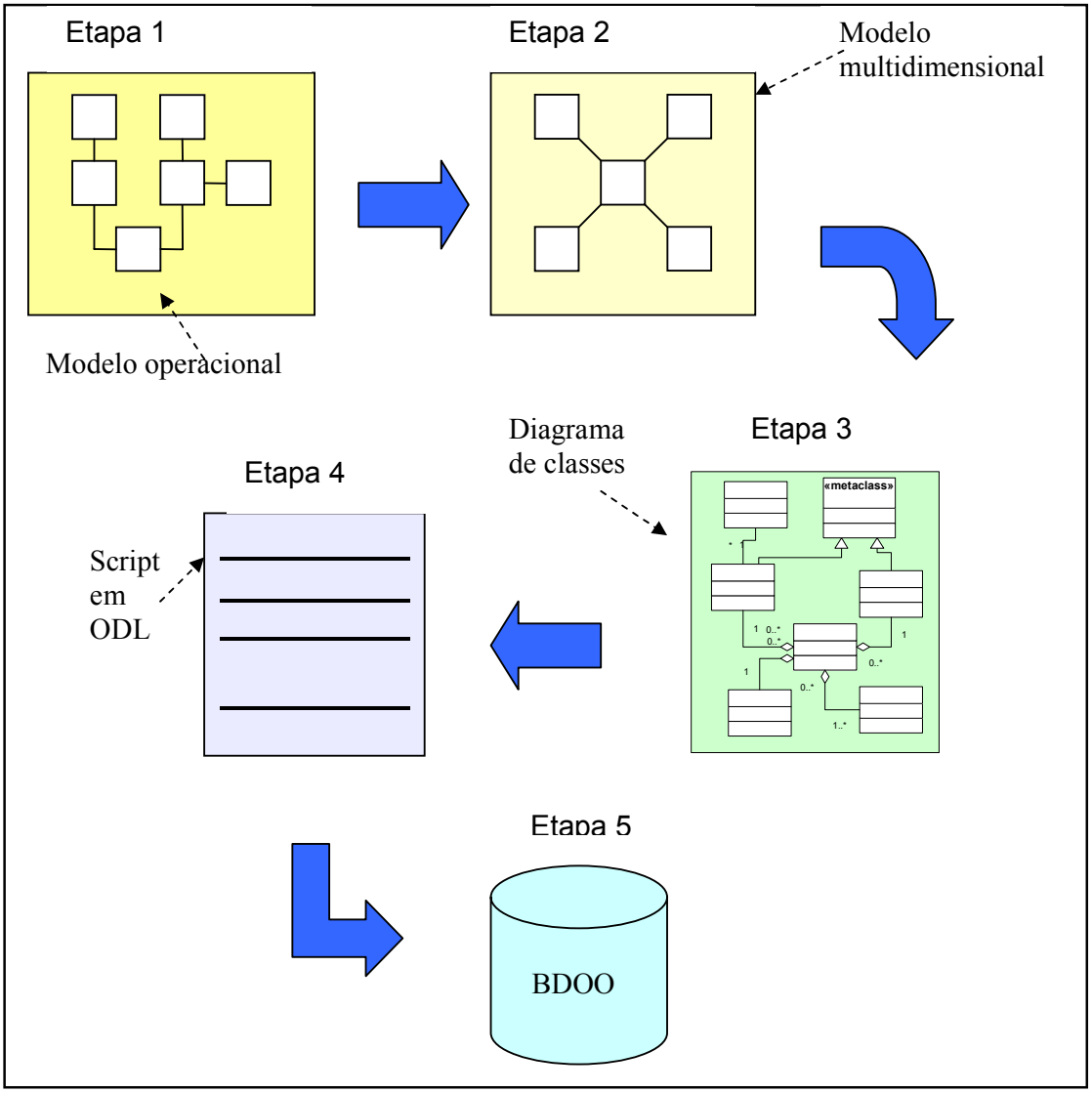

Figura 27 - Resumo da metodologia.

5.8 Estudo comparativo

Observando as metodologias analisadas no capítulo anterior, o estudo comparativo entre estes trabalhos com a metodologia proposta abrange as etapas definidas e suas respectivas tarefas.

5.8.1 Etapa 1

Esta etapa define o negócio a ser modelado, em que se observam estruturas genéricas nos esquemas ER existentes. Como resultado desta etapa, são descritos os limites e a finalidade do projeto, além da definição de ferramentas ou aplicativos.

A proposta de Kimball (1997, 2002) estabelece como etapa inicial do processo de DW a definição do negócio a ser modelado, a partir do modelo ER.

O trabalho de Moody e Kortink (2000) considera o modelo multidimensional a partir dos tradicionais modelos ER, porém sem a preocupação de formalizar a definição do negócio a ser modelado. A base da proposta concentra-se na representação das variações do modelo Floco de Neve.

A metodologia definida por Cavero et al. (2002, 2003) baseia-se no modelo conceitual ER e nos requisitos do usuário. O esquema inicial corresponde a definição do negócio a ser modelado.

Trujillo et al. (2000, 2003) e Abelló et al. (2002, 2005) não mencionam o modelo ER e os requisitos do usuário, baseando sua proposta diretamente sobre o modelo multidimensional.

Observando o objetivo do trabalho, a metodologia visa abranger todo o processo de modelagem do modelo multidimensional e sua implementação. Portanto, a primeira etapa da metodologia busca a origem do modelo no ambiente operacional, assemelhando-se neste sentido, às propostas de Kimball (1997, 2002), Moody e Kortink (2000) e Cavero et al. (2002, 2003), que relatam a origem do modelo multidimensional. Tais procedimentos não são abordados nas propostas de Trujillo et al. (2000, 2003) e Abelló et al. (2002, 2005).

Funcionalmente, na maioria dos casos, o modelo multidimensional é gerado a partir de modelos organizacionais existentes. Estes modelos são representados pelos diagramas ER. Portanto, a modelagem multidimensional deve buscar a origem dos dados no ambiente em que são efetivamente armazenados e, conseqüentemente, documentados. Observa-se a fundamental importância desta etapa para que o modelo multidimensional final possa ser validado com o modelo operacional, em uma análise de preservação e geração das informações originais.

5.8.2 Etapa 2

Segundo a metodologia proposta, esta etapa visa definir o esquema conceitual preliminar, gerando o modelo multidimensional através da representação no esquema Estrela ou Floco de Neve. Propõe-se a definição da granularidade, dos fatos, dimensões e medidas de derivação. Ainda são identificadas as hierarquias de classificação e os relacionamentos do modelo multidimensional.

A metodologia de Kimball (1997, 2002) define a granularidade no menor nível de detalhamento, visando atender a todos os requisitos do usuário. As dimensões são geradas a partir do processo de desnormalização de tabelas conectadas diretamente ao negócio a ser modelado, dando origem ao modelo Estrela, ou mantendo-se a normalização das dimensões, dando origem ao modelo Floco de Neve. Os fatos são identificados como sendo, geralmente, numéricos e aditivos. As medidas de derivação são analisadas como fatos calculados. As hierarquias de classificação não são detalhadas na proposta de Kimball (2002), e analisadas como tabelas de dimensão desnormalizadas. Quanto aos relacionamentos, são tratados com as mesmas considerações existentes no modelo ER.

O trabalho de Moody e Kortink (2000) não analisa a granularidade separadamente. As dimensões são nomeadas como entidades componentes ou classificação (desnormalizadas) e os fatos são nomeados como entidades de transação. As hierarquias são seqüências originadas de relacionamentos um-para-muitos, alinhados na mesma direção, no modelo ER. Estas hierarquias podem ser desmontadas (desnormalizadas) ou agregadas (Floco de Neve). A proposta aplica operador de agregação em uma entidade de transação para criar dados sumarizados e medidas derivadas.

A metodologia definida por Cavero et al. (2002, 2003) envolve várias etapas para definir dimensões e fatos. Inicialmente são definidas as dimensões e, na seqüência, suas hierarquias de agregação e, subseqüentemente, os domínios para estas dimensões. As hierarquias são originadas do banco de dados operacional. As medidas de derivação são atributos e funções de sumarização, em que usualmente é utilizada a função *SUM*. A tabela Fatos é definida a partir da associação das dimensões e hierarquias. A chave primária da tabela Fatos é composta das chaves estrangeiras das tabelas de dimensão.

Trujillo et al. (2000, 2003) definem fatos como classes compostas de relações de agregação com as dimensões, representando um relacionamento muitos-para-muitos entre várias dimensões. As dimensões são geradas a partir de objetos que fornecem características dos dados reais. A granularidade é determinada com base nos atributos definidos como medidas dos fatos. A hierarquia de classificação é gerada através de atributos nas dimensões e através de níveis de agregação.

Abelló et al. (2002, 2005) consideram que os Fatos representam os dados a serem analisados e as Dimensões mostram as diferentes visões destes dados. Os

relacionamentos identificados na metodologia são a classificação/ instanciação, generalização/ especialização e agregação/ composição. As medidas de derivação são geradas a partir da classificação e agrupamento dos dados e sua sumarização. O modelo sugere gerar a granularidade em seu mais alto nível, através da aplicação de funções de agregação nas medidas.

Considerando que a segunda etapa da metodologia proposta trata-se da concepção do modelo conceitual, os trabalhos estudados abordam, com algumas variações, os itens que compõem esta etapa.

As propostas de Kimball (1997, 2002) e Moody e Kortink (2000) não abordam conceitualmente as hierarquias de classificação, considerando-as apenas como dimensões não normalizadas. Portanto, neste aspecto, a metodologia proposta assemelha-se aos trabalhos de Trujillo et al. (2000, 2003) e Abelló et al. (2002, 2005).

Kimball (1997, 2002) propõe a granularidade em seu mais baixo nível, enquanto o trabalho de Abelló et al. (2002, 2005) sugere o inverso. Os demais trabalhos não sugerem regras para a definição da granularidade. Neste aspecto, a metodologia proposta indica sempre a menor granularidade, similar à proposta de Kimball (1997, 2002).

Quanto aos relacionamentos, os trabalhos de Kimball (1997, 2002), Moody e Kortink (2000) e Cavero et al. (2002, 2003) abordam os relacionamentos do modelo ER, com soluções equivalentes entre si. Trujillo et al. (2000, 2003) consideram os relacionamentos de agregação, enquanto Abelló et al. (2002, 2005) além da agregação, abordam a classificação e a generalização. A metodologia proposta, considerando os aspectos da OO, aborda todos os relacionamentos anteriormente citados. Neste contexto, observa-se a importância de tal abordagem, principalmente no aspecto da posterior implementação e não apenas em sua representação conceitual.

5.8.3 Etapa 3

Na etapa 3 da metodologia proposta o modelo multidimensional é mapeado para o modelo estático multidimensional orientado a objetos, representado através dos diagramas de classes e estrutura composta. São relacionadas as classes, respectivos atributos e relacionamentos. São definidos os identificadores e

descritores. Concluindo a etapa, o modelo é refinado através da validação com o modelo ER original.

A proposta de Kimball (1997, 2002) não aborda a orientação a objetos, tratando dos paradigmas relacional para a representação do modelo multidimensional. Conseqüentemente, os conceitos de identificadores e descritores não são utilizados na metodologia do autor. Considerando que a metodologia de Moody e Kortink (2000) é uma variação da proposta de Kimball (1997), sua proposta também não contempla o paradigma orientado a objetos. Assim como Cavero et al. (2002, 2003), que enfocam a metodologia estruturada de desenvolvimento de software, não abordando a orientação a objetos.

O modelo GOLD de Trujillo et al. (2000, 2003) aborda os aspectos da orientação a objetos, incluindo os conceitos da UML. O modelo apresenta os elementos básicos do modelo multidimensional – Dimensão e Fatos – como classes Dimensão e Fatos. Os atributos são definidos caracterizando os objetos das classes. Os autores utilizam o diagrama de classes para a representação do modelo de dados. A metodologia, porém, não apresenta regras para a geração do modelo multidimensional a partir de um modelo operacional. Apenas aplica os conceitos da orientação a objetos no modelo multidimensional.

O modelo YAM^2 proposto por Abelló et al. (2002, 2005) considera a orientação a objetos e a dualidade Fatos-Dimensões é representada por classes Fatos e Dimensão. Descritores são identificados como um atributo usado para selecionar suas instâncias. Nos três níveis de desenvolvimento do modelo, o nível intermediário (segundo nível) aborda a definição dos níveis de dimensões e fatos que formam uma hierarquia. O nível mais baixo mostra os atributos, incluindo os descritores.

As propostas de Kimball (1997, 2002), Moody e Kortink (2000) e Cavero et al. (2002, 2003) possuem enfoque na metodologia estruturada de desenvolvimento de software, não abordando a orientação a objetos. Os trabalhos de Trujillo et al. (2000, 2003) e Abelló et al. (2002, 2005) abordam a OO e sua representação através do diagrama de classes da UML.

Neste contexto, a proposta diferencia-se das demais na solução dos relacionamentos n:n, adaptando a solução proposta por Kimball (1997), através da utilização do diagrama de estrutura composta, encapsulando, porém, solucionando a questão deste relacionamento. Neste sentido, observa-se a importância de tal procedimento, considerando a grande utilização do relacionamento n:n para a

representação dos itens de dados. Os trabalhos analisados omitem este problema, apesar da utilização desta representação (n:n) nos exemplos das propostas.

Nesta etapa, a metodologia propõe o refinamento do modelo, através da comparação da funcionalidade com o modelo operacional, diferenciando-se dos demais trabalhos. Este aspecto é extremamente importante para que consultas *ad hoc* possam ser atendidas e que o modelo possa ser referenciado com os dados do modelo operacional original.

5.8.4 Etapa 4

Na fase subseqüente são gerados os *scripts* para a criação do modelo multidimensional orientado a objetos, abordando o padrão ODL. As classes e relacionamentos estáticos dos diagramas de classes e estrutura composta são definidos pela linguagem padrão, estabelecida pela ODMG.

Os trabalhos de Kimball (1997, 2002), Moody e Kortink (2000) e Cavero et al. (2002, 2003) não abordam o paradigma orientado a objetos e, conseqüentemente, não oferecem tratamento para a etapa 4 da metodologia proposta.

A proposta de Trujillo et al. (2000, 2003) aborda aspectos da multidimensionalidade e o paradigma da orientação a objetos conceitualmente, não implementando a proposta.

Abelló et al. (2002, 2005) fazem a validação da proposta através de uma aplicação, modelando os níveis conceituais propostos. Como o modelo aborda apenas o modelo conceitual, o tratamento e o mapeamento para posterior implementação não são considerados.

A etapa 4 da metodologia proposta não é abordada em nenhum dos trabalhos analisados, considerando que são apresentados apenas no nível conceitual. Portanto a etapa 4 e, conseqüentemente, a etapa 5, não possuem equivalência nas demais propostas.

5.8.5 Etapa 5

Finalmente, a última etapa da metodologia visa implementar o modelo em um BDOO, através da nomeação e geração de classes, atributos e relacionamentos. Portanto, a validação e adaptação dos padrões da ODMG para o modelo multidimensional.

Os modelos de Kimball (1997, 2002) e Moody e Kortink (2000) abordam o modelo relacional como implementação do modelo multidimensional. O modelo de Cavero et al. (2002, 2003) considera os modelos relacionais e multidimensionais para uma possível implementação. As propostas de Trujillo et al. (2000, 2003) e Abelló et al. (2002, 2005) abordam o paradigma da orientação a objetos, porém, sem analisar sua implementação.

As etapas 4 e 5 da proposta mapeiam o modelo conceitual para sua implementação, analisando os aspectos da OO e da mutidimensionalidade. Estas etapas oferecem um roteiro de equivalências dos conceitos da OO e sua real implementação, observando sua aplicação no modelo multidimensional.

Esta etapa busca verificar a viabilidade da utilização do modelo de BDOO, considerando que todo o processo da modelagem multidimensional será definido e implementado segundo o paradigma da orientação a objetos, não necessitando do processo de reengenharia para sua aplicação em outros modelos de BD.

5.9 Resumo da comparação das propostas

Considerando que os trabalhos de Kimball (1997, 2002), Moody e Kortink (2000) e Cavero et al. (2002, 2003) abordam o modelo multidimensional com base no modelo relacional, a metodologia apresentada aproxima-se destas propostas sob o aspecto da geração do modelo multidimensional a partir do modelo ER. Este procedimento justifica-se pelo fato das empresas apresentarem seus modelos de negócio implementados no ambiente operacional através deste modelo, pois a modelagem multidimensional visa atender consultas de dados históricos, efetivamente advindos do meio operacional.

Analisando a proposta com os trabalhos que abordam o paradigma da orientação a objetos, observa-se que o trabalho de Trujillo et al. (2000, 2003) gera o modelo conceitual sem a preocupação da persistência de dados deste modelo. Enquanto a metodologia de Abelló et al. (2002, 2005) representa os conceitos do modelo formalmente e apresenta aplicações, baseadas no modelo de Kimball (1997) para a representação do modelo, porém, também não apresenta modelo de persistência de dados.

Portanto, a proposta apresentada utiliza o paradigma e conceitos da orientação a objetos para agrupar todos os conceitos da modelagem multidimensional. O modelo

de DW é estático, portanto, o modelo é representado através dos diagramas de classes e estrutura composta da UML, diferenciando-se das demais propostas pela contemplação de aplicações conceituais da OO. Complementando seu diferencial, a metodologia aborda o mapeamento do modelo para o padrão OO, resultando na persistência de dados. Os demais trabalhos analisados voltam-se para um referencial do modelo relacional, sem a preocupação de sua implementação física, ou a geração de um modelo intermediário para sua implementação em um modelo relacional ou multidimensional.

O quadro 12 mostra um resumo das características dos modelos analisados e da proposta apresentada. O quadro 13 relaciona a equivalência dos parâmetros entre os modelos analisados e a metodologia proposta.

5.10 Considerações finais

No presente capítulo foi definida a metodologia para a geração do modelo multidimensional orientado a objetos e a persistência de dados em banco de dados orientado a objetos representado através da UML. As seções anteriormente descritas mostraram cada uma das etapas abordadas pela metodologia, desde a concepção do modelo multidimensional, a partir do modelo operacional, até a geração do *script* para a implementação, seguindo o padrão ODMG.

Na análise comparativa da proposta com outros estudos foram destacados seus diferenciais e sua aplicabilidade. Observando que os trabalhos anteriores apenas representam a orientação a objetos, mas não a persistência de dados, não implementando a proposta e não abordando a questão da necessidade do processo de reengenharia para a implementação em BDR.

Complementando a pesquisa, o próximo capítulo aplica a metodologia, através de sua implementação prática, seguindo cada uma das etapas e verificando as equivalências do paradigma orientado a objetos.

Etapa	Kimball	Moody e Kortink	Cavero	Trujillo	Abelló	Proposta
1	Definição do negócio a ser modelado a partir do modelo ER.	Não formaliza a definição do negócio a ser modelado.	Definição do negócio a ser modelado como esquema inicial.	Não considera o modelo ER e os requisitos do usuário.	Não considera o modelo ER e os requisitos do usuário.	Definição do negócio a ser modelado a partir do modelo ER.
2	Define a granularidade, fatos e dimensões. Aditividade e medidas de derivação nos fatos. Hierarquias são dimensões desnormalizadas. Relacionamentos do modelo ER.	Não analisa a granularidade. Define dimensões e fatos. Aditividade e medidas de derivação nos fatos. Hierarquias são seqüências de relacionamentos 1:n do modelo ER.	Define dimensões e hierarquias de agregação. Aditividade e medidas de derivação nos fatos. Fatos são definidos pela associação das dimensões e hierarquias.	Define granularidade. Fatos e dimensões são classes. Relacionamento de agregação entre Fatos e dimensões. Hierarquias são atributos nas dimensões.	Define a granularidade. Fatos e dimensões são classes. Aditividade e medidas de derivação nos fatos. Relacionamentos de classificação, generalização e agregação.	Define a granularidade, fatos e dimensões Aditividade e medidas de derivação nos fatos. Identifica as hierarquias de classificação e os relacionamentos.
3	Não contempla o paradigma OO.	Não contempla o paradigma OO.	Não contempla o paradigma OO.	Aborda a OO. Representa Dimensão e Fatos como classes e atributos como objetos, através do diagrama de classes.	Aborda a OO. Define Fatos e Dimensão como classes. Define Descritores como atributos para selecionar instâncias.	Aborda a OO. Representa Dimensão e Fatos como classes e atributos como objetos, através do diagrama de classes. Define identificadores e descritores. Valida o esquema original.
4	Não contempla o paradigma OO.	Não contempla o paradigma OO.	Não contempla o paradigma OO.	Aborda a OO conceitualmente, não implementando a proposta.	Aborda a OO conceitualmente, não implementando a proposta.	Gera *scripts* do modelo multidimensional OO, pelo padrão ODL.
5	Não contempla a implementação do modelo. Considera o modelo relacional.	Não contempla a implementação do modelo. Considera o modelo relacional.	Não contempla a implementação do modelo. Considera o modelo relacional e multidimensional.	Não contempla a implementação do modelo.	Não contempla a implementação do modelo.	Implementa o modelo em BDOO, valida e adapta os padrões da ODMG para o modelo.

Quadro 12 - Resumo das características dos modelos analisados e da proposta apresentada.

Parâmetro	Kimball	Moody e Kortink	Cavero	Trujillo	Abelló	Proposta
Definição do negócio a ser modelado a partir do modelo ER	✓		✓			✓
Granularidade	✓		✓	✓	✓	✓
Fatos e Dimensões	✓	✓	✓	✓	✓	✓
Aditividade e medidas de derivação	✓	✓	✓	✓	✓	✓
Hierarquias de classificação	✓	✓	✓	✓	✓	✓
Relacionamentos	✓	✓	✓	✓	✓	✓
OO e UML				✓	✓	✓
Classes e objetos				✓	✓	✓
Diagrama de classes				✓	✓	✓
Diagrama de estrutura composta						✓
Descritores					✓	✓
Identificadores						✓
Validação do modelo gerado com o esquema original						✓
Geração de *scripts* para implementação						✓
Implementação do modelo OO em BDOO						✓
Validação e adaptação de padrões da OMG						✓

Quadro 13 - Existência dos parâmetros entre os modelos analisados e a metodologia proposta.

6 ESTUDO DE CASO DA METODOLOGIA PROPOSTA

6.1 Introdução

Considerando as etapas definidas no capítulo anterior, como proposta metodológica para implementar o modelo multidimensional no paradigma orientado a objetos, este capítulo visa aplicar esta metodologia, observando as fases e etapas definidas previamente.

O presente capítulo apresenta as seguintes seções, correspondendo às seções do capitulo anterior: a seção 2 apresenta a definição do modelo de negócio exemplo para a aplicação; na seção 3 é gerado o modelo multidimensional, que é representado através da UML na seção 4. A seção 5 mapeia o modelo proposto utilizando a ODL e, finalmente o modelo é implementado na seção 6.

6.2 Etapa 1: definir o modelo de negócio

Considerando o objetivo de aplicar a metodologia proposta, inicia-se sua geração através da definição do modelo de negócios a ser modelado. Nesta primeira etapa da metodologia, gera-se uma documentação relacionando a finalidade do projeto, o limite do escopo do negócio a ser modelado e as ferramentas e aplicativos a serem utilizados. O modelo de negócio para aplicar a proposta baseia-se no modelo adaptado de serviços financeiros, proposto como exemplo por Kimball (2002, p. 227). A figura 28 mostra a definição da etapa 1 da aplicação da metodologia.

Considerando o processo decisório do negócio a ser modelado, propõe-se gerar um modelo baseado e adaptado do exemplo de Kimball (2002, p. 227) de serviços financeiros, modelo de negócios amplamente utilizado em sistemas empresariais. O modelo de negócios, representado no ambiente operacional pelo modelo ER é apresentado na figura 29.

O objetivo do modelo proposto é a análise de contas bancárias. A adaptação do modelo é centrada nos seguintes diferenciais:

- a posição da contas é diária, ao contrário do modelo de Kimball (2002), que é mensal;
- em conseqüência da posição diária, a unidade Tempo também é registrada em sua menor granularidade (dia, mês e ano);
- a dimensão Gerente foi adicionada ao modelo;

- o modelo considera que cada conta esteja relacionada ao menos a um produto;
- as unidades relacionadas a localização (Cidade, UF e Região) foram incluídas no modelo.

Implantação de modelos multidimensionais mapeados em BDOO

Etapa 1

- Finalidade do projeto: aplicar a metodologia proposta de implantação de modelos multidimensionais mapeados em BDOO.
- Limite do escopo do negócio: apresentar um modelo exemplo típico de aplicações organizacionais – serviços financeiros.
- Ferramentas e aplicativos:
  - Representação do modelo ER: Microsoft Visio Professional 2002;
  - Representação do modelo multidimensional: Microsoft Visio Professional 2002;
  - Representação do modelo multidimensional OO: Enterprise Architect 6.0;
  - Padrão OO: ODMG 3.0;
  - Banco de Dados: Caché 5.0.

Figura 28 - Etapa 1 da metodologia proposta.

6.3 Etapa 2: gerar o modelo multidimensional

Para a geração do modelo multidimensional, a proposta baseia-se no padrão do ambiente de DW, que é gerado a partir do meio operacional. Definido o modelo de negócio – Serviços Financeiros, são estabelecidos os parâmetros de granularidade e, conseqüentemente, as dimensões e fatos, inicialmente em um formato geral, para um posterior detalhamento de seus atributos e medidas de derivação.

Observando o modelo de negócio, os esquemas operacionais são simples e de baixa complexidade, indicando claramente os Fatos do modelo multidimensional das transações e posição da conta bancária. São analisadas as hierarquias e, conseqüentemente, a necessidade de se anexar seus dados às tabelas Fatos ou Dimensão correspondentes.

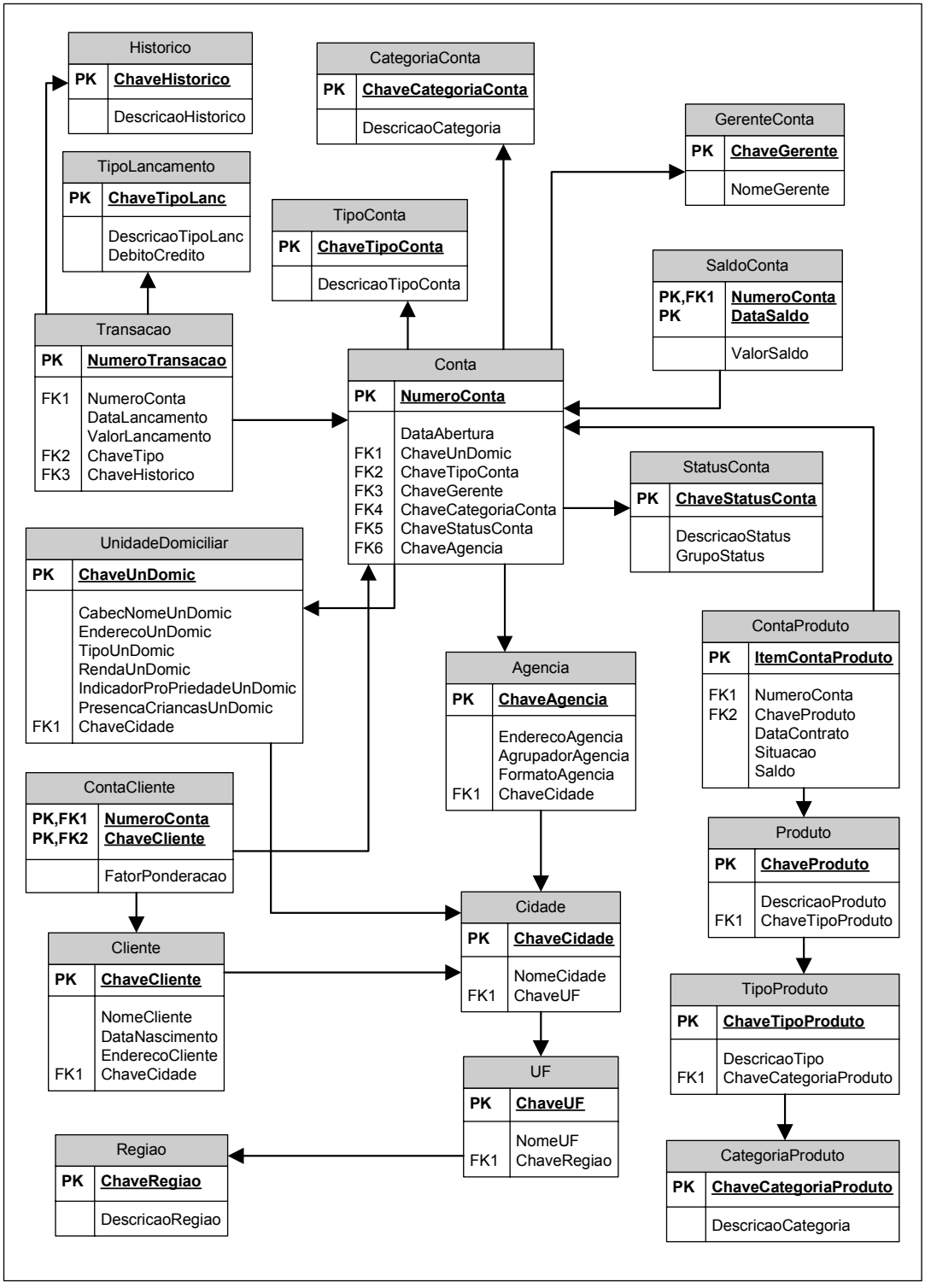

Figura 29 - Modelo Entidade-Relacionamento de um sistema de Serviços Financeiros.

6.3.1 Granularidade

Como regra geral, os dados podem ser mantidos com o maior nível de detalhamento e posteriormente, sumarizados, gerando um nível mais baixo de granularidade, oferecendo flexibilidade às consultas do usuário.

Na geração do modelo multidimensional deve ser considerada a questão da aditividade. Partindo do modelo ER, o modelo multidimensional pode ser concebido:

a) com alta granularidade, gerando uma tabela Fatos apenas com dados aditivos para os atributos de análise, os dados atômicos são mantidos em uma tabela dependente;
b) com baixa granularidade, gerando uma tabela Fatos apenas com dados atômicos.

As duas concepções apresentam suas vantagens e desvantagens, como se observa no quadro 14.

Granularidade	Aditividade	Vantagens	Desvantagens
Alta	Tabela Fatos com dados aditivos	Consultas sumarizadas mais rápidas	Busca de dados atômicos em tabela dependente
Baixa	Tabela Fatos com dados atômicos, sem aditividade	Atendimento de praticamente todas as consultas	Consultas sumarizadas mais lentas

Quadro 14 - Vantagens e desvantagens na proposição do nível de granularidade.

O modelo proposto permite a geração do mais baixo nível de granularidade, considerando que apresenta o maior nível de detalhamento, derivado do ambiente operacional. Porém, será utilizada a aditividade para sumarizar os dados na tabela Fatos, visando atender os objetivos do modelo, que implica em uma análise sob a perspectiva da agência, gerente, conta e período.

6.3.2 Dimensões e Fatos

No modelo de Serviços Financeiros, toda medida descreve exatamente o mesmo grupo de dimensões, gerando, portanto, o fato PosicaoConta. Esta tabela de fatos pode gerar o mais alto ou baixo nível de granularidade, porém, com um único tipo, não caracterizando granularidade mista.

Observa-se então, a partir do modelo ER de Serviços Financeiros proposto, que a entidade de transação é definida, ou seja, a tabela Fatos, com base na posição

financeira diária das contas de clientes, é identificada como a questão de negócio a ser analisada.

As entidades componentes são representadas pelas entidades que possuem um relacionamento um-para-muitos com a entidade de transação, que no modelo são geradas a partir das entidades Agencia, StatusConta, UnidadeDomiciliar e Gerente. A entidade de classificação do modelo também caracteriza um relacionamento um-para-muitos. Segundo o modelo proposto, estes relacionamentos são identificados entre as entidades Conta, Saldo e Transação. A tabela de fatos é gerada a partir do relacionamento destas entidades operacionais, que possuem os principais atributos para a identificação do negócio a ser modelado. Considerada a questão central a ser analisada, a entidade de classificação é gerada a partir das entidades Transação e Saldo, enquanto a entidade Conta gera uma entidade componente.

Em uma complementação das entidades componentes e de classificação, são definidos os atributos de cada uma destas dimensões. Os atributos operacionais que não são importantes para a análise do negócio podem ser descartados.

6.3.2.1 Dimensão Tempo

Lembrando a importância do fator tempo para o modelo multidimensional, anteriormente justificado, esta dimensão é gerada a partir dos dados da entidade Transação e Saldo, considerando o tempo como o período em que ocorreu determinada transação, no exemplo, em determinada data. No modelo proposto, portanto, não se faz necessário levantar tais informações, pois estão definidas no próprio ambiente operacional. A dimensão tempo será armazenada em dias (data da ocorrência) considerando o mais baixo nível de granularidade para o atributo de data. Dados adicionais podem ser incluídos, como dia da semana, trimestre, estação do ano, etc., de acordo com os requisitos definidos pelo usuário das consultas gerenciais.

6.3.2.2 Medidas de derivação

Para iniciar o processo de refinamento e detalhamento do modelo são analisados os atributos existentes. Podem ser detectados atributos novos e atributos desnecessários podem ser eliminados.

O modelo segue as proposições da seção 5.3, visando evitar o processo de sumarização em cada consulta, mantendo na tabela Fatos os dados sumarizados, segundo o objetivo do modelo de negócio.

O quadro 15 mostra as medidas derivadas: SaldoMedioMensal, NumeroTransacoes, JurosPagos, JurosCobrados, TaxasCobradas, TotalDebito e TotalCredito, e suas regras de derivação, geradas a partir das tabelas de dimensão e implementadas na tabela Fatos. As medidas derivadas referenciadas no exemplo são sumarizadas através da utilização dos operadores *SUM, AVG* e *COUNT* na agregação dos valores de medidas de dimensões.

Atributo(s) origem	**Regra de derivação**	**Medida derivada**
SaldoConta.ValorSaldo	AVG(ValorSaldo) no mês	SaldoMedioMensal
Transação.NumeroTransacao	COUNT(NumeroTransacao) na data	NumeroTransacoes
Transação.ValorLancamento	SUM(ValorLancamento) para tipo = juros pagos	JurosPagos
Transação.ValorLancamento	SUM(ValorLancamento) para tipo = juros cobrados	JurosCobrados
Transação.ValorLancamento	SUM(ValorLancamento) para tipo = taxas cobradas	TaxasCobradas
Transação.ValorLancamento	SUM(ValorLancamento) para tipodebitocredito = debito	TotalDebito
Transação.ValorLancamento	SUM(ValorLancamento) para tipodebitocredito = credito	TotalCredito

Quadro 15 - Regras de derivação e medidas derivadas para o modelo de Serviços Financeiros.

A figura 30 mostra a tabela Fatos PosicaoConta gerada com os atributos sumarizados: SaldoMedioMensal, NumeroTransacoes, JurosPagos, JurosCobrados, TaxasCobradas, TotalDebito e TotalCredito que correspondem, respectivamente ao resumo dos valores para saldo mensal, número de transações, juros, taxas, débitos e créditos da conta em determinada data.

6.3.3 Relacionamentos

A cardinalidade dos relacionamentos no modelo operacional proposto são da ordem n:n no relacionamento entre Conta-Cliente e Conta-Produto e 1:n para os demais relacionamentos.

Portanto, o problema abrange uma situação que permite a mudança do relacionamento de acordo com a granularidade sugerida e adotada para o modelo. Observando a necessidade de se manter os dados referenciais dos clientes de cada

conta, uma tabela *bridge* é gerada entre as tabelas Dimensão Conta e Cliente. O mesmo procedimento é adotado para o relacionamento entre Conta e Produto.

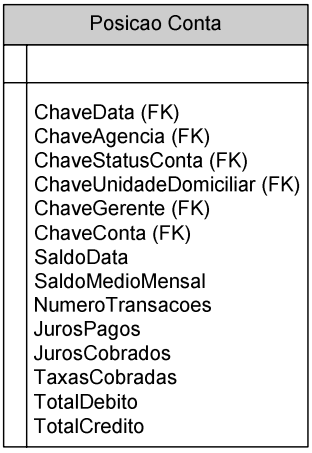

Figura 30 -Tabela Fatos Posição Conta com atributos de agregação (sumarizados).

6.3.3.1 Hierarquias de classificação

Considerando que para uma Dimensão, uma entidade básica representa todos os níveis de hierarquia de classificação, as dimensões StatusConta e Tempo encontram-se devidamente definidas.

As dimensões Cliente, Agencia e UnidadeDomiciliar utilizam a hierarquia de classificação da entidade Cidade e definem individualmente seu DAG. A estrutura do DAG representa um caminho alternativo e múltiplas hierarquias de classificação. Assim, a entidade Cidade é uma associação com as dimensões Cliente, Agencia e UnidadeDomiciliar. A figura 31 mostra uma hierarquia de classificação especificada para estas dimensões, que compartilham um caminho de hierarquia de classificação.

O modelo proposto apresenta hierarquias explicitamente embutidas, mostradas na figura 32. A figura 32a mostra a hierarquia Cidade-UF-Regiao como sendo o enquadramento da cada cidade em sua respectiva região e a figura 32b registra a hierarquia entre produto-tipo-categoria.

A figura 33 apresenta o modelo com as respectivas entidades desnormalizadas. Na figura 33a a entidade Conta absorveu os atributos com informações relativas ao tipo de conta e sua respectiva categoria. A figura 33b mostra a entidade Cidade “desmontada” na entidade Agencia. A figura 33c mostra as entidades Tipo e Categoria, relacionadas à entidade Produto.

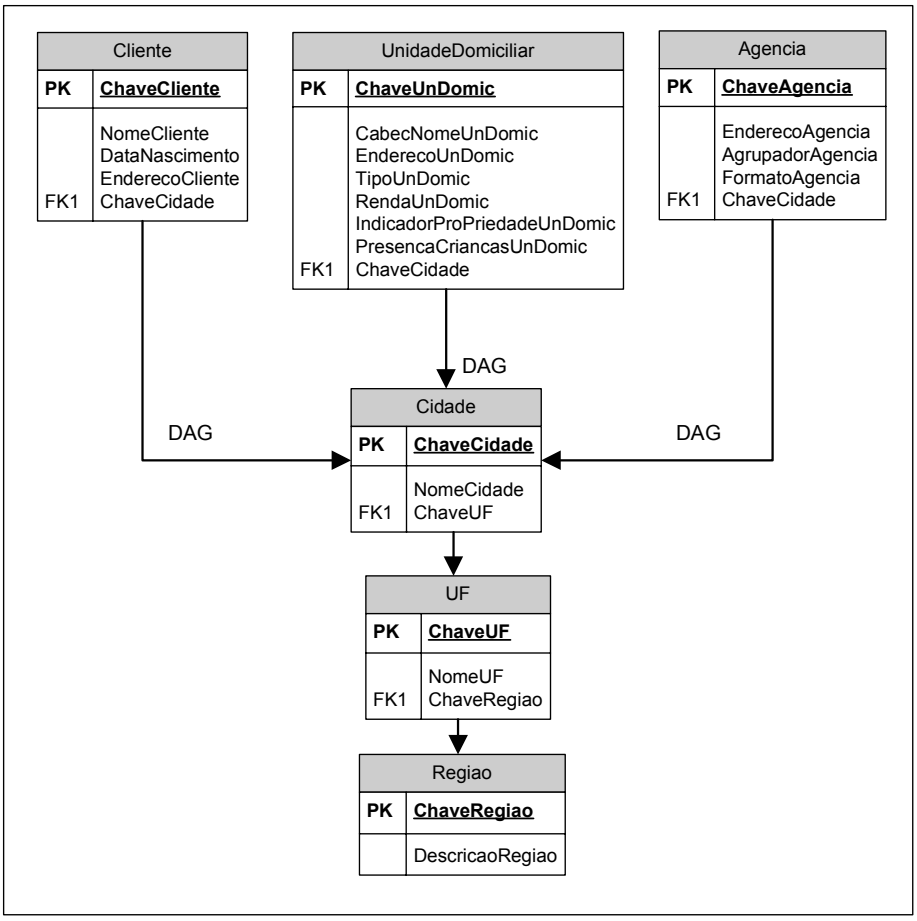

Figura 31 - Hierarquias de classificação.

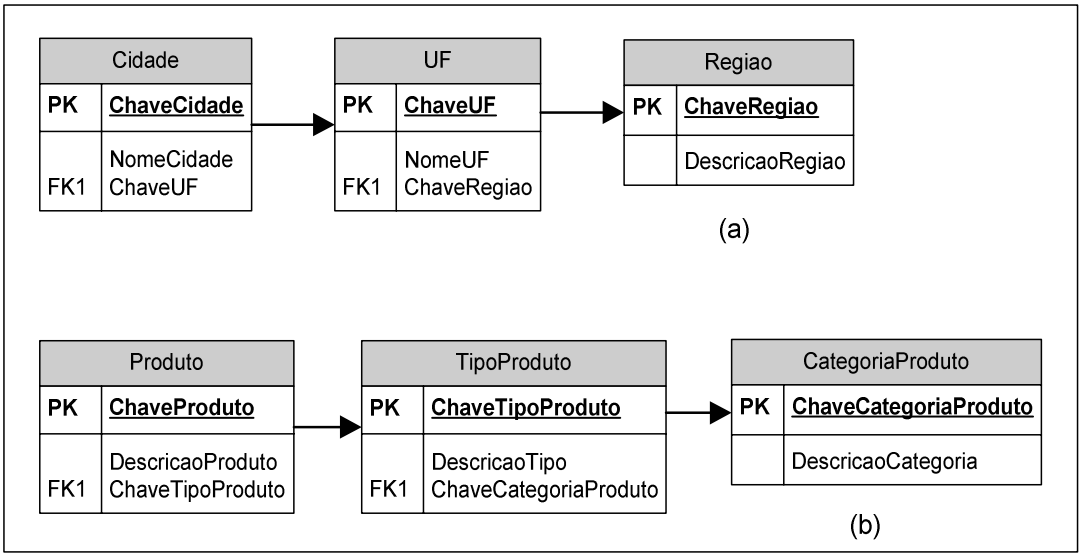

Figura 32 - Hierarquias nas entidades Cidade e Produto.

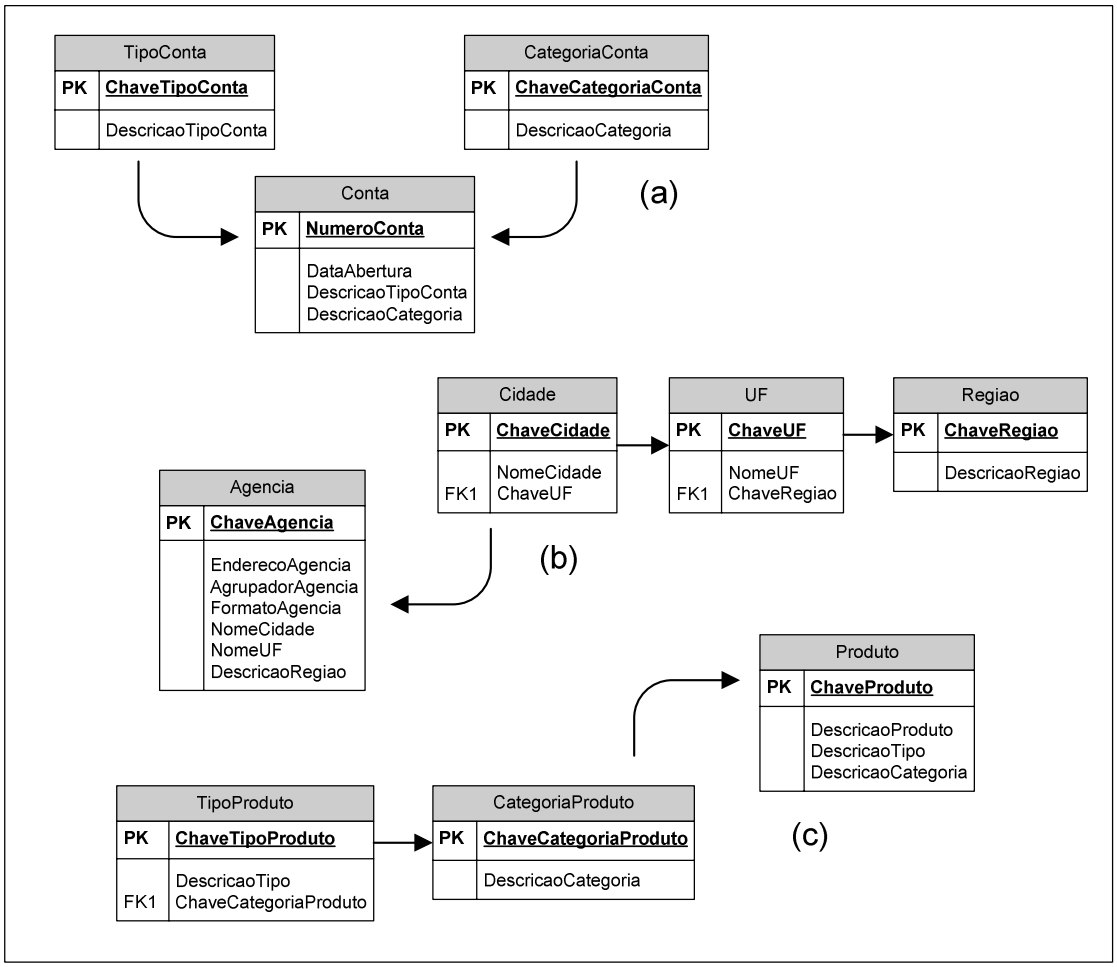

Figura 33 - Entidades do modelo ER desnormalizadas para gerar o modelo multidimensional.

6.3.3.2 Exatidão e completeza

O modelo registra as características de exatidão e completeza na hierarquia representada na figura 31, em que uma Região só se relaciona a uma UF. Uma Região é formada por todas as UFs existentes e todas as UFs que formam uma Região são armazenadas.

Aplicando a metodologia proposta, a etapa 2 é concluída com as definições abaixo relacionadas:

- estabelecida a granularidade do modelo, identificando a necessidade do nível sumarizado;
- gerada a tabela Fatos, a partir do escopo do problema de Transações Financeiras;

- identificadas as dimensões não associativas, que estão ligadas aos fatos, com respectivos atributos, que são representadas pelas dimensões Agencia, StatusConta, UnidadeDomiciliar e Gerente;
- identificada a dimensão associativa, e respectivos atributos multidimensionais, representada pela dimensão Conta;
- gerada a dimensão Tempo, a partir do período da transação da operação, com sua respectiva granularidade;
- determinados os atributos das entidades que serão utilizados nas várias dimensões geradas, considerando o negócio em análise;
- definidas as medidas e regras de derivação, relacionadas aos valores e números de transações;
- verificadas as hierarquias de classificação, definindo se estas serão agregadas às dimensões associativas ou estabelecer os DAGs, gerando o modelo Floco de Neve;
- verificada a existência de valores nulos, que não atendam aos requisitos de exatidão e completeza estabelecidos para o modelo;
- gerada a tabela *bridge* para solucionar a necessidade de se manter o relacionamento n:n de associação entre Conta-Cliente e de composição entre Conta-Produto;
- comparados e revisados os relacionamentos gerados no modelo multidimensional com o modelo operacional original;
- verificada a necessidade da utilização dos atributos para se preservar a análise do negócio.

Portanto, o modelo multidimensional proposto, gerado a partir do modelo operacional ER de Pedidos de Clientes é representado na figura 34.

Concluindo a segunda etapa da metodologia proposta, a figura 35 apresenta o resumo das tarefas desenvolvidas, mostrando os procedimentos realizados ao término desta etapa. O modelo multidimensional é proposto e convalidado com o modelo original operacional.

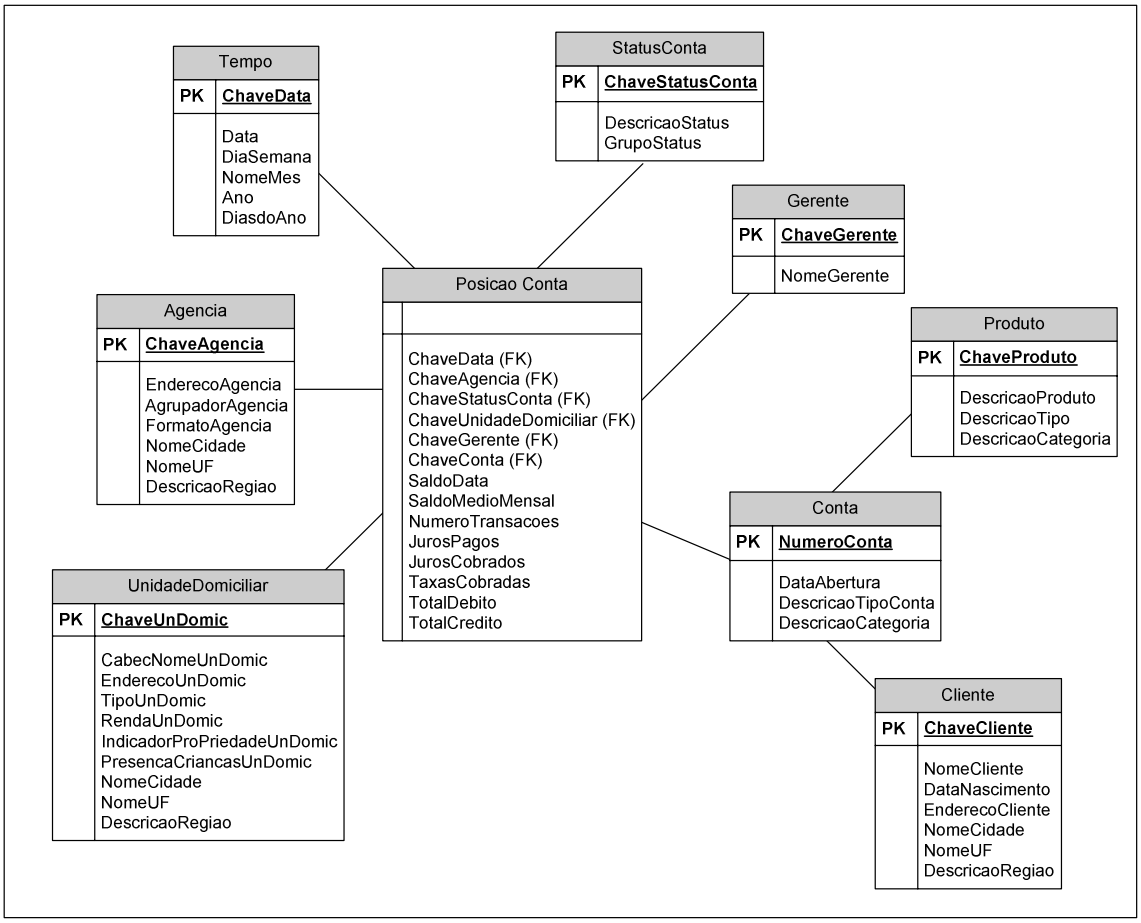

Figura 34 - Modelo multidimensional de Posição de Conta gerado a partir do modelo ER.

Implantação de modelos multidimensionais mapeados em BDOO

Etapa 2

- Granularidade: alta granularidade com a sumarização dos dados;
- Tabela Fatos: PosicaoConta, a partir das entidades operacionais Conta, Saldo e Transação;
- Tabelas Dimensão: Agencia, StatusConta, UnidadeDomiciliar, Gerente, Conta e Tempo;
- Modelo: Floco de Neve com a necessidade das tabelas Cliente e Produto;
- Dimensão Tempo: gerada a partir da data do lançamento operacional (armazenada em dias);
- Medidas derivadas: SaldoMedioMensal, NumeroTransacoes, JurosPagos, JurosCobrados, TaxasCobradas, TotalDebito e TotalCredito, com as

respectivas regras de derivação;

- Relacionamentos entre Fatos e Dimensões: 1:n;
- Relacionamentos entre as dimensões Conta-Cliente e Conta-Produto: n:n;
- DAGs: são desmontadas as hierarquias de classificação da entidade Cidade nas dimensões Agencia, Gerente, Cliente e UnidadeDomiciliar. As entidades TipoProduto e CategoriaProduto incluíram suas características em Produto, assim como TipoConta e CategoriaConta em Conta.
- Exatidão e completeza: não existem valores nulos;
- Tabelas *bridge*: necessárias para atender o relacionamento n:n de associação entre Conta-Cliente e de composição entre Conta-Produto;
- Comparação com modelo operacional: não são necessários novos atributos (além daqueles gerados na sumarização); os atributos das tabelas de Histórico e TipoLancamento são descartados, considerando que são descritores da tabela Transacao e que seus atributos foram sumarizados; os relacionamentos do modelo multidimensional atendem às regras de negócio do modelo operacional.

Figura 35 - Etapa 2 da metodologia proposta.

6.4 Etapa 3: gerar o modelo multidimensional representado pelo diagrama de classes

Como resultado final da etapa anterior é gerado o modelo multidimensional, representado através do esquema Floco de Neve. Na etapa 3 o modelo multidimensional é mapeado para o diagrama de classes, representando o modelo estático multidimensional segundo o paradigma OO. Após o mapeamento das classes, atributos e relacionamentos, são determinados identificadores e descritores, para posterior validação com o modelo operacional ER. Os relacionamentos de associação composição são representados através do diagrama de estrutura composta.

6.4.1 Classes

Observando a metodologia proposta, o modelo multidimensional OO é gerado a partir do modelo multidimensional projetado, em que as dimensões e fatos são

representados por classes Dimensão e Fatos. A figura 36 representa genericamente as classes dimensão Tempo, Agencia, StatusConta, UnidadeDomiciliar, Conta, Cliente, Produto e Gerente, além da classe Fatos PosicaoConta, com um esboço inicial do modelo a ser gerado.

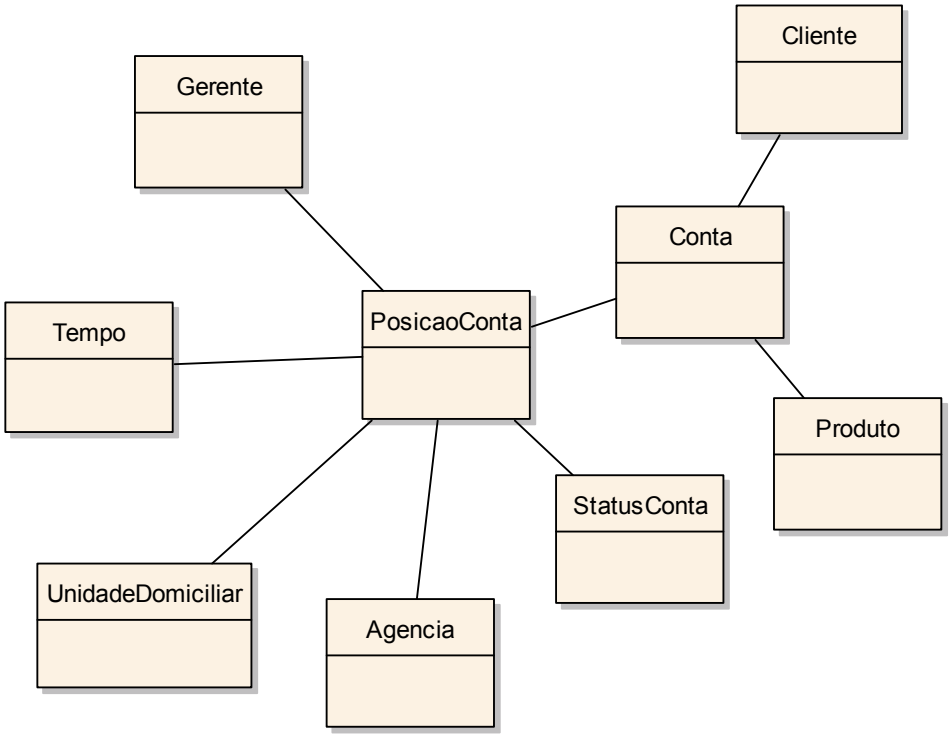

Figura 36 - Diagrama de classes genérico do modelo multidimensional.

6.4.2 Relacionamentos

A figura 37 mostra o relacionamento entre a classe Fatos PosicaoConta como uma agregação compartilhada entre as classes dimensão. Na figura 38, o relacionamento de associação composição entre Conta e Produto é representado através do diagrama de estrutura composta, registrando que uma conta é sempre constituída por pelo menos um produto, ou seja, quando a conta é registrada, registram-se os produtos a ela associados.

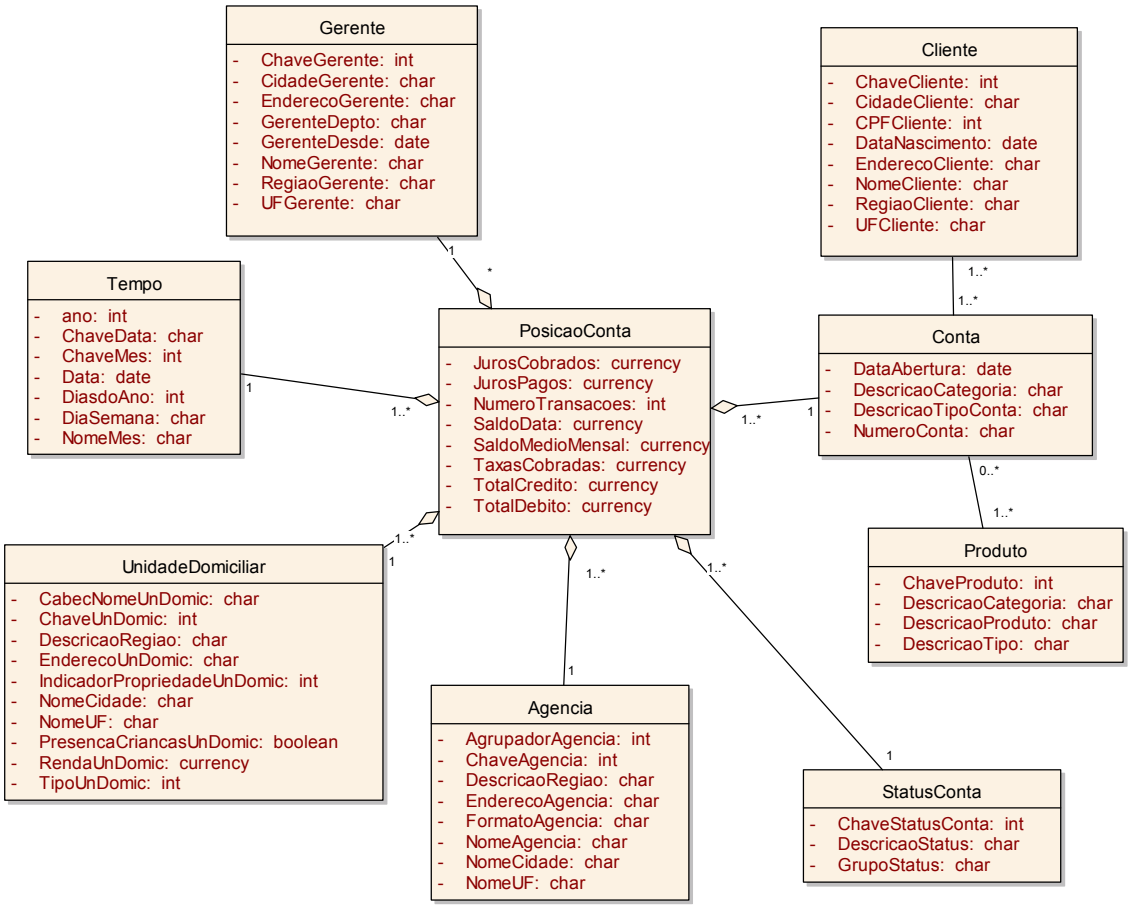

Figura 37 - Relacionamento de agregação compartilhada entre as Classes Fatos e Dimensão.

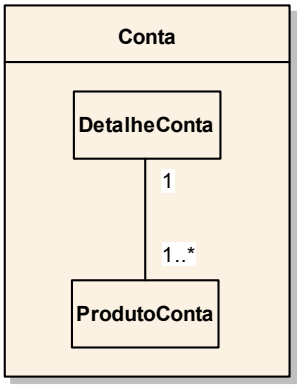

Figura 38 - Diagrama de estrutura composta de Conta.

6.4.3 Identificadores e Descritores

As classes básicas, inclusive as classes dimensão, que pertencem à hierarquia de classificação, devem conter um atributo identificador explicitamente definido. Para

isto, coloca-se a restrição {OID} próximo a um atributo em cada classe, conforme mostra a figura 39, para se representar seu identificador.

Na representação do modelo, os descritores são identificados {D} de acordo com os atributos representativos de cada classe como o atributo de referência para as consultas do usuário.

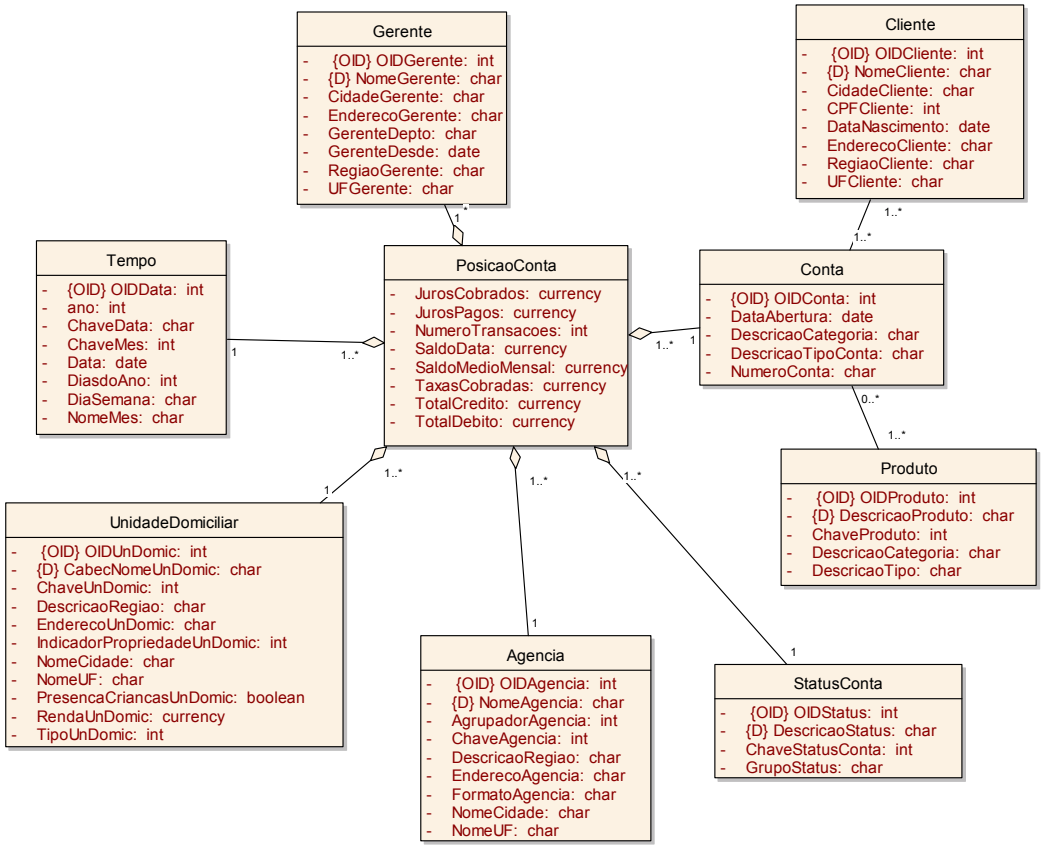

Figura 39 - Diagrama de Classes com descritores e identificadores.

6.4.4 Refinar o modelo

A partir das considerações acima referenciadas sobre as características conceituais da UML, o modelo proposto agrega tais funcionalidades, refinando e concluindo o modelo exemplo elaborado.

O diagrama representa a especialização da classe Pessoa para as classes dimensão Cliente e Gerente, sob a análise dos atributos diferenciados para as classes especializadas, conforme representa a figura 40.

A dimensão Tempo engloba os atributos dos relacionamentos de exatidão e completeza das classes Mês e Ano, incorporando os atributos de referência de uma

data. Finalmente, as hierarquias de classificação de Cidade-Uf-Regiao têm seus atributos agregados à classe Pessoa.

Aplicando a metodologia proposta, a etapa 3 é concluída com as definições abaixo relacionadas:

- representação das dimensões e fatos através das classes Dimensão e Fatos;
- representação do relacionamento de agregação compartilhada entre a classe Fatos e as classes Dimensão;
- verificação da necessidade da utilização do diagrama de estrutura composta para a representação de relacionamentos de composição entre dimensões;
- definição dos atributos identificadores das classes Dimensão e Fatos;
- definição dos atributos descritores nas classes Dimensão;
- representação através da especialização/ generalização das classes que possuem características semelhantes;
- refinamento do modelo através da análise dos conceitos da OO que podem ser aplicados no modelo utilizado.

O diagrama de classes do modelo multidimensional PosicaoConta é, portanto, representado na figura 40.

Concluindo a terceira etapa da metodologia, a figura 41 apresenta o resumo das tarefas desenvolvidas nesta seção. O modelo multidimensional definido na etapa 2 é mapeado para a orientação a objetos e representado através dos diagramas de classes e de estrutura composta da UML.

6.5 Etapa 4: mapear o diagrama de classes para BDOO

Nesta etapa são gerados os modelos de *script* para o mapeamento do diagrama representado pela UML para BDOO. Os *scripts* são gerados segundo as referências da ODL.

Considerando a proposta do mapeamento do modelo de PosicaoConta, os primeiros elementos a serem mapeados são as classes Fatos e Dimensões.

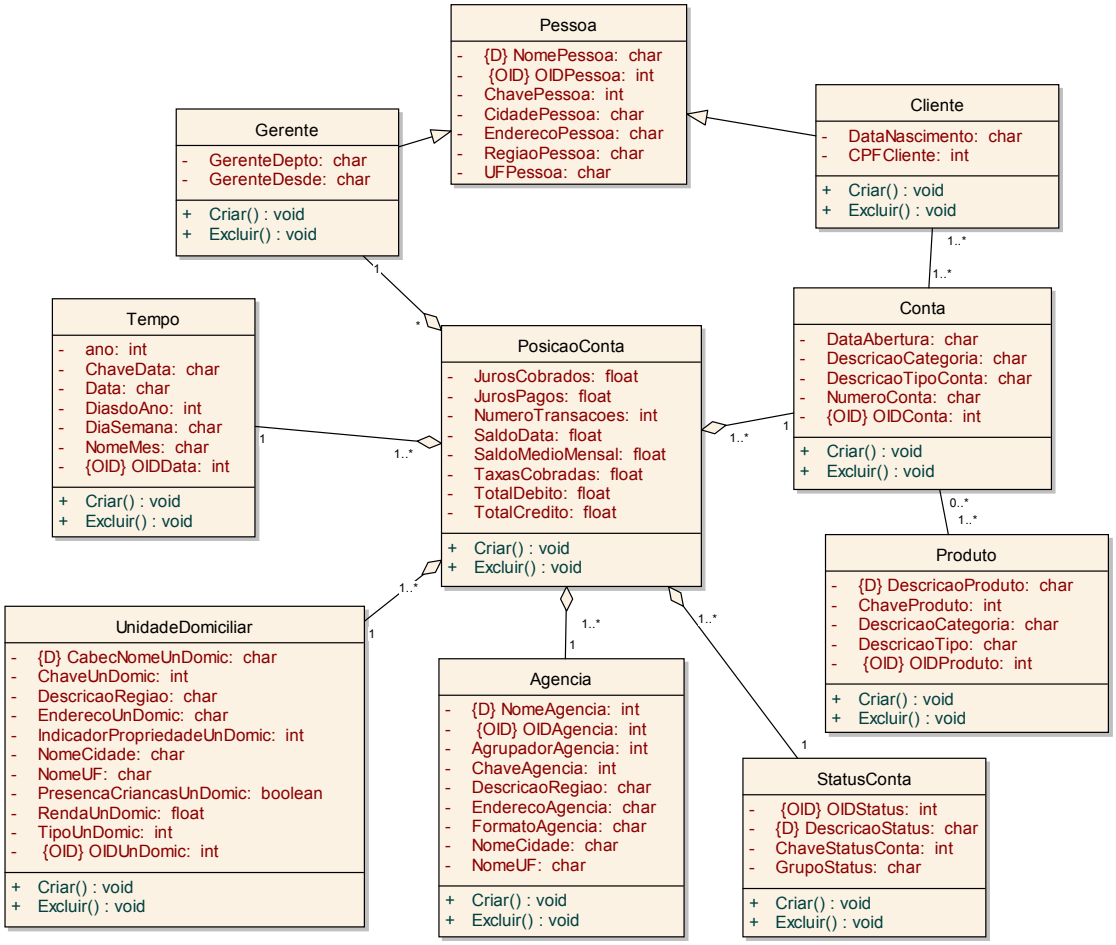

Figura 40 - Modelo multidimensional PosicaoConta com a especialização da superclasse Pessoa nas classes dimensão Cliente e Gerente.

Implantação de modelos multidimensionais mapeados em BDOO

Etapa 3

- Classes:
  - mapeamento das tabelas dimensão Tempo, Agencia, StatusConta, UnidadeDomiciliar, Conta e Gerente em classes Dimensão;
  - mapeamento da tabela Fatos PosicaoConta como uma classe Fatos;
- Relacionamentos:
  - representação dos relacionamentos entre as classes dimensão Tempo, Agencia, StatusConta, UnidadeDomiciliar, Conta e Gerente e a classe Fatos PosicaoConta;

- representação dos relacionamentos de associação composição entre Conta-Produto através do diagrama de estrutura composta;
- definição do relacionamento de generalização para a superclasse Pessoa, que agrega as características comuns das classes Cliente e Gerente, que são as classes especializadas;
- definição do modelo Floco de Neve através da utilização das classes Cliente e Produto;

- Identificadores: representados os OIDs de cada uma das classes (Dimensão e Fatos);
- Descritores:
  - classe Pessoa → NomePessoa (cliente e gerente);
  - classe UnidadeDomiciliar → CabecNomeUnDomic;
  - classe Agencia → NomeAgencia;
  - classe StatusConta → DescricaoStatus;
  - classe Produto → NomeProduto.

Figura 41 - Etapa 3 da metodologia proposta.

6.5.1 Gerar classes

As dimensões Agencia, StatusConta, UnidadeDomiciliar, Tempo e Gerente são classificadas como não associativas e a dimensão Conta como uma associativa. Analisando a natureza da classe Fatos, cada fato é uma classe associativa, de acordo com a granularidade dos dados.

Considerando que as dimensões são representadas como classes, é possível uma especialização de dimensões, definidas por uma organização e subclasses. No modelo proposto, a superclasse Pessoa abrange características das classes Gerente e Cliente, com atributos gerais para as classes e específicos para cada uma delas separadamente. Portanto, os atributos generalizados são gerados na superclasse Pessoa, conforme o *script* em ODL da figura 42.

A partir da superclasse, definida através da geração da classe Pessoa, são geradas as classes especializadas de Gerente e Cliente, com os atributos correspondentes. Estas classes herdam as características da classe Pessoa, e

agregam suas próprias características, conforme a definição abaixo da figura 43 em ODL.

```
class Pessoa (extent Pessoas)
{
attribute string NomePessoa;
attribute string ChavePessoa;
attribute string CidadePessoa;
attribute string EnderecoPessoa;
attribute string RegiaoPessoa;
attribute string UfPessoa;
}
```

Figura 42 - *Script* da superclasse Pessoa em ODL.

```
class Cliente extends Pessoa
{
attribute unsigned long CPFCliente;
attribute date DataNascimento;
}
```

```
class Gerente extends Pessoa
{
attribute string GerenteDepto;
attribute date GerenteDesde;
}
```

Figura 43 - *Script* das classes especializadas de Gerente e Cliente.

6.5.2 Gerar identificadores

Os identificadores são gerados na implementação e não representados pela ODL.

6.5.3 Gerar descritores

As propriedades de descritor do objeto {D} são representadas no diagrama de classes, mas ignorados no mapeamento.

6.5.4 Gerar relacionamentos

No relacionamento entre as classes Gerente-Cidade e Cliente-Cidade, os atributos representando a cidade estão em cada uma das classes - Gerente e Cliente - através de um tipo de objeto denominado estrutura, que é composto de outros objetos, conforme a declaração no trecho ODL da figura 44. Considerando que os atributos são comuns para as classes Gerente e Cliente, estes são gerados

na superclasse Pessoa, para que as classes especializadas herdem tais características.

```
struct Cidade
{
string NomeCidade;
string UFCidade;
string RegiaoCidade;
}
```

```
class Pessoa (extent Pessoas)
{
attribute string NomePessoa;
attribute string ChavePessoa;
attribute struct Cidade CidadePessoa;
attribute string EnderecoPessoa;
}
```

Figura 44 - *Script* do tipo de objeto estrutura Cidade.

A declaração ODL *enum* pode assumir valores específicos e enumerados na classe Tempo, que tem o atributo NomeMes com os valores respectivamente relacionados para cada um dos meses do ano, conforme representa a figura 45.

```
class Tempo (extent DataPed)
{
...
enum NomeMes {JANEIRO, FEVEREIRO, MARÇO, ABRIL, MAIO, JUNHO, JULHO, AGOSTO, SETEMBRO, OUTUBRO, NOVEMBRO, DEZEMBRO};
...
}
```

Figura 45 - *Script* da declaração *enum* na classe Tempo.

Em ODL, a definição de um relacionamento origem-destino inclui a geração de um tipo destino, a cardinalidade do lado destino e informações sobre o relacionamento inverso, do lado origem.

O relacionamento de agregação não é representado diretamente em ODL, portanto, deve ser transformado em um relacionamento de associação.

No exemplo da figura 46, o tipo de coleção literal *set*<> indica que cada gerente pode estar em uma ou mais ocorrências na classe PosicaoConta, definido em ODL:

```
class Gerente extends Pessoa
{
...
Relationship set<PosicaoConta> ContaGerente inverse Gerente::posicao;
...
}
```

Figura 46 - *Script* do tipo de coleção literal *set*<>.

No modelo proposto, observa-se que cada cliente pode ter várias contas, bem como cada conta pode conter vários produtos, sem a definição de um número exato de ocorrências. Portanto, para a solução deste relacionamento, o modelo apresentado inclui uma tabela *bridge* entre as tabelas Conta-Cliente, como ilustrado na figura 47.

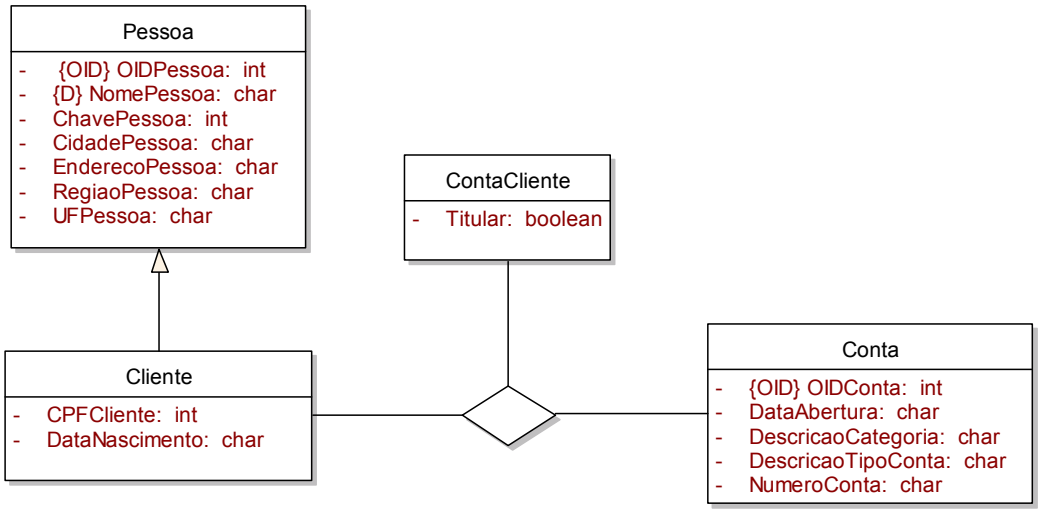

Figura 47 - Tabela bridge entre as classes Cliente e Conta.

Em ODL a geração da classe ContaCliente é representada na figura 48. Nas classes Conta e Cliente são gerados os relacionamentos correspondentes, em ODL, conforme mostra a figura 49.

Ao término da quarta etapa da metodologia, a figura 50 apresenta o resumo das tarefas desenvolvidas nesta fase, quando o modelo multidimensional representado pelos diagramas de classe e estrutura composta são mapeados para o padrão ODMG, utilizando a ODL..

```
class ContaCliente (extent CtaCli)
{
attribute boolean Titular;
relationship set<Conta> ItemCta inverse Conta::contas;
relationship set<Cliente> ItemCli inverse Cliente::clientes;
}
```

Figura 48 - *Script* da classe *bridge* ContaCliente.

```
class Conta (extent Contas)
{
attribute date DataAbertura;
attribute string DescricaoCategoria;
attribute string DescricaoTipoConta;
attribute string NumeroConta;
relationship set< ContaCliente> contas inverse ContaCliente:: ItemCta;
}

class Cliente extends Pessoa
{
attribute long CPFCliente;
attribute date DataNascimento;
relationship set<ContaCliente> clientes inverse ContaCliente::ItemCli;
}
```

Figura 49 - *Script* das classes Conta e Cliente dos relacionamentos com a classe *bridge* ContaCliente.

Implantação de modelos multidimensionais mapeados em BDOO

Etapa 4

- Classes:
  - *script* de criação das classes dimensão Tempo, Agencia, StatusConta, UnidadeDomiciliar e Conta;
  - *script* de criação da superclasse Pessoa, com as propriedades comuns de Cliente e Gerente;
  - *script* de criação das classes Cliente e Gerente, herdando as propriedades da superclasse Pessoa;
  - *script* de criação da classe PosicaoConta como uma classe Fatos;
  - *script* de criação das classes associativas ContaCliente e ContaProduto;
- Relacionamentos:
  - transposição do relacionamento de especialização / generalização para o relacionamento *ISA*, definido na superclasse Pessoa;
  - transposição do relacionamento de herança de estado e comportamento para o relacionamento *EXTENDS* para as classe Cliente e Gerente, especializadas da superclasse Pessoa;

- transformação dos relacionamentos de agregação entre as classes Dimensão e Fatos para relacionamentos de associação, definindo os relacionamentos e seus inversos;
- utilização da declaração *enum* para definir restrições de valores para os atributos das classes;
- transformação do relacionamento n:n entre as classes Conta-Cliente para relacionamento de associação, gerando a classe *bridge* ContaCliente, definindo os relacionamentos e seus inversos;
- transformação do relacionamento da estrutura composta entre as classes Conta e Produto na classe *bridge* ContaProduto, definindo os relacionamentos e seus inversos;

- Identificadores: não são representados pela ODL.
- Descritores: não são representados pela ODL.
- Atributos:
  - transformação dos atributos complexos em atributos estruturados, através da utilização da estrutura Cidade, utilizada como atributo nas classes UnidadeDomiciliar, Agencia e na superclasse Pessoa;
  - definição do tipo de dados dos atributos das classes, segundo o padrão ODMG.

Figura 50 - Resumo da etapa 4 da metodologia.

6.6 Etapa 5: Implementar o modelo

Para a aplicação da proposta de modelagem multidimensional segundo o paradigma da orientação a objetos e seu respectivo mapeamento, gerando a persistência de objetos, foi utilizado o banco de dados pós-relacional Caché – Intersystems.

A seção apresenta a utilização do banco de dados adotado para a aplicação do trabalho, apresentando suas características e funcionalidades. Para cada especificação do modelo multidimensional é apresentada a equivalência no banco de dados Caché, observando-se ainda, a padronização especificada pela ODL.

O mapeamento considera as classes do modelo multidimensional, seus relacionamentos e tipo de dados, observando as soluções propostas nas seções

anteriores. As ferramentas associadas ao banco de dados são exploradas no contexto de verificar as facilidades para a implementação do modelo definido.

O banco de dados deve fornecer as características apresentadas nos capítulos anteriores, considerando particularmente os conceitos do paradigma da orientação a objetos.

6.6.1 Banco de Dados Orientado a Objetos Caché

6.6.1.1 Introdução

O Caché pertence a uma geração de bancos de dados conhecida como "pós-relacional". Como um banco de dados pós-relacional, combina um banco de dados de objetos, SQL e acesso multidimensional aos dados - todos eles podendo acessar os mesmos dados simultaneamente. O Caché suporta SQL padrão como linguagem de consulta e de atualização usando uma tecnologia de banco de dados multidimensional e o SQL estendido para incluir capacidades de objetos (Intersystems, 2005).

O modelo de objetos do Caché é baseado no padrão ODMG. Além disso, suporta um conjunto de conceitos de programação por objeto que inclui encapsulamento, objetos embutidos, herança múltipla, polimorfismo e coleções.

O Caché armazena dados em vetores esparsos, o que torna seu servidor de dados compacto, armazenando mais dados em um mesmo espaço em disco. Esta eficiência do Caché resulta em melhor desempenho e em menores custos de hardware.

O anexo A apresenta resumos de artigos sobre desempenho realizados com o Caché e outros bancos de dados, indicando resultados favoráveis ao Caché. Complementando este anexo, o apêndice A apresenta um estudo comparativo realizado entre o Caché e o banco de dados Oracle, aplicado a um modelo multidimensional.

O apêndice A registra as características do sistema utilizado para o teste comparativo de desempenho, relacionando hardware e software. O teste emprega resumidamente o exemplo utilizado na etapa imediatamente anterior, através das classes Fatos PosicaoConta e as dimensões: Agencia, Gerente, Cliente, Conta e Tempo, além da associação Conta-Cliente.

O teste foi realizado visando o cumprimento da proposta do modelo multidimensional, que é atender consultas e solicitações do usuário, buscando informações para o processo de apoio a decisão organizacional. Neste contexto, os testes demonstraram a viabilidade do produto e, conseqüentemente, do paradigma da orientação a objetos e suas caracterisiticas.

6.6.1.2 Relacionamentos no Caché

Para o banco de dados Caché, um relacionamento é uma associação entre dois objetos, de um tipo específico. Para criar um relacionamento entre dois objetos, cada um deve ter uma propriedade *Relationship*, que define sua parte do relacionamento, seguindo a padronização ODMG.

Os relacionamentos no Caché têm as seguintes características:

- são binários, em que o relacionamento é definido entre duas, e apenas duas classes, ou com uma classe e ela mesma;
- são definidos apenas para classes persistentes;
- devem ser bidirecionais, ou seja, ambos os lados do relacionamento devem ser definidos;
- administram automaticamente seu comportamento na memória e no disco;
- provêem escalamento superior e concorrência sobre coleções de objeto;
- são visíveis para o SQL como chaves estrangeiras.

Na versão 5.0, o Caché suporta dois tipos de relacionamento: 1:n (independente) e pai-filho (dependente). Relacionamentos 1:1 e n:n não são suportados.

Relacionamentos fornecem integridade referencial automaticamente, ao contrário de uma propriedade de referência (ou *object-valued*), o valor do relacionamento é limitado para estar correto, ou seja, não há nenhuma referência oscilando.

O anexo B apresenta um estudo mais amplo das características do banco de dados Caché. As informações completas, com referências técnicas de todos os componentes do banco de dados Caché são encontradas no site do fornecedor (www.intersystems.com.br).

6.6.2 Implementação do modelo proposto

Os *scripts* do modelo exemplo são gerados através do Caché Studio, que é a *interface* de desenvolvimento do banco de dados Caché, permitindo a criação de classes, métodos, etc. Os exemplos são alocados no pacote denominado Modelo.

6.6.2.1 Geração das Classes Dimensão

As classes dimensão são geradas com os atributos como propriedades. Observando que as classes são definidas como persistentes, para armazenamento definitivo dos dados, utilizam-se conceitos de herança, possibilitando a utilização dos métodos da classe *%Persistent*.

A persistência de objetos é provida pela classe *%Persistent*, que define os métodos da *Persistence Interface* e o compilador de classe que administra a evolução do esquema e a projeção do SQL. Para que a classe possa ser persistente, os seguintes requisitos devem ser atendidos:

- sua superclasse primária deve ser também *%Persistent*;
- seu *ClassType* deve ser persistente.

A figura 51 mostra o *script* de geração da classe exemplo Agencia no banco de dados orientado a objetos Caché.

```
Class modelo.Agencia Extends %Persistent [ ClassType = persistent, ProcedureBlock ]
{
Relationship PosicaoConta As modelo.Posicao [ Cardinality = many, Inverse = Agencia ];
Property  ChaveAgencia As %Integer [ Required ];
Index ChaveAgenciaIndex On ChaveAgencia [ Unique ];
Property NomeAgencia As %String [ Required ];
Property  AgrupadorAgencia As %Integer [ Required ];
Property EnderecoAgencia As %String [ Required ];
Property FormatoAgencia As %String [ Required ];
Property CidadeAgencia As Cidade [ Required ];
}
```

Figura 51 - *Script* da criação da classe Agencia.

A classe dimensão Tempo também é definida como persistente. O atributo NomeMes utiliza a declaração *Property* para definir uma propriedade de valores possíveis para uma *String*, especificando valores para a coleção na lista de palavras-

chave dos meses do ano. Esta aplicação da declaração permite a utilização do conceito da declaração *enum* da ODL.

O atributo ChaveMes utiliza a mesma propriedade, porém, especificando o maior e menor valor para a identificação numérica de cada mês do ano, considerando do mês 01 ao mês 12. A figura 52 mostra o *script* de geração da classe no banco de dados orientado a objetos Caché.

```
Class modelo.Tempo Extends %Persistent [ ClassType = persistent, ProcedureBlock ]
{
Relationship PosicaoTempo As modelo.Posicao [ Cardinality = many, Inverse = Tempo ];
Property Ano As %Integer [ Required ];
Property ChaveData As %String [ Required ];
Property ChaveMes As %Integer(MAXVAL = 12, MINVAL = 1);
Property Data As %Date [ Required ];
Property DiasdoAno As %Integer;
Property DiaSemana As %String;
Property NomeMes As %String (VALUELIST=",JANEIRO,FEVEREIRO,MARÇO,ABRIL,MAIO,
JUNHO, JULHO,AGOSTO,SETEMBRO,OUTUBRO,NOVEMBRO,DEZEMBRO");
Index DataIndex On Data [ Unique ];
}
```

Figura 52 - *Script* da criação da Classe Tempo.

Considerando a superclasse Pessoa, uma generalização das classes Gerente e Cliente, com atributos gerais para a superclasse e específicos para as classes especializadas, o *script* da figura 53 define tais características.

```
Class modelo.pessoa Extends (%Persistent) [ ClassType = persistent, ProcedureBlock ]
{
Property nomepessoa As %String [ Required ];
Property chavepessoa As %Integer [ Required ];
Property cidadepessoa As Cidade [ Required ];
Property enderecopessoa As %String;
}
```

Figura 53 - *Script* da criação da superclasse Pessoa.

A partir da superclasse definida através da geração da classe Pessoa, são especializadas as classes Gerente e Cliente, com os atributos correspondentes. As classes herdam as características da classe Pessoa, e agregam suas próprias características, conforme mostra o *script* da figura 54a (classe Gerente) e figura 54b (classe Cliente).

```
Class modelo.Gerente Extends modelo.Pessoa [ ClassType = persistent, ProcedureBlock ]
{
Relationship PosicaoGerente As modelo.Posicao [ Cardinality = many, Inverse = Gerente ];
Property GerenteDepto As %String;
Property GerenteDesde As %Date [ Required ];
}
```

(a)

```
Class modelo.Cliente Extends modelo.Pessoa [ ClassType = persistent, ProcedureBlock ]
{
Relationship PosicaoCliente As modelo.Posicao [ Cardinality = many, Inverse = Cliente ];
Property CPFCliente As %Integer;
Property DataNascimento As %Date;
Index ClienteIndex On CPFCliente [ Unique ];
}
```

(b)

Figura 54 - *Script* da criação das classes Gerente e Cliente .

O Caché suporta uma variedade de tipos de classe para comportamentos especializados. As classes são divididas em classes tipo de dados e classes objeto. Classes tipo de dados representam valores literais como *string*, *integer* e *date* e são usadas para criar propriedades literais de outros objetos, não possuindo propriedades e não podendo ser instanciadas.

Classes objeto podem ter propriedades e podem ser instanciadas. A maior parte das classes consiste de subclasse de classes de sistema chamadas %*RegisteredObject*, que fornecem automaticamente muito do comportamento básico de um objeto. Uma classe objeto tem um conjunto padrão de comportamentos que incluem:

- sua instanciação ativa a alocação automática de memória do sistema para suas propriedades;
- sua instanciação ativa a criação automática da referência OREF;
- suporta polimorfismo.

Classes objeto são classificadas de acordo com seu comportamento no banco de dados, como objeto transiente, persistente ou serial. Um objeto transiente (derivado diretamente da classe %*RegisteredObject*) não tem nenhum comportamento armazenado, existindo apenas em memória. Uma classe persistente (derivada da classe *%Persistent*) pode ser armazenada em um banco de dados. Objetos seriais (derivados da classe *%SerialObject*) podem ser embutidos em outros objetos; um

objeto serial só pode ser armazenado em banco de dados quando é embutido em um objeto persistente.

A classe *%SerialObject* gera um objeto com a habilidade de produzir uma *string* representando o estado do objeto (ou seja, suas propriedades atuais). A criação desta *string* é conhecida como serialização do objeto. Como um objeto persistente, um objeto serial pode ser usado como um tipo propriedade, mas seu comportamento em disco difere de um objeto persistente. Quando usado como uma propriedade, é conhecido como um objeto embutido.

Objetos embutidos têm representação diferente em memória e em disco:

- em memória é representado como um objeto separado e não se diferencia de outro tipo de objeto; o valor em memória de um atributo de objeto embutido é um OREF que se refere à representação do objeto em memória;
- em disco é armazenado como parte do objeto no qual está contido, não possuindo identidade individual (OID) e não podendo ser referenciado por outros objetos; os valores das propriedades do objeto embutido são serializados e armazenados com os outros atributos do objeto.

A figura 55 mostra uma aplicação da classe *%SerialObject* com a utilização da classe Cidade, referenciada por outras classes. A classe serial é embutida nas classes Pessoa, Agencia e UnidadeDomiciliar do modelo exemplo. A figura 55 registra a classe serial Cidade como um objeto embutido na classe Pessoa.

```
Class modelo.Cidade Extends %SerialObject [ ClassType = serial, ProcedureBlock ]
{
Property descricaoregiao As %String (VALUELIST = ",NORTE,NORDESTE,SUL,SUDESTE,
CENTRO-OESTE");
Property nomecidade As %String [ Required ];
Property ufcidade As %String(MAXLEN = 02) [ Required ];}
```

Figura 55 - *Script* da criação da classe serial Cidade.

6.6.2.2 Geração da Classe Fatos

Os Fatos também são mapeados para classes. Analisando a natureza da classe Fatos, cada fato é uma classe associativa, de acordo com a granularidade dos dados. Como definido na seção anterior, o relacionamento entre a classe Fatos

caracteriza uma agregação com as classes dimensão. A figura 56 mostra o *script* dos atributos da classe Fatos PosicaoConta.

```
Class modelo.PosicaoConta Extends %Persistent [ ClassType = persistent, ProcedureBlock ]
{
Property JurosCobrados As %Currency;
Property JurosPagos As % Currency;
Property NumeroTransacoes As %Integer;
Property SaldoData As % Currency;
Property SaldoMedioMensal As % Currency;
Property TaxasCobradas As % Currency;
Property TotalCredito As % Currency;
Property TotalDebito As % Currency;
…
}
```

Figura 56 - *Script* da geração dos atributos da classe Fatos.

6.6.2.3 Geração de identificadores

Um objeto é uma instância específica de uma classe. Um objeto pode existir em disco, em memória ou em uma aplicação do cliente. No Caché, os objetos podem estar em disco ou em memória. Os objetos podem ser referenciados de dois modos diferentes: OREF ou OID.

- OREF: uma referência do objeto. Um valor que se refere a uma instância específica do objeto na memória. Cada vez que um objeto é carregado para a memória pode ter um valor de OREF diferente.
- OID: um identificador do objeto. Um valor que identifica unicamente um objeto persistente no banco de dados. Uma vez que o objeto receba um valor de OID, este valor não muda.

Pode-se usar o OID de um objeto persistente para a localização do objeto e trazê-lo para a memória. Uma vez na memória, o sistema assinala-o com um valor OREF que a aplicação pode usar para acessar o objeto e seu conteúdo. Quando um objeto persistente é armazenado no banco de dados, os valores de quaisquer referências para outros objetos persistentes são armazenados como valores OID. Para atributos de objeto que não têm OIDs, o valor literal do objeto é armazenado juntamente com o restante do estado do objeto persistente.

O Caché gerencia o OID através de mecanismos próprios, atribuindo automaticamente para cada nova instância de objeto criado um OID, na forma de um número inteiro seqüencial.

Quando as classes Dimensão e Fatos são criadas, automaticamente são implementados os OIDs de cada uma delas.

6.6.2.4 Geração dos Relacionamentos entre as Classes

As classes dimensão Conta e Cliente caracterizam um relacionamento n:n, gerando a classe *bridge* de associação ContaCliente, conforme o *script* gerado na figura 57 para as classes Conta, Cliente e ContaCliente, respectivamente:

```
Class modelo.ContaCliente Extends %Persistent [ ClassType = persistent, ProcedureBlock ]
{
Relationship Conta As modelo.Conta [ Cardinality = one, Inverse = ItemCta ];
Relationship Cliente As modelo.Cliente [ Cardinality = one, Inverse = ItemCli ];
Property Titular As %boolean [ Required ];
}

Class modelo.Cliente Extends modelo.Pessoa [ ClassType = persistent, ProcedureBlock ]
{
Relationship PosicaoCliente As modelo.Posicao [ Cardinality = many, Inverse = Cliente ];
Property CPFCliente As %Integer;
Property DataNascimento As %Date;
Index ClienteIndex On CPFCliente [ Unique ];
}

Class modelo.Conta Extends Extends %Persistent [ ClassType = persistent, ProcedureBlock ]
{
Relationship PosicaoConta As modelo.Posicao [ Cardinality = many, Inverse = Conta ];
Property DataAbertura As %Date;
Property DescricaoCategoria As %String;
Property DescricaoTipoConta As %String;
Property NumeroConta As % String;
Index ContaIndex On NumeroConta [ Unique ];
}
```

Figura 57 - *Script* da geração das classes associadas.

A característica de encapsulamento no Caché é utilizada para se manter a representação externa de uma classe - propriedades e métodos - independente de sua implementação atual:

- propriedades encapsulam dados: é possível mudar o banco de dados enquanto se mantém a mesma propriedade de *interface*;

- métodos encapsulam comportamento: quando um objeto persistente é carregado do banco de dados é automaticamente associado com os métodos definidos para sua classe, podendo mudar a implementação destes métodos independentemente dos dados no banco de dados e de qualquer usuário destes métodos.

O Polimorfirmo é a habilidade para invocar o mesmo método em objetos de tipos diferentes (mas com uma superclasse comum). Cada objeto executa a implementação específica do método. Isto permite a utilização de novos tipos de objeto em aplicações existentes sem a necessidade de se modificar as aplicações.

As aplicações podem se adaptar facilmente a novas circunstâncias fornecendo implementações especializadas das operações. O sistema de *runtime* executa automaticamente a implementação correta baseado no tipo de objeto.

6.6.3 Mapeando Objetos

O tipo de dados objeto é uma abstração das entidades do mundo real, que pode ser armazenado em um banco de dados (Zendulka, 2001). O objeto é um esquema com um nome, grupo de atributos e métodos. Um atributo pode ser um tipo de dado embutido ou um tipo de dado definido pelo usuário. Isso permite determinar tipos de objeto com uma complexa estrutura de dados. Os métodos são tipos de objeto de três categorias: membro, estático ou comparação.

Métodos da categoria membro são usados para modelar o comportamento dos objetos. Métodos estáticos são usados para a modelagem do tipo de objeto como um todo. E métodos de comparação são usados para comparar instâncias de um tipo de objeto. Cada tipo de objeto tem um método construtor definido pelo sistema. No modelo, Criar() e Excluir() são os métodos construtor e destrutor dos objetos, respectivamente, como registra a figura 40.

Similarmente a uma classe em linguagem de programação, um tipo de objeto é um modelo para instâncias chamado objetos. Objetos podem ser instanciados através de um método construtor de um dado tipo de objeto e armazenados em uma tabela objeto.

Observando o paradigma da OO, pode-se agrupar objetos em classes. Estas classes podem ser encapsuladas tanto com as propriedades estáticas como com as

dinâmicas destes objetos. As propriedades dinâmicas são as operações que podem ser aplicadas nos objetos para mudar suas características. Porém, no contexto de bancos de dados, os objetos (dados) são estáticos no sentido de que, uma vez que existam no sistema, eles não alteram suas características (propriedades estáticas), até que sejam levados a um armazenamento auxiliar. Conseqüentemente, as duas primeiras ações a serem aplicadas nestes objetos devem ser a de criá-los e destruí-los.

As classes do modelo exemplo são povoadas através do instanciamento de objetos. Utilizando o Caché Terminal é possível trabalhar com aplicativos Caché. O Caché Terminal suporta *links* DDE (*Dynamic Data Exchange*) para permitir que outras aplicações utilizem sua habilidade de comunicação com host remoto.

O método de classe Caché *%New* é usado para a criação de uma nova instância de um objeto. A figura 58a mostra o comando para criar um novo objeto da classe Agencia e instanciá-lo, através do Caché Terminal.

O método de classe Caché *%Save* é usado para armazenar um objeto persistente no banco de dados. Quando um objeto torna-se persistente pela primeira vez o método *%Save* gera automaticamente um valor de OID. A figura 58b mostra o comando que torna persistente um objeto da classe Agencia.

```
Set objAgencia = ##class(modelo.Agencia).%New()
```
(a)

```
Set ok = objAgencia.%Save()
```
(b)

Figura 58 - *Script* da geração e persistência de objetos.

A figura 59 apresenta o resumo das tarefas desenvolvidas ao término da quinta etapa da metodologia. Os padrões da OO são mapeados para o BDOO.

Implantação de modelos multidimensionais mapeados em BDOO

Etapa 5

- Classes:
    - nomeação de classes e respectivos atributos;

- definição de restrições;
- validação e adaptação dos padrões da ODMG no BDOO Caché;
- *script* de criação da superclasse Pessoa;
- *script* de criação das classes Cliente e Gerente, herdando as propriedades da superclasse Pessoa;
- *script* de criação da classe PosicaoConta como uma classe Fatos;
- *script* de criação das classes associativas ContaCliente e ContaProduto;
- *script* de criação das demais classes de dimensão;
- *script* para instanciação de objetos.

- Relacionamentos:
  - transposição do relacionamento de especialização / generalização para a herança através de *Extends*, definido nas classes especializadas;
  - transposição do relacionamento 1:n em relacionamentos inversos;
  - utilização das classes seriais do BD Caché para compor relacionamentos de referência simples;
  - definição de uma classe de associação para representar a tabela *bridge* com relacionamentos inversos entre as classes associadas.

Figura 59 - Resumo da etapa 5 da metodologia.

6.7 Considerações Finais

O presente capítulo apresentou a aplicação da metodologia proposta, definida no capítulo anterior, que pode ser utilizada em um processo de implantação do modelo multidimensional no ambiente de DW através do paradigma da orientação a objetos.

A aplicação da proposta metodológica foi apresentada através de um estudo de caso. Este desenvolvimento seguiu as etapas e respectivas tarefas, detalhando os resultados das atividades desenvolvidas. Foi definido o modelo de negócio, utilizando o diagrama ER para a geração do modelo multidimensional, que foi mapeado, posteriormente, para o modelo multidimensional utilizando a UML, através dos diagramas de classe e estrutura composta. A partir destes diagramas, a implementação foi definida com as propriedades da ODL e mapeadas para o banco de dados Caché.

As etapas aqui aplicadas auxiliam no processo de definição e implantação dos modelos multidimensionais, definidas sistematicamente e com a descrição das tarefas em uma seqüência de execução. Estas etapas geram um roteiro a ser seguido pelo projetista do modelo multidimensional, partindo do ambiente operacional até sua efetiva implementação e persistência no ambiente de DW, sistematizando os procedimentos e relacionado todas as tarefas de conversão e mapeamento a serem realizadas.

A utilização da representação do modelo estático através dos diagramas de classes e estrutura composta da UML auxilia na compreensão e especificação do modelo.

A utilização da linguagem padrão ODL impõe o formalismo para posterior implementação em qualquer modelo de banco de dados orientado a objeto, através da aplicação das propriedades e conceitos da OO de forma genérica, independente de plataforma ou fabricante de banco de dados.

As conclusões gerais obtidas com esta tese, bem como sugestões de pesquisas futuras, encontram-se no próximo capítulo.

7 CONCLUSÕES

Neste capítulo são apresentadas as principais conclusões a respeito das etapas da metodologia apresentada e de sua aplicação. Finalizando o trabalho proposto são analisados os resultados da metodologia e sua aplicação.

Na seção 7.1 são apresentadas as conclusões sobre a metodologia apresentada, tendo como referencial as etapas propostas e aplicadas no capítulo anterior.

A seção 7.2 relaciona sugestões para trabalhos futuros para a continuidade desta pesquisa envolvendo as aplicações do paradigma orientado a objetos.

7.1 Conclusões

O estudo apresentado abordou metodologias existentes para a modelagem multidimensional, mostrando que as propostas não sugerem etapas que possam orientar um projetista do início ao fim do processo de implantação de ambientes de DW, segundo o paradigma da orientação a objetos. As propostas que analisam a OO oferecem contribuições específicas, focando conceitualmente este paradigma exclusivamente pela representação através do diagrama de classes.

Portanto, visando contribuir no sentido de oferecer um roteiro de implementação do modelo OO no ambiente multidimensional, a proposta foi elaborada relacionando atividades nas várias etapas do processo. A metodologia orienta o mapeamento do modelo operacional para o modelo multidimensional, sua posterior representação através da UML, sua formalização em ODL e a persistência em BDOO.

Através da aplicação da proposta no estudo de caso, pode-se observar que a metodologia estabelece os procedimentos e tarefas a serem concluídas em cada uma das etapas, permitindo ao projetista o desenvolvimento completo do processo, utilizando os conceitos da OO e, conseqüentemente, a geração de um modelo eficiente para o seu mapeamento no banco de dados. A metodologia foi aplicada a um sistema baseado no modelo de Serviços Financeiros proposto por Kimball, podendo ser aplicado a qualquer modelo de negócios, considerando que a representação dos modelos operacionais centram-se basicamente no modelo ER.

Analisando o paradigma da orientação a objetos e sua real aplicação em BDOO, a metodologia utiliza em todo seu processo os conceitos e componentes da orientação a objetos, desde sua concepção inicial, através do modelo estático, até

sua implementação. A metodologia utiliza as vantagens do paradigma orientado a objetos, permitindo o tratamento de objetos complexos, utilizando o conceito de herança múltipla sem restrições, o que permite aumentar a reusabilidade do processo.

Observa-se que algumas propostas da utilização da OO, independente do ambiente operacional ou de DW, utilizam a representação através do diagrama de classes, porém, necessitam de um processo de reengenharia para o mapeamento no modelo ER (Ambler, 1999; Keller, 1997). Este processo não é analisado nos trabalhos analisados, que mapeiam o modelo ER em diagrama de classes, mas necessitam de novo mapeamento, do modelo OO em modelo relacional, para a implantação em BDR. Isso gera um trabalho complementar no processo e não permite a utilização de conceitos de otimização de aplicação da OO, como a herança múltipla, por exemplo.

Diante disso, a concepção da metodologia proposta abrange todas as suas etapas baseando-se no paradigma da OO, gerando um único processo de conceituação, definição e implementação, utilizando uma única abordagem.

A partir dos dados gerados no teste de desempenho, realizado com o modelo multidimensional (Apêndice A), o modelo de BDOO apresenta-se como uma opção viável para o modelo multidimensional, permitindo o tratamento de objetos complexos e utilizando o conceito de herança múltipla sem restrições, que aumenta a reusabilidade do processo e obtém um desempenho compatível com os modelos relacionais.

Neste contexto, as principais contribuições desta pesquisa são assim relacionadas:

- uma metodologia contemplando o ciclo para o desenvolvimento e aplicação do modelo multidimensional em BDOO;
- a descrição da utilização dos conceitos da OO no modelo multidimensional;
- a aplicação da linguagem padrão da OO, a ODL no modelo multidimensional;
- a utilização da representação do modelo multidimensional através da linguagem UML, utilizando os diagramas estáticos de classes e estrutura composta;

- a integração do processo completo da modelagem à implementação do modelo multidimensional no paradigma OO, com a aplicação de seus conceitos e características.

Como proposto no início do trabalho, a pesquisa aqui apresentada fez um estudo das características dos modelos multidimensionais. O trabalho analisou e elaborou um estudo comparativo dos modelos existentes para a modelagem multidimensional, independente das abordagens utilizadas. Complementarmente a este item, fez-se um estudo das características das abordagens que utilizam a orientação a objetos em sua concepção. O resultado deste estudo mostrou que nenhuma abordagem propõe uma metodologia completa da modelagem multidimensional, desde a escolha do modelo de negócio até sua implementação, utilizando o paradigma da OO.

Foram analisados os conceitos e mecanismos da OO que pudessem ser utilizados na modelagem multidimensional e observou-se que podem ser aplicados ao modelo utilizado em ambiente de DW, deixando-o mais parecido com a realidade do negócio, através da implementação de herança múltipla, tipos de dados complexos e encapsulamento, facilitando o trabalho do projetista. Conforme definido no capítulo 3, os conceitos da orientação a objetos que podem ser utilizados para a modelagem multidimensional são: herança, polimorfismo e o encapsulamento. Quanto aos componentes da orientação a objetos, podem ser utilizados na modelagem multidimensional: classes, objetos, atributos, relacionamentos e operações.

Durante o estudo dos modelos existentes para a abordagem multidimensional, pode-se verificar que as poucas propostas que se preocupam com a real aplicabilidade fazem-no com base no modelo relacional e não sugerem o processo de reengenharia para mapear o modelo conceitual OO para o modelo físico relacional.

A tecnologia de BDOO apresenta algumas vantagens sobre a tecnologia de BDR, especificamente seu mapeamento direto dos conceitos da OO, utilizada largamente como padrão para o modelo conceitual de sistemas. O teste de desempenho realizado mostrou que no ambiente de DW o modelo de BDOO oferece desempenho equivalente ao modelo relacional, além de sua facilidade de implementação.

Portanto, os conceitos e componentes da orientação a objetos podem ser utilizados integralmente no modelo multidimensional, gerando uma sistemática única de processo - da definição à implementação. E o presente trabalho propõe e aplica a metodologia para que as tarefas possam ser desenvolvidas seqüencialmente, através de etapas e refinamentos, que convalidam o modelo operacional com o multidimensional, oferecendo ao ambiente de DW uma proposta integral no paradigma orientado a objetos.

7.2 Trabalhos futuros

Observando os resultados da pesquisa proposta, podem ser relacionadas algumas sugestões para trabalhos futuros, complementando o trabalho:

- desenvolvimento de uma ferramenta que possa fazer a geração do modelo multidimensional para o banco de dados OO;
- extensão do modelo conceitual para o modelo lógico formal;
- desenvolvimento de componentes para a extração de dados do modelo operacional e mapeamento para o modelo OO;
- aplicação da metodologia proposta em ambiente real;
- desenvolvimento de componentes para o mapeamento do padrão ODMG para BDOO;
- estudo das aplicações de front-end com os modelos multidimensionais em banco de dados OO;
- especificação das operações OLAP segundo o paradigma da orientação a objetos, para a análise dos dados.

Estes tópicos podem ser desenvolvidos a partir da metodologia proposta e dos resultados aqui apresentados.

REFERÊNCIAS BIBLIOGRÁFICAS

ABELLÓ, A.; SAMOS, J.; SALTOR, F. A Data Warehouse multidimensional data models classification. Technical Report LSI-2000-6. Dep. Lenguages y Sistemas Informáticos, Universidad de Granada, 2000a.

___. Benefits of an Object-Oriented Multidimensional Data Model. International Symposium on Objects and Databases, pág 141-152, France, 2000, vol. 1.944 of Lecture Notes in Computer Science, Springer, 2000b.

___. A Framework for the classification and description of multidimensional data models. 12th International Conference on Database and Expert Systems Applications (DEXA), 2001a.

___. Multi-star conceptual schemas for OLAP systems. Technical Report LSI-01-45-R. Dep. Lenguages y Sistemas Informáticos, Universidad Politecnica de Catalunya, 2001b.

___. Understanding analysis dimensions in a multidimensional object-oriented model. International Workshop on Design and Management of Data Warehouses (DMDW'2001). Switzerland, 2001c.

___. Understanding facts in a multidimensional object-oriented model. 4th International Workshop on Data Warehousing and OLAP (DOLAP 2001). Atlanta (USA), 2001d.

___. YAM² (Yet Another Multidimensional Model): An extension of UML. International Database Engineering & Applications Symposium (IDEAS'02). Canada, July 2002.

___. YAM²: a multidimensional conceptual model extending UML. Information Systems, In Press, Corrected Proof, Available online, February 2005.

ABEPRO – Associação Brasileira de Engenharia da Produção. Engenharia de Produção: Grande área e diretrizes curriculares. Disponível em <http://www.abepro.producao.ufrgs.br/>. Acesso em: out 2005.

AGRAWAL, R.; GUPTA, A.; SARAWAGI, S. Modeling Multidimensional Databases. 13th Int. Conf. on Data Engineering (ICDE), IEEE, 1997.

ALSAADI, A. A Performance Analysis Approach Based on the UML Class Diagram. 4th International Workshop on Software and Performance. (WOSP, 2004), 2004.

AMBLER, S. W. Análise e projeto orientados a objeto, volume II: seu guia para desenvolver sistemas robustos com tecnologia de objetos. Rio de Janeiro: Infobook, 1998.

___. UML 2 Class Diagrams. 2005. Disponível em:<http://www.agilemodeling.com/artifacts**>**. Acesso em: 20 abr 2005.

ATKINSON, M.; BANCILHON, F.; DEWITT, D.; DITTRICH, K.; MAIER, D.; ZDONIK, S. The Object-Oriented Database System Manifesto. First International Conference on Deductive and Object-Oriented Databases, Japan, 1989.

BADIA, A. From Conceptual Models to Data Models. In P. Van Bommel, Transformation of Knowledge, Information and Data: Theory and Applications (pp. 148-170), Hershey, PA: Information Science Publishing, 2005.

BAEKGAARD, L. Event-Entity-Relationship Modeling in data warehouse environments. In: Proc. of the ACM 2nd Int. Workshop on Data warehousing and OLAP (DOLAP'99), 1999.

BALABAN, M.; SHOVAL, P. Enforcing Cardinality Constraints in the ER Model with Integrity Methods. In K. Siau (Ed.), Advanced Topics in Database Research (pp. 1-16). Hershey, PA: Idea Group Publishing, 2002.

BARRY, D. Comparison of object and relational concepts. 2004. Disponível em: <http://www.service-architecture.com/database/articles>. Acesso em: 20 jan 2005

BATRA, D. Conceptual Data Modeling Patterns: Representation and Validation. Journal of Database Management, Vol. 16, No. 2, pp. 84-106, 2005.

BECKER, S. A. Conceptual Data Modeling in an Object-Oriented Process (Part One). Journal of Conceptual Modeling. Issue: 18, 2001. Disponível em: <http://www.inconcept.com/jcm>. Acesso em: 15 ago 2003.

BLOOR, R.; BLOOR, B. The failure of relational database, the rise of object technology and the need for the hybrid database. Baroudi Bloor International, 2004.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML guia do usuário. Rio de Janeiro: Campus, 2000.

BUZYDLOWSKI, J.; SONG, II-Y.; HASSELL, L. A Framework for object-Oriented On-Line Analytic Processing. In Proc. Of the ACM 1st Int. Workshop on Data warehousing and OLAP (DOLAP). Washington DC, USA, 1998.

CAMPOS, M. L. M.; AMARAL, G. C. M. Modelando Metadados de Qualidade no Data Warehouse. IV Simpósio de Desenvolvimento e Manutenção de Software da Marinha (SDMS 2004). Rio de Janeiro, 2004.

CARDOSO, C. UML na prática: do problema ao sistema. Rio de Janeiro: Ciência Moderna Ltda, 2003.

CATTELL, R. G. G.; BARRY, D. K.; BERLER, M. The object data standard: ODMG 3.0. San Francisco: Morgan Kaufmann Publishers, 2000.

CAVERO, J. M.; MARCOS, E.; PIATTINI, M.; SANCHEZ, A. A Methodology for Datawarehouse Design: Conceptual Modeling. In S. A. Becker (Ed.), Data Warehousing and Web Engineering (pp. 185-197). Hershey, PA: IRM Press, 2002.

CAVERO, J. M.; COSTILLA, C.; MARCOS, E.; PIATTINI, M. G.; SANCHEZ, A. A Multidimensional Data Warehouse Development Methodology. In P. C. Pendharkar (Ed.), Managing Data Mining Technologies in Organizations: Techniques and Applications. Hershey, PA: Idea Group Publishing, 2003.

CHAO, C. M. Incremental maintenance of object-oriented data warehouses. Information Sciences, Volume 160, Issues 1-4, 2004.

CHEN, P. The Entity-Relationship Model: Toward a Unified View of Data. ACM Transactions on DataBase Systems, v.1, n.1, 1976.

___. Modelagem de Dados. São Paulo: Makron Books, 1990.

CODD, E.F. A Relational Model of Data for Large Shared Data Banks. ACM, Vol. 13, No. 6, 1970, pp. 377-387.

CODD, E.F.; CODD, S.B.; SMALLEY, C.T. Providing OLAP (Online Analytical Processing) to user-analysts: an it mandate. Technical Report, 1993. Disponível em: <http://www.essbase.com/download_files/resource_library/white_papers/providing_olap_to_user_analysts.pdf>. Acessado em 10 Mar 2004.

COUGO, P. S. Modelagem conceitual e projeto de banco de dados. Rio de janeiro: Campus, 1997.

DATE. C.J. Introdução a sistemas de banco de dados. Rio de janeiro: Elsevier, 2003.

DANOCH, R., SHOVAL, P. & BALABAAN, M. Comprehension of Hierarchical ER Diagrams Compared to Flat ER Diagrams. In J. Krogstie, T. Halpin & K. Siau, Information Modeling Methods and Methodologies (pp. 241-257), Hershey, PA: Idea Group Publishing, 2005.

DINTER, B.; SAPIA, C.; BLASCHKA, M.; HÖFLING, G. OLAP Market and Research: initiating the cooperation. Journal of Computer Science and Information Management. Vol. 2, No. 3, 1999.

FOWLER, M. UML essencial: um breve guia para a linguagem de modelagem de objetos. Porto Alegre: Bookman, 2000.

FREITAS, G. M.; LAENDER, A. H. F.; CAMPOS, M. L. M. Getting Users Involved in the Development of Data Warehouse Applications. 4th Design and Management of Data Warehouses. (DMDW'2002). Canada, 2002.

GIOVINAZZO, W. A. Object-oriented data warehouse design: building a star scheme. New Jersey: Prentice Hall, 2000.

GOLFARELLI, M.; MAIO, D.; RIZZI, S. A methodological Framework for Data Warehouse Design. ACM 1st Int. Workshop on Data warehousing and OLAP (DOLAP'98), 1998a.

___. Conceptual Design of Data Warehousing from E/R Schemes. In Proceedings of the 31st Hawaii International Conference on System Sciences, p. 334-343. IEEE Computer Society, 1998b.

___. The Dimensional Fact Model: a Conceptual Model for Data Warehouses. International Journal of Cooperative Information Systems, 215-247, 1998c.

GOPALKRISHNAN, V.; LI, Q.; KARLAPALEM, K. Star/Snow-flake Schema Driven Object-Relational Data Warehouse Design and Query Processing Strategies. In First Internacional Conference on Data Warehousing and Knowledge discovery. Italy (DAWAK'99), 1999.

GROSSMAN, M.; ARONSON, J. A.; MCCARTHY, R.V. Does UML make the grade? Insights from the software development community. Information and Software Technology, Volume 47, November 2005.

HALPIN, T. Information Analysis in UML and ORM: A Comparison. In K. Siau (Ed.), Advanced Topics in Database Research (pp. 307-323). Hershey, PA: Idea Group Publishing, 2002.

HIREMATH, H. R.; SKIBNIEWSKI, M. J. Object-oriented modeling of construction processes by unified modeling language. Automation in Construction, Volume 13, Issue 4, 2004.

HITCHMAN, S. The Entity Relationship Model And Practical Data Modelling. Journal of Conceptual Modeling. March, 2004a. Disponível em <http://:www.inconcept.com/jcm>. Acesso em: 15 jan 2005.

___. Entity class classification and entity relationship types. Journal of Conceptual Modeling. March, 2004b. Disponível em <http://:www.inconcept.com/jcm>. Acesso em: 15 jan 2005.

___. Strategic Conceptual Modelling: An Example From Practice. Journal of Conceptual Modeling. March, 2004c. Disponível em <http://:www.inconcept.com/jcm>. Acesso em: 15 jan 2005.

HÜSEMANN, B.; LECHTENBÓRGER, J.; GOTTFRIED, V. Conceptual Data Warehouse Design. 2nd Int. Workshop on Design and Management of DW (DMDW). Sweden, 2000.

INMON. W. H. Como construir o data warehouse. Rio de Janeiro: Campus, 1997.

___. Data Warehousing – como transformar informações em oportunidades de negócios. São Paulo: Berkeley, 2001.

___. A Generic Data Model. 2002. Disponível em <http://www.dbazine.com/inmon14.shtml>. Acesso em: 12 out 2003.

INTERSYSTEMS. Documentação técnica. 2005. Disponível em <http://platinum. intersystems.com>. Acesso em: 20 out 2005.

JENSEN, M. R.; MOLLER, T. H.; PEDERSEN, T. B. Converting XML DTDs to UML diagrams for conceptual data integration. Data & Knowledge Engineering, Volume 44, Issue 3, 2003.

JONES, T. H.; SONG, I.-Y. Ternary Relationships: Semantic Requirements and Logically Correct Alternatives. In K. Siau (Ed.), Advanced Topics in Database Research (pp. 17-33), Hershey, PA: Idea Group Publishing, 2002.

KELLER, W. Mapping objects to tables - a pattern language. European Pattern Languages of Programming Conference. Siemens Technical Report 120/SW1/FB. Germany, 1997.

KHOSHAFIAN, S. Banco de Dados orientado a objetos. Rio de Janeiro: Infobook, 1994.

KIMBALL, R. A dimensional modeling manifesto. 1997. Disponível em: <http://www.rKimball.com/html/articlesArchitecture.html>. Acesso em: 10 nov 2003.

___. The Data Warehouse lifecycle toolkit. New York: Wiley computer Publishing, 1998.

___. Data Warehouse toolkit: o guia completo para modelagem multidimensional. Rio de Janeiro: Campus, 2002.

___. The Fundamentals of Data Warehousing. Intelligente Enterprise. 2003. Disponível em: <http://www.datamirror.com>. Acesso em: 20 abr 2004.

KIMBALL, R.; ROSS, M. Fables and Facts. Intelligente Enterprise. Outubro, 2004. Disponível em: <http://www.intelligententerprise.com/info_centers/data_warehousing/>. Acesso em: 20 mar 2005.

LECHTENBÖRGER, J.; VOSSEN. G. Multidimensional normal forms for data warehouse design. Information Systems, Volume 28, Issue 5, July 2003.

LEE, H. J.; LEE, S. W.; KIM, H. J. Design and implementation of an extended relationship semantics in an ODMG-compliant OODBMS. The Journal of Systems and Software, 76. 2005.

LEHNER, W.; ALBRECHT, J.; WEDEKIND, H. Normal Forms for Multidimensional Databases. 8th Int. Conf. on Statistical and Scientific Database Management (SSDBM). IEEE Computer Society, 1998.

LEVENE, M.; LOIZOU, G. Why is the snowflake schema a good data warehouse design? Information Systems, Volume 28, Issue 3. 2003.

LONGMAN, C. Data Warehouse Lifecycle Management - Concepts and Principles. DMReview. April 2004. Disponível em <http://www.dmreview.com/whitepaper>. Acesso em 20 mar 2005.

LUJÁN-MORA, S.; TRUJILLO, J..A Comprehensive Method for Data Warehouse Design. Design and Management of Data Warehouses (DMDW'2003).Germany, 2003.

___. A Data Warehouse Engineering Process. 3rd International Conference in Advances in Information Systems (ADVIS 2004), p. 14-23: Lecture Notes in Computer Science 3261. Turkey, 2004.

LUJÁN-MORA, S.; TRUJILLO, J.; SONG, I.Y. Extending UML for Multidimensional Modeling. 5th International Conference on the Unified Modeling Language (UML 2002), p. 290-304. Lecture Notes in Computer Science 2460, Germany, 2002.

LUJÁN-MORA, S.; TRUJILLO, J.; VASSILIADIS, P. Advantages of UML for Multidimensional Modeling. 6th International Conference on Enterprise Information Systems (ICEIS 2004), p. 298-305: ICEIS Press. Portugal, 2004a.

___. Data Mapping Diagrams for Data Warehouse Design with UML. 23rd International Conference on Conceptual Modeling (ER 2004), p. 191-204: Lecture Notes in Computer Science 3288. China, 2004b.

MACHADO, F. N. R. Projeto de data warehouse: uma visão multidimensional. São Paulo: Érica, 2000.

___. Banco de Dados: projeto e implementação. São Paulo: Érica, 2004.

MANIATIS, A.; VASSILIADIS, P.; SKIADOPOULOS, S.; VASSILIOU, Y.; MAVROGONATOS, G.; MICHALARIAS, I. A Presentation Model & Non-Traditional Visualization for OLAP. International Journal of Data Warehousing and Mining, Vol. 1, No. 1, pp. 1-36, 2005.

MATOS, A. V. UML: prático e descomplicado. São Paulo: Érica, 2002.

MELO, A. C. Desenvolvendo aplicações com UML 2.0: do conceitual à implementação. Rio de Janeiro: Brasport, 2004.

MENDELZON, A. O. & VAISMAN, A. A. Time in Multidimensional Databases. In M. Rafanelli (Ed.), Multidimensional Databases: Problems and Solutions (pp. 166-199), Hershey, PA: Idea Group Publishing, 2003.

MONTEIRO NETO, R. R. Mapeamento entre os modelos ER e Star. 1998. Disponível em: <http://genesis.nce.ufrj.br/dataware/>. Acesso em: 15 out 2001.

MOODY, D. L.; KORTINK, M. A. R. From Enterprise Models to Dimensional Models: A methodology for Data Warehouse and Data Mart Design. International workshop on Design and Management of Data Warehouses (DMDW'2000). Sweden, 2000.

MULLER, R. J. Projeto de Banco de dados: usando UML para modelagem de dados. São Paulo: Berkeley Brasil, 2002.

NASSU, E. A.; SETZER, V. W. Bancos de dados orientados a objetos. São Paulo: Edgard Blücher Ltda, 1999.

NGUYEN, T. B.; TJOA, A. M.; WAGNER R. An Object Oriented Multidimensional Data Model for OLAP. 1st Conf. on web_age Information Management (WAIM), no. 1846 in LNCS, pg. 69-82, 2000.

ODMG. Object Data Management Group. Disponível em <http://www.odmg.org>. Acesso em 10 jun 2003.

OLIVEIRA, A. G. Datawarehouse, conceitos e soluções. Florianópolis: Relativa Editora Ltda ME, 1998.

OLIVEIRA, W. J. Data Warehouse. Florianópolis: Visual Books, 2002.

OMG. Object Management Group. Unified Modeling Language: Superstructure version 2.0. Revised Final Adapted Specification (ptc/04-10-02). 2005. Disponível em <http://www.omg.org>. Acesso em 20 abr 2005.

ORR, K. Data Warehousing Technology. A White Paper. The Ken Orr Institute. Edição Revisada, 2000. Disponível em <http://www.kenorrinst.com>. Acesso em 15 abr 2004.

PATERSON, J. H., HADDOW, J. Approaches to object persistence in Java projects. The 9th Annual Conference on Innovation and Technology in Computer Science Education. ITICSE 2004. Leeds, United Kingdom. 2004.

PERALTA, V.; RUGGIA, R. Using Design Guidelines to Improve Data Warehouse Logical Design. 5th Design and Management of Data Warehouses (DMDW'2003). Germany, 2003.

PERALTA, V.; ILLARZE, A.; RUGGIA, R. On the Applicability of Rules to Automate Data Warehouse Logical Design. 15th Conference on Advanced Information Systems Engineering (CAiSE '03). Austria, 2003.

PHILIPPI, S. Model driven generation and testing of object-relational mappings. Journal of Systems and Software, In Press, Corrected Proof, September 2004.

PHIPPS, C.; DAVIS, K. C. Automating data warehouse conceptual schema design and evaluation. 4th Design and Management of Data Warehouses (DMDW'2002). Canada, 2002.

PRAKASH, W.; GOSAIN, A. Requirements Driven Data Warehouse Development. 15th Conference on Advanced Information Systems Engineering (CAiSE '03). Austria, 2003.

RAHAYU, J. W.; CHANG, E.; DILLON, T. S.; TANIAR, D. Performance evaluation of the object-relational transformation methodology. Data & Knowledge Engineering, Volume 38, Issue 3, September 2001.

RIZZI, S. UML-based Conceptual Modeling of Pattern-Bases. International Workshop on Pattern Representation and Management. Greece, 2004.

ROWEN, W.; SONG, I. Y.; ARYNTH, C. M.; EWEN, E. An Analysis of Many-to-Many Relationships Between Fact and Dimension Tables in Dimensional Modeling. Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2001). Switzerland, 2001.

RUMBAUGH, J. et al. Modelagem e projetos baseados em objetos. Rio de Janeiro: Campus, 1994.

SAPIA, C.; BLASCHKA, M.; HÖFLING, G.; DINTER B. Finding your way through multidimensional data models. In: Proc. Of 9th Int. Conf. on Database and Expert systems Application (DEXA), no. 1460 in LNCS, 1998a.

___.Extending the E/R Model for the Multidimensional Paradigm. In: Proc. of the 1st Intl. Workshop on Data Warehouse and Data Mining (DWDM'98). LNCS Vol. 1552, 1998b.

SIAU, K.; CAO, Q. How Complex is the Unified Modeling Language?. In K. Siau (Ed.), Advanced Topics in Database Research (pp. 294-306), Hershey, PA: Idea Group Publishing, 2002.

SIAU, K.; ERICKSON, J.; LEE, L.Y. Theoretical vs. Practical Complexity: The Case of UML. Journal of Database Management, Vol. 16, No. 3, pp. 40-57, 2005.

SINGH, H. Data warehouse. São Paulo: Makron Books, 2001.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. Sistema de Banco de Dados. São Paulo: Pearson Education do Brasil, 1999.

SHAH, D.; SLAUGHTER, S. Transforming UML Class Diagrams into Relational Data Models. In L. Favre (Ed.), UML and the Unified Process (pp. 217-236), Hershey, PA: IRM Press, 2003.

SHIN, S. K.; SANDERS, G. L. Denormalization strategies for data retrieval from data warehouses. Decision Support Systems, In Press, Corrected Proof, January 2005.

SONG, I.-Y. YANO, K.; TRUJILLO, J.; LUJAN-MORA, S. A Taxonomic Class Modeling Methodology for Object-Oriented Analysis. In J. Krogstie, T. Halpin & K. Siau, Information Modeling Methods and Methodologies. Hershey, PA: Idea Group Publishing, 2005.

THE COMMITTEE FOR ADVANCED DBMS FUNCTION. Third Generation Database System Manifesto. Computer Standards and Interfaces 13 (1991), North Holland. SIGMOD Record 19:3 Sept, 1990.

TORLONE, R. Conceptual Multidimensional Models. In M. Rafanelli (Ed.), Multidimensional Databases: Problems and Solutions (pp. 69-90). Hershey, PA: Idea Group Publishing, 2003.

TRUJILLO, J.; PALOMAR, M. An Object Oriented Approach to Multidimensional Database Conceptual Modeling (OOMD). In Proc. Of the ACM 1st Int. Workshop on Data Warehousing and OLAP (DOLAP), Washington DC (USA), 1998.

TRUJILLO, J.; PALOMAR, M.; GÓMEZ, J. The_GOLD definition language (GDL): an object oriented formal specification language for multidimensional databases. Symposium on

Applied Computing. Proceedings of the 2000 ACM Symposium on Applied Computing. Italy, p.346-350, 2000.

TRUJILLO, J.; PALOMAR, M.; GÓMEZ, J; SONG, I.-Y. Designing Data warehouse with OO conceptual models. IEEE – Institute of Electrical and Electronics Engineer. Vol 34, nº 12, p. 66-75, 2001.

TRUJILLO, J., LUJAN-MORA, S.; SONG, I.-Y. Applying UML for Designing Multidimensional Databases and OLAP Applications. In K. Siau (Ed.), Advanced Topics in Database Research, Volume 2 (pp. 13-36), Hershey, PA: Idea Group Publishing, 2003.

___. Applying UML and XML for Designing and Interchanging Information for Data Warehouses and OLAP Applications. Journal of Database Management, Vol. 15, No. 1, pp. 41-72, 2004.

TRYFONA, N.; BUSBORG, F.; CHRISTIANSEN, J. StarER: A Conceptual Model for Data Warehouse Design. In: Proc. of the ACM 2nd Int. Workshop on Data warehousing and OLAP (DOLAP'99), 1999.

TUROWSKI, K. Establishing Standards for Business Components. In K. Jakobs (Ed.) Information Technology Standards and Standardization: A Global Perspective (pp. 131-151), Hershey, PA: Idea Group Publishing, 2000.

URBAN, S. D. ; DIETRICH, S. W. Using UML Class Diagrams for a Comparative Analysis of Relational, Object-Oriented, and Object-Relational Database Mappings. Proceedings of the 34th SIGCSE technical symposium on Computer science education. New York: ACM Press, 2003.

VERSANT CORPORATION. Objects End-to-End The ODBMS Advantage. White Paper. 2001. Disponível em: <http://www.versant.com/resources/resources>. Acesso em: 15 ago 2004.

VERSANT CORPORATION. Which database is right for the new information systems? White Paper. 2004a. Disponível em: <http://www.versant.com/resources/resources>. Acesso em 20 jan 2005.

VERSANT CORPORATION. How to evaluate an object database. White Paper. 2004b. Disponível em: <http://www.versant.com/resources/resources>. Acesso em 20 jan 2005.

WAN, Q. C. Beyond the Star-Schema - Transitioning from an Entity-Oriented Model to a Metrics-Oriented Model for Multi-Dimensional Data Analysis. DMReview. 2004. Disponível em <http://www.dmreview.com/whitepaper>. Acesso em: 10 abr 2005.

ZENDULKA J. Object-Relational Modelling in UML. In 4th International Conference on Information Systems Modelling (ISM), 2001.

ANEXOS

ANEXO A - Estudos de Benchmark do Caché

ANEXO B - Banco de Dados Caché

ANEXO A

Estudos de Benchmark do Caché

Este anexo relaciona três estudos de caso de benchmark do Caché com banco de dados relacional.

Estudo de Caso 01

BENCHMARK NO MUNDO REAL: CACHÉ VS. ORACLE EM UMA APLICAÇÃO DE DATA WAREHOUSE
White Paper técnico de Mark Ramsay
Senior Sales Engineer
InterSystems Corporation
Artigo disponível em < http://www.intersystems.com.br/isc/downloads/wp/CacheOracle.pdf>.
Acesso em: 10 ago 2005.

Este artigo relata o benchmark realizado com uma aplicação real de data warehouse baseada em Oracle 8i. O teste mediu o tempo necessário para completar determinadas tarefas rodando o módulo baseado em Caché, comparado ao tempo no Oracle.
O teste foi realizado em uma companhia de fornecimento de energia nas Filipinas, que usa o Oracle para seu data warehouse corporativo. Foi replicado o módulo "Field Order" da aplicação ETL (Extract, Transform, Load) do seu DW em um sistema baseado em Caché, a fim de testar o desempenho do Caché em comparação com o Oracle.
O módulo "Field Order", baseado no Oracle 8i, foi projetado para colher entradas de arquivos de texto gerados no campo e produzir duas tabelas – uma tabela de dimensões do *field order* e uma tabela de fatos do *field order* – que subseqüentemente são usadas para análise de dados e relatório. Para realizar esta tarefa, o módulo executa as seguintes operações: Os múltiplos arquivos texto (ordenes.txt e gcahorde.txt) gerados no campo são carregados em duas tabelas preparatórias, ORDENES e GCAHORDE. Na aplicação baseada em Oracle, isto é feito usando o Oracle SQL*Loader. As tabelas de dimensão e fatos são geradas usando pacotes do Oracle e *stored procedures*.
Foram replicadas as funcionalidades do módulo Field Order no Caché ObjectScript.
As versões do módulo Field Order em Caché e em Oracle 8i rodaram usando o mesmo conjunto de arquivos de dados em texto. Foi medido o tempo que cada versão levou para completar as várias tarefas.
Dependendo da tarefa, os Run-Times do Caché foram de quatro a vinte e três vezes mais rápidos do que o Oracle. Os resultados estão resumidos a seguir, no quadro A1:

Tarefas	Oracle		Caché		Tempo Decorrido (Oracle : Caché)
	Tempo Decorrido	Nº de registros	Tempo Decorrido	Nº de registros	
Carregar ORDENES	138 min.	927.857	6 min.	927.857	23 : 1
Carregar e Filtrar GCAHORDE	245 min.	90.349 *	23 min.	90.349	10 : 1
Gerar a tabela de dimensões FO	168 min.	1.018.216	24 min.	1.018.216	7 : 1
Gerar a tabela de fatos FO	890 min.	1.018.216	233 min.	1.018.216	4 : 1
Tempo total para carregar e filtrar dados	1.441 min.		286 min.		5 : 1

Quadro A1 – carga de dados no Caché e Oracle

* Foram filtrados 13.396.510 registros fonte, que produziram 90.349 registros carregados.

Neste benchmark de um caso real de aplicação de data warehousing, o Caché foi 5 vezes mais rápido que o Oracle na execução das operações de carga e filtragem de dados. Foram requeridas 40 horas para replicar a aplicação original baseada em Oracle para Caché.

Estudo de Caso 02

ESTUDO DE CASOS ASPECTOS RELATIVOS A PERFORMANCE
White Paper técnico de Mary Finn
Product Marketing Manager
InterSystems Corporation
Artigo disponível em
<http://www.intersystems.com.br/isc/downloads/wp/PerformanceMaryFinn_Portugues.pdf>.
Acesso em 10 ago 2005.

Segundo o artigo, a melhor maneira de comparar o desempenho de bancos de dados é criar um teste real usando uma aplicação real, preferivelmente uma de desenvolvimento próprio.
Muitas companhias executam comparações de Caché e outros bancos de dados. Esta análise apresenta o resultado de benchmarks de desempenho obtido por analistas independentes e por clientes e prospects da InterSystems quando eles avaliaram suas opções em tecnologia de bancos de dados. Por causa das restrições contratuais impostas por grande parte dos players de banco de dados do mercado, os nomes das companhias que conduziram os benchmarks em desempenho foram omitidos, assim como os nomes dos bancos de dados relacionais que competem com Caché.
Estes testes foram executados usando aplicações e dados reais sob condições reais. Usando SQL para consultar os bancos de dados, o Caché constantemente supera Oracle, SQL Server e outros bancos de dados relacionais por cinco ou mais fatores.
Na maioria dos estudos apresentados, SQL foi utilizado como a linguagem de consulta porque facilita a condução de um por um dos testes de Caché e dos bancos de dados relacionais. Nestes casos, o desempenho superior de Caché é devido a sua arquitetura multidimensional, que elimina as despesas necessárias de processamentos para executar *joins* por meio de tabelas múltiplas. Entretanto, SQL não é o único método disponível de consulta no banco de dados Caché.
O artigo relaciona como estudo de caso nº 1 o teste apresentado no artigo imendiatamente anterior, o white paper de Mark Ramsay, engenheiro sênior da InterSystems Corporation, intitulado "BENCHMARK NO MUNDO REAL: CACHÉ VS. ORACLE EM UMA APLICAÇÃO DE DATA WAREHOUSE".
O estudo de caso nº 2 foi realizado em um provedor de serviços de saúde comparando o tempo de respostas SQL do Caché com o de um banco de dados relacional bem conhecido. Usando dados históricos de pacientes (7 tabelas com mais de 6,5 milhões de registros), rodaram uma bateria de 8 consultas com carga simulada de 30, 60, 90 e 120 usuários concorrentes.
O quadro A2 dá os tempos de resposta médios em milissegundos para os dois bancos de dados. Ambos eram rápidos, mas o Caché conseguiu ser de 4 a 6 vezes mais rápido.

Nº de usuários concorrentes	Tempo Médio de Resposta (para todas as oito consultas)		Desempenho Relativo
	RDBMS	Caché	Caché: RDBMS
30	375.125 ms	59.125 ms	6.3 : 1
60	637.25 ms	137.75 ms	4.6 : 1
90	915.625 ms	206.875 ms	4.2 : 1
120	1146.375 ms	290.125 ms	3.9 : 1

Quadro A2 – tempo de resposta com usuários concorrentes.

Adicionalmente, o gráfico A1 dos resultados mostra que o Caché é mais escalonável. Por exemplo, quando a carga é ampliada de 90 para 120 usuários, o aumento do tempo de resposta de Caché é de apenas 84 ms, enquanto o banco de dados relacional reduz sua velocidade em 231 ms – 2,7 vezes mais lento.

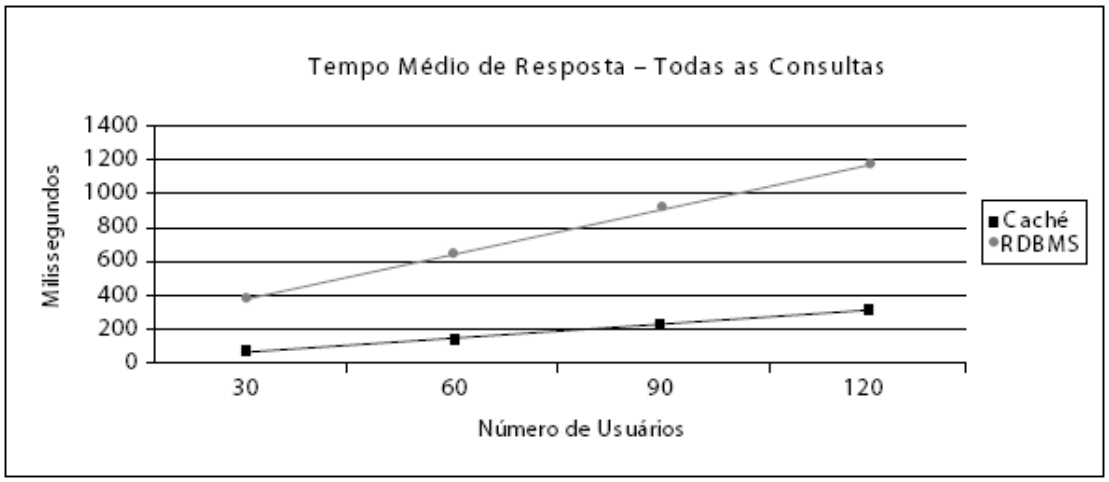

Gráfico A1 – tempo médio de resposta entre Caché e RDBMS.

No estudo de caso nº 3 um consultor comparou o Caché e um banco de dados relacional bem conhecido, para medir seu desemepnho em um cenário de data warehousing. Usando uma tabela com um milhão de registros contendo dados reais de processamentos de ordens, ele mediu os tempos de execução da validação/carga, bem como os tempos de resposta às consultas.
Embora a estrutura de dados do Caché não tivesse sido otimizada para a tarefa, o tempo de validação/carga para 1.000.000 de registros foi três vezes mais rápida para o Caché em comparação ao RDBMS (2.681 segundos vs. 8.597 segundos). Além do mais o banco de dados Caché resultante precisou de apenas 146 MB de espaço em disco enquanto o RDBMS precisou de 216 MB.
Os resultados do teste de resposta à consulta SQL são mostrados no quadro A3.

Consulta	Tempo de Resposta		Desempenho Relativo
	RDBMS	Caché	Caché: RDBMS
Agregação ampla de toda a tabela com todas as dimensões	638 s	180 s	3.5 : 1
Agregação de todas as dimensões com restrição em 2 dimensões	3 s	<1 s	>3 : 1
Localização de "purpose codes" não utilizados em toda a tabela	8 s	3 s	4 : 1
Localização de todos os registros nos quais uma dimensão é maior do que um número específico	50 s	16 s	3.1 : 1

Quadro A3 – teste de desempenho entre Caché e RDBMS.

O estudo de caso nº 4 relata o teste em uma companhia telefônica, usando sua aplicação on-line de catálogo telefônico para testar o desempenho, em que o Caché operava, em média, 10 vezes mais rapidamente do que um banco de dados muito conhecido.
Carregar o banco de dados relacional de arquivos de texto levou, aproximadamente, 10 horas e necessitou de 60 GB de espaço em disco. Em comparação, o banco de dados Caché necessitou de, aproximadamente, 45 minutos e 2GB de espaço em disco. Como o banco de dados Caché foi mais eficiente no uso de espaço, ele fez uso muito melhor da

memória cache. Como resultado, o tempo de resposta à consulta foi mais rápido para o Caché em relação ao RDBMS, conforme o quadro A4:

Consulta	Tempo de Resposta		Desempenho Relativo
	RDBMS	Caché	Caché: RDBMS
Busca por nome, rua	0.33 s	0.07 s	4 : 1
Busca por nome	0.46 s	0.07 s	23 : 1
Busca por nome, rua, CEP	0.76 s	0.04 s	19 : 1
Busca pelo número da casa	0.14 s	0.02 s	7 : 1
Busca por nome1, nome2, rua, CEP	0.01 s	0.01 s	1 : 1
Busca por país	0.02 s	0.01 s	2 : 1
Média de todas as queries	0.287 s	0.028 s	10 : 1

Quadro A4 – teste de desemepnho para consultas entre Caché e RDBMS.

Foram usados índices nos dois bancos de dados para aumentar o desempenho da consulta, e ainda que o RDBMS fosse mais lento que o Caché, ele também forneceu tempos de resposta de fração de segundo. No entanto, as atualizações do RDBMS (e a conseqüente necessidade de reconstruir os índices) precisam ser feitas off-line para evitar problemas significativos de desempenho. A aplicação foi efetivamente projetada para usar dois bancos de dados relacionais – um "vivo" enquanto o outro está sendo atualizado – trocando-os a cada dia. No caso do Caché, ao contrário, o banco de dados "vivo" pode ser atualizado sem degradação significativa do desempenho. Adicionalmente, o Caché permite fazer buscas com a informação "começando por" que o RDBMS não permite.

O estudo de caso nº 5 foi realizado em uma das mais importantes empresas fornecedoras de aplicações para a área de saúde nos USA. Eles fizeram um teste de escalabilidade de sua suite de aplicações EpicCare para um grande cliente que está planejando ampliar seu sistema para suportar 8.000 usuários concorrentes. O teste foi conduzido em um servidor HP Superdome com 24 processadores e 8 GB de memória, rodando a versão 5.0 do Caché e a versão completa do produto de software da EpicCare. Os bancos de dados foram preenchidos com dados históricos realistas (mas anônimos), e o teste de stress simulou padrões reais de uso da aplicação para o cliente usuário final.
O sistema excedeu amplamente as expectativas, suportando 14.600 usuários concorrentes e sustentando 540.000 acessos ao banco de dados por segundo, enquanto se mantinha dentro dos limites previstos para tempo de resposta e uso da CPU.

Estudo de Caso 03

TESTE DE PERFORMANCE COM O ÍNDICE BIT MAP
CACHÉ VERSUS ORACLE
Artigo disponível em < http://www.intersystems.com.br/>. Acesso em: 10 ago 2005.

Embora a indexação bit map seja usada há muito tempo para acelerar a recuperação em sistemas de data warehouse, seu baixo desempenho de atualização tem impedido seu uso em aplicações que processam transações. Para abordar a crescente necessidade de Análise em Tempo Real, que demanda acesso de queries complexas a dados vivos, o Caché 5 introduz a nova tecnologia Transactional Bit Map Indexing que permite um desempenho de atualização radicalmente mais rápida. Esta inovação exclusiva permite que a indexação bit map seja usada em aplicações que processam transações de desempenho crítico.
Para demonstrar a melhora de desempenho de atualização de bit map do Caché 5, a InterSystems realizou os testes descritos a seguir.

Estes testes usam uma tabela contendo nove colunas, definidas como:

```
Create tabela Person ( ID INTEGER Primary Key,
        Name VARCHAR(50),
        Title VARCHAR(50),
        Sex VARCHAR(1),
        State VARCHAR(2),
        Salary INTEGER,
        Age INTEGER,
        Code VARCHAR(10),
        HairColor VARCHAR(10))
```

Foram criados índices Bit map para sete colunas - todas menos ID que é única e Name que é quase única. (Os índices bit map geralmente não são recomendados para colunas que têm um número muito grande de valores únicos.) Os índices foram definidos usando a seguinte sintaxe SQL:

```
create bitmap index ITitle on Person(Title)
```

Carga do Banco de Dados

O banco de dados foi preenchido com 10.000.000 de linhas. Como mostra o quadro A5, a seletividade da coluna varia desde muitos valores distintos (por exemplo, Name) para apenas uns poucos valores distintos (por exemplo, Sex).

Coluna	**Valores Distintos**
Name	711.253
Title	136
Sex	2
State	50
Salary	99
Age	98
Code	6
HairColor	5

Quadro A5 - seletividade das colunas de teste.

O quadro A6 mostra a configuração do sistema utilizado nos testes.

CPU	Dual 2.0 GHz Intel Pentium 3 Xeon com 512KB cache
Memória	1 GB RDRAM
Disco	Dual 18.4 GB Seagate Cheetah SCSI
Sistema Operacional	Windows 2000 Server (SP 2 Build 2195)
Banco de Dados	Caché Oracle 9i Enterprise Edition

Quadro A6 – configuração do sistema.

Foi usado um comando simples, SQL UPDATE, para testar o desempenho de atualização:

```
Update Person Set Code = 'A' where ID= :id
```

Este comando foi executado 1000 times (com um procedimento armazenado para eliminar questões referentes à latência de cliente/servidor), modificando linhas espalhadas entre as 10.000.000 de linhas da tabela.

Como mostram os resultados do quadro A7, o Caché executa as atualizações cerca de 300 vezes mais depressa que o Oracle.

	Tempo Decorrido para Atualizar 1000 Linhas	Linhas Atualizadas por Segundo
Caché	0,11 segundos	8787
Oracle	34,01 segundos	29

Quadro A7 – atualizações no Caché e Oracle.

Para confirmar que esta diferença foi devida ao uso de indexação bit map, rodamos também testes de atualização das tabelas Caché e Oracle sem índices. Com base na relação destes

resultados (índice bit map index:sem índice) foi calculado o "custo" do tempo de atualização da indexação bit map, conforme mostra o quadro A8. A adição de um índice bit map ao Caché aumentou o tempo de atualização do índice de um pouco menos do que 50%. Fazendo a mesma coisa com o Oracle o tempo de atualização aumentou 6800 vezes.

	Custo da Indexação Bit Map
Caché	1,45
Oracle	6800

Quadro A8 – custo de indexação no Caché e Oracle.

Desemepnho de Recuperação de Dados

Para assegurar que a vantagem no desempenho da atualização do Caché não ocorre às custas do desemepnho de recuperação, foram executados também uma série de testes de recuperação. De modo geral, o desempenho de recuperação foi semelhante, com o Caché comportando-se melhor em cinco casos e o Oracle em um caso, conforme o quadro A9.

		Tempo Decorrido (s)	
Query	**Nº de Linhas Encontradas**	**Tempo do Caché**	**Tempo do Oracle**
Select count(*) from Person where State = 'MA'	200.108	0,0028	0,0050
Select count(*) from Person where Sex = 'M'	4.998.018	0,0056	0,0100
Select count(*) from Person where Age < 40	3.038.594	0,0367	0,0400
Select count(*) from Person where Age between 10 and 40	3.184.366	0,0366	0,0300
Select Name from Person where State = 'MA' and Age = 50	4.448	13,6239	14,0200
Select Name from Person where (Age between 5 and 6 or State = 'MA') and HairColor='Red'	42.562	17,8577	18,0100

Quadro A9 – testes de recuperação de dados no Caché e Oracle.

ANEXO B

Banco de Dados Caché

INTRODUÇÃO

Caché e InterSystems são marcas registradas da InterSystems Corporation.
Caché é um banco de dados pós-relacional que oferece três opções integradas de acesso que podem ser usadas simultaneamente pelo mesmo dado: um banco de dados orientado a objetos, desempenho com SQL e acesso multidimensional. O processo de mapeamento não é necessário entre acessos de views de dados objetos, relacionais e multidimensionais, resultando em economia tanto em tempo de desenvolvimento como de processamento.
O principal produto da InterSystems, o Caché é um banco de dados pós-relacional que atua como servidor de dados multidimensional e/ou um servidor de aplicações.

HISTÓRICO DO CACHÉ

O desenvolvimento do Caché começou em 1995 como um produto sucessor da família Intersystems baseado em banco de dados ANSI-Standard M. O objetivo principal era de criar um produto com banco de dados de alto desempenho e com rápido desenvolvimento de aplicações. O quadro A10 apresenta as características de cada uma de suas versões.

Versão	Ano	Características
2.0	1997	Distributed Cache Protocol Suporte ao Windows Suporte ao UNIX
2.1	1997	Caché Objects ActiveX binding ODBC Suporte ao OpenVMS
3.0	1998	Suporte ao Native UNICODE Suporte ao Localization
3.1	1999	Unified Data Architecture Suporte ao Native object Caché Explorer Caché Control Panel Caché Configuration Manager
3.2	2000	JDBC Java Binding Suporte a Linux Caché SQL Manager Caché Studio
4.0	2000	Caché Server Pages Incremento do CachéObjectScript Caché SQL Gateway Class Packages
4.1	2001	Novo Caché database engine Novo Lock Manager
5.0	2002	Integração do Caché Studio Sistema de documentação Online Suporte a ECP Distributed Database Suporte a Bitmap Index Suporte a XML Suporte a SOAP e Web Services Suporte a Java e EJB Suporte a C++ Caché Activate

Quadro A10 – características e versões do Caché.

ARQUITETURA CACHÉ

As marcas registradas do Caché são: alto desempenho, escalabilidade maciça, desenvolvimento rápido de aplicações e custo-benefício. Estes valores são refletidos na arquitetura básica do Caché.

O Caché armazena dados de uma forma multidimensional, assegurando desempenho, mesmo sobre pesados arquivos rodando em um hardware significantemente menor, dferente de outros bancos de dados. Além disso, o Caché deixa seus dados acessíveis por meio de uma ampla variedade de tecnologias.

O Caché inclui um Servidor de Aplicações com uma capacidade de programação em objeto, a habilidade de integrar facilmente com uma ampla variedade de tecnologias e um ambiente de desempenho com única tecnologia de caching. A figura A1 mostra a arquitetura do Caché.

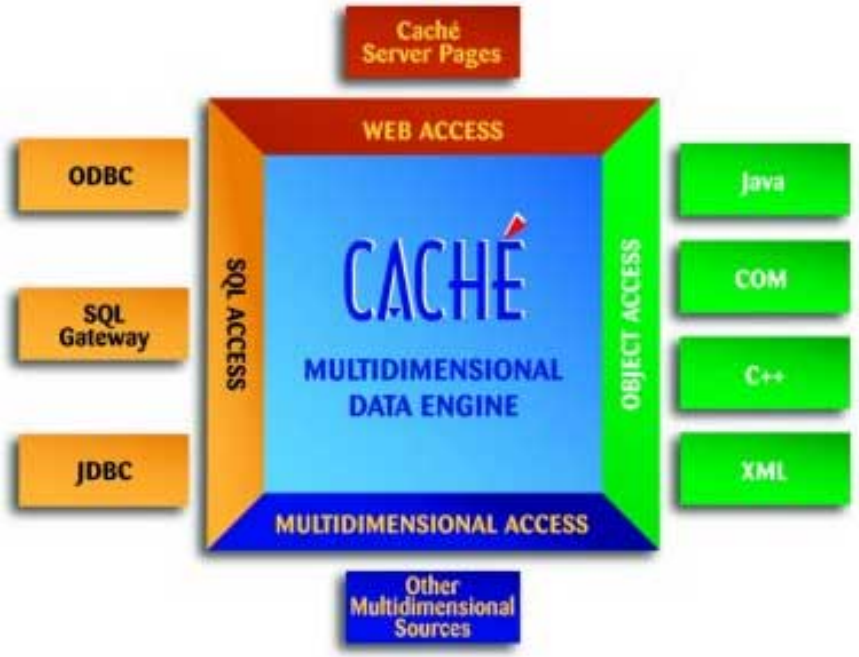

Figura A1 – Arquitetura do Caché.

O Caché também possui características de bancos de dados tradicionais por incorporar um ambiente rico para desenvolvimento de aplicações baseadas em browser (Web). A tecnologia do Caché Server Pages (CSP) permite desenvolvimento e execução de páginas Web geradas dinamicamente. Milhares de usuários Web simultâneos podem acessar as aplicações de bancos de dados, até com hardwares de baixo custo.

Para aplicações não baseadas em browser, a *interface* com o usuário é tipicamente programada em uma das tecnologias de *interface*, como VB, Delphi, Java, ou C++. Os melhores resultados (rápida programação, desempenho e baixa manutenção) são geralmente obtidos executando todo o resto do desenvolvimento com Caché. Entretanto, o Caché também fornece níveis de interoperabilidade com outras tecnologias e suporta as ferramentas de desenvolvimento mais comuns.

O Caché é desenvolvido em ANSI C com otimizações em Assembly para cada uma das plataformas suportadas. Isto o torna leve e pouco exigente com relação ao hardware utilizado. O IDE que acompanha o Caché é específico para o desenvolvimento das classes persistentes no banco, Caché Server Pages para a Web e rotinas internas.

PLATAFORMAS E SO

O Caché 5.0 suporta as seguintes plataformas e sistemas operacionais definidas no quadro A11.

Plataforma	Sistema operacional	SQL Gateway
Apple	Mac OS X 10.3, 10.4	Não
HP Alpha	OpenVMS 7.2-2, 7.3-1, 7.3-2	Não
HP Alpha	Tru64 UNIX 5.1, 5.1A, 5.1B	Não
HP (PA-RISC)	HP-UX 11, 11i v1, 11i v2	Sim
HP (Itanium)	HP-UX 11i v2	Sim
IBM pSeries	AIX 4.3.3, 5.1, 5.2, 5.3	Sim
Red Hat Enterprise Linux AS (Intel 32-bit)	3, 4	Sim
Red Hat Enterprise Linux AS (64-bit)	3, 4	Sim
Sun Solaris (SPARC)	8, 9, 10	Sim
SuSE Linux (Intel 32-bit)	7.3, 8.0, 9.0	Sim
SuSE Linux Enterprise Server (Intel 32-bit)	8, 9	Sim
SuSE Linux Enterprise Server (64-bit)	8, 9	Sim
Microsoft Windows (32-bit)	NT 4 (SP4, SP5, SP6), 2000 (SP3, SP4), XP (SP1, SP2), Server 2003 (SP1)	Sim
Microsoft Windows (64-bit)	Server 2003 (SP1)	Sim

Quadro A11 - plataformas e sistemas operacionais que o Caché suporta.

Requisitos de instalação no Windows

Uma instalação padrão do Caché no Windows que inclui suporte a Caché Server Pages (CSP) usa aproximadamente de 173–187 MB de espaço em disco. Deve-se ter 10 MB de espaço adicional para a instalação.

Plataformas e Web Servers: Windows Server 2003, Windows XP (SP1 ou SP2), Windows 2000 (SP3 ou SP4) e Windows NT 4.0 with SP4, SP5 ou SP6).

O Caché é suportado apenas como cliente nas plataformas Windows ME, Windows 98 e Windows 95.

MECANISMO MULTIDIMENSIONAL DE DADOS

Diferente de bancos de dados relacionais que forçam os dados em tabelas dimensionais, o Caché armazena o dados em tabelas multidimensionais. Além de permitir modelagens realistas de dados, tabelas multidimensionais são mais rápidas de acessar, pois eliminam os custos fixos associados com os table-hopping e joins que exemplificam a tecnologia relacional. Outro fator de aumento de desempenho é o Distributed Cache Protocol do Caché, que reduz o tráfico de trabalho em sistemas distribuídos.

Embora os dados estejam armazenados de forma multidimensional, o Caché dá ao desenvolvedor a liberdade para modelar seus dados da forma que escolher: como objeto, como tabela ou como matriz multidimensional. O Caché vem com uma *interface* gráfica para criar Caché Objects. Ele também pode aceitar dados vindos do Rational Rose e arquivos DDL.

Pela eficiência da Arquitetura Unificada de Dados do Caché, todos os dados são automaticamente acessados tanto como objetos como tabelas. Não existe a necessidade de mapear uma forma para a outra, e nenhum custo fixo é necessário para converter formas. A Arquitetura Unificada de Dados aumenta tanto a produtividade quanto ao desempenho.

O Caché também permite escolhas quando o banco de dados fica com o *script* da lógica de negócios. O CachéObjectScript (COS) suporta todos os métodos de acessos a dados: objetos, SQL, multidimensional e HTML embutido. O Caché Basic é similar ao Visual Basic, com poucas modificações para se tirar todas as vantagens únicas das capacidades do Caché.

Acesso Web

Na arquitetura única Web do Caché, o Caché Server Pages executa no servidor de dados, o dado que ele precisa acessar. Esta abordagem aumenta o desempenho, além de aumentar consideravelmente a escalabilidade, tirando muito do processo de carga do servidor Web, liberando-o para lidar com mais pedidos do browser.

O Caché aplica o poder do rápido desenvolvimento da tecnologia objeto para a criação do Caché Server Pages. Cada Caché Server Page é um objeto, e pode melhorar o

comportamento de gerenciamento de sessões (de vários níveis de segurança) de um sistema de objetos fornecido pela InterSystems. A herança do objeto é também uma forma rápida para assegurar um visual consistente por todas as páginas da aplicação.
Usando ferramentas web pode-se adicionar funcionalidades às páginas, incorporando os Caché Application Tags (CATs) da mesma forma que eles usariam qualquer HTML tag padrão. Para algumas funções padrões os CATs vêm com o Caché, ou podem ser construídos conforme a necessidade.
Com Caché Server Pages desenvolver complexas aplicações web fica mais simples e rápido, porque as aplicações web podem ser programadas usando ferramentas já conhecidas e com as quais o desenvolvedor já está familiarizado. É possível criar páginas CSP (Caché Server Pages) usando a ferramenta Caché para definição de classes, usando ferramentas de mercado para a criação de páginas web ou um editor de texto.
As páginas CSP herdam todo o código necessário para o gerenciamento de sessão. O código de programação que é derivado dos objetos de sistema fornecidos pela InterSystems permite ao desenvolvedor escolher o nível de segurança desejado para a sessão, e o Caché se encarrega do resto.
É possível adicionar funcionalidade às páginas por meio dos tags de aplicação CSP. Podem ser usados os tags que vêm com o Caché, ou podem ser desenvolvidos novos tags personalizados para necessidades específicas.

Caché e XML

Além de todas as suas características de conectividade, o Caché combina também com XML. É possível usar objetos Caché como uma representação direta de documentos XML e vice-versa. O Caché oferece os seguintes recursos:

- objetos que podem ser automaticamente convertidos em documentos XML que, por sua vez, podem ser usados como arquivos ou como conteúdo on-line. As classes Caché também podem criar automaticamente seus próprios arquivos XML DTD (Document Type Definition).
- documentos XML podem ser transformados automaticamente em objetos Caché equivalentes. Conteúdos XML podem ser recebidos a partir de arquivos, streams ou requisições HTTP, e também ser validados usando XML DTD padrão.

O suporte a XML do Caché pode ser adaptado às necessidades especiais das aplicações desenvolvidas.

Acesso a Objetos

O Caché suporta uma vasta gama de técnicas de modelagem de objetos, incluindo heranças múltiplas, encapsulamento, polimorfismo, referências, coleções, relações e BLOBs. Os objetos Caché podem ser criados com a Arquitetura de Objetos do Caché ou por meio de um *link* bi-direcional do Caché para o Rational Rose. Diferente de um sistema de banco de dados objeto-relacional, o Caché permite evolução do esquema de dados, para que as definições de objetos possam ser alteradas para atender mudanças nas aplicações. Devido a Arquitetura Unificada de Dados do Caché, todos os Objetos Caché são automaticamente compatíveis com ODBC. Os objetos Caché são também compatíveis com tecnologias e ferramentas orientadas a objetos. Eles podem ser usados por desenvolvedores Java e C++ e por ferramentas (como Visual Basic e Delphi) que usam a *interface* COM. O Caché também vem com uma *interface* CORBA bi-direcional.

Acesso SQL

Muitas aplicações de software, particularmente aquelas para relatórios de dados e análises, usam SQL como linguagem de consulta e pedem um banco de dados concordante ODBC ou JDBC. Via o acesso de dados SQL, o Caché está disponível para todas estas aplicações. Além disso, o Caché SQL Gateway permite que as aplicações Caché acessem dados armazenados em bancos de dados relacionais - útil quando não existe nenhuma necessidade de integrar dados de uma variedade de fontes.
Alguns desenvolvedores podem querer migrar aplicações de um banco de dados relacional para Caché para ter vantagens do desempenho do Caché e da tecnologia de orientação a objetos. O Caché pode criar uma estrutura de dados de definições de uma tabela relacional contida em arquivos DDL. Por virtude da Arquitetura Unificada de Dados do Caché, cada

tabela definida se transforma em um objeto simples que pode ser usado como está, ou como blocos construídos de estruturas mais complexas. Então, usando o SQL Gateway, os dados podem ser transferidos de um banco de dados relacional para o Caché.
Os dados podem ser usados na forma de objetos, SQL, ou como acesso direto a estruturas multidimensionais. Independentemente do método de acesso, todos os dados na base de dados do Caché são armazenados nos arrays multidimensionais do Caché.
Uma vez armazenados os dados, todos os três métodos de acesso podem ser usados simultaneamente sobre os mesmos dados com plena concorrência. Sempre que uma classe de objetos de banco de dados é definida, o Caché gera, automaticamente, uma descrição relacional pronta para SQL daqueles dados. Desta forma, se uma descrição DDL de um banco de dados relacional é importada para o Dicionário de Dados, o Caché gera, automaticamente, ambas as descrições, tanto a descrição relacional como a descrição dos dados por objetos, permitindo seu acesso imediato como objeto. O Caché mantém estas descrições coordenadas, existindo apenas uma definição de dados a editar. O programador pode editar e ver o dicionário tanto da perspectiva de objetos como na forma de uma tabela relacional.
O Caché cria automaticamente o mapeamento de como são armazenados os objetos e as tabelas nas estruturas multidimensionais, ou o programador pode controlar o mapeamento explicitamente.

Escalabilidade multidimensional

O Caché armazena dados em vetores esparsos, o que torna seu servidor de dados compacto, capaz de armazenar eficientemente, no mesmo espaço em disco, mais informações que outros bancos de dados relacionais. Esta eficiência resulta não só em desempenho superior do Caché, como também em um menor investimento em recursos de hardware.
Descongestionamento do tráfego na rede é outra forma que o Caché encontrou de melhorar o desempenho e a escalabilidade em sistemas distribuídos através de sua abordagem única de cache de dados. Ao tirar vantagem da natureza multidimensional dos dados, o Caché reduz drasticamente o tráfego na rede. Menos tráfego significa que as aplicações ganham velocidade e podem atender mais usuários concorrentes.

Servidor de Dados Caché

Quando se trata do processamento de transações, o desempenho é crítico. A tecnologia do servidor de dados do Caché permite levar suas aplicações até dezenas de milhares de usuários sem comprometer o desempenho. O quadro A12 apresenta as características da arquitetura Caché.

MODELAGEM DE DADOS

Como os objetos podem modelar dados complexos de forma simples, a programação por objetos é a melhor escolha para programar aplicações complexas. O Caché complementa o acesso a objeto com uma linguagem de consulta SQL estendida a objetos.

Modelo de dados por objetos e a programação de objetos

O modelo de objeto do Caché é baseado no padrão ODMG (Object Database Management Group) e suporta muitas características avançadas, inclusive herança múltipla.
Diferentemente das tabelas relacionais, os objetos reúnem dados e código. Por exemplo, um objeto Invoice (Fatura) poderia ter dados, como o número de fatura e o valor total, e um código tal como Print (Imprimir).
Conceitualmente, um objeto é um pacote que inclui os valores dos dados daquele objeto (propriedades) e uma cópia de todo o seu código (métodos). Os métodos de um objeto enviam mensagens para se comunicarem com outros métodos. Para reduzir o armazenamento, é comum que objetos de uma mesma classe compartilhem uma mesma cópia de código. Além disso, no Caché, tipicamente, as chamadas de métodos resultam em eficientes chamadas de funções em lugar de suportar a sobrecarga de passar mensagens. Porém, como estas técnicas de implementação ficam ocultas do programador, é sempre acertado pensar em termos de objetos passando mensagens.

Característica	Descrição	Vantagens

Data Engine Multidimensional	Todos os dados são armazenados em arrays multidimensionais esparsos que eliminam a sobrecarga do processamento relacionado à ligação de dados que comumente ocorre nos bancos de dados relacionais.	Alto desempenho. Escalabilidade maciça. Modelagem realista de dados complexos. Armazenamento de dados eficiente que consome menos espaço em disco e requer menos hardware.
Acesso a dados objetos	Os dados podem ser modelados como objetos. O Caché suporta encapsulamento, heranças múltiplas, polimorfismo, objetos embutidos, referências, coleções, relações e BLOBs.	Rápido desenvolvimento de aplicações. Modelagem intuitiva de dados complexos.
Acesso a Dados SQL	Permite acesso relacional à base de dados Caché. Suporta tanto ODBC como JDBC.	Aumenta o desempenho de aplicações relacionais legadas. Fornece conectividade SQL a consultas-padrão, relatórios e ferramentas de análise.
Acesso a Dados Multidimensionais	Fornece controle direto de estruturas multidimensionais no Banco de Dados Caché.	Alto desempenho. Permite conectividade com sistemas legados.
Arquitetura Unificada de Dados	Classes de objetos e tabelas relacionais são automaticamente geradas a partir de uma única definição de dados.	Rápido desenvolvimento. Elimina problemas de impedância entre objetos e tabelas.
Indexação Bit-Map Transacional	Os índices bit-map do Caché podem ser atualizados de forma extremamente rápida, tornando-os adequados para uso com dados em tempo real.	Resposta rápida a consultas complexas. Rápida atualização que permite efetuar análise de dados em tempo real enquanto mantém alto desempenho de processamento das transações.
API para Monitoramento do Desempenho	Conecta-se com ferramentas de monitoramento de mercado como o Patrol da BMC e o Sightline da Fortel.	Ajuda na otimização de aplicações. Fornece um método para demonstrar o atendimento de especificações de desempenho.

Quadro A12 - características da arquitetura Caché.

A tecnologia de objetos também promove uma visão natural dos dados sem restringir suas propriedades a tipos de dados simples, próprios de computadores. Os objetos podem conter outros objetos, ou referências a outros objetos, o que facilita construir modelos de dados úteis e significativos.

Modelo de dados multidimensionais

Em seu núcleo, o banco de dados Caché é movido por um data engine multidimensional. O acesso direto às estruturas multidimensionais é fornecido através de linguagens *script* Caché incorporadas – que fornecem alto desempenho e várias possibilidades de armazenamento – e muitas aplicações são inteiramente implementadas usando diretamente este motor de dados. O uso do acesso global direto aos dados é particularmente comum quando há estruturas incomuns ou muito especializadas e não há necessidade de fornecer acesso SQL ou como objeto às mesmas, ou onde é requerido o mais alto desempenho possível.

Os arrays multidimensionais do Caché são chamados globais. Os dados podem ser armazenados em uma global com qualquer número de subscritos. O que é mais importante: os subscritos não precisam de informação de tipo e podem conter qualquer tipo de dados. Um subscrito poderia ser um inteiro, enquanto outro pudesse ser um nome significativo, ainda que no mesmo nível de subscritos.

Freqüentemente, apenas um único elemento de dados é armazenado em um nodo de dados, como uma data ou quantidade, mas às vezes é útil armazenar múltiplos elementos de dados juntos em um único nodo de dados. Isto é particularmente útil quando há um

conjunto de dados relacionados que são freqüentemente acessados juntos. Também se pode melhorar o desempenho requerendo menos acessos ao banco de dados.
Para simplificar este tratamento, o Caché suporta uma função chamada $list() que pode montar múltiplos elementos de dados em bytes tipo string, de comprimento delimitado, e depois os desmontar, preservando seu datatype.
Devido ao fato de os dados do Caché terem comprimento variável, por construção, e serem armazenado em arrays esparsos, o Caché freqüentemente requer menos da metade do espaço requerido por um banco de dados relacional. Além de reduzir as exigências de disco, o armazenamento compacto dos dados aumenta o desempenho porque mais dados podem ser lidos ou podem ser escritos com uma única operação de I/O e o cache dos dados pode ser mais eficiente.
Os arrays multidimensionais do Caché, por construção, não têm tipo, sejam eles relativos a dados ou aos subscritos. Não é requerida nenhuma declaração, definição, ou alocação de armazenamento. Os dados da global passam a existir simplesmente quando é feita a inserção dos dados.
No Caché, dados e códigos são armazenados em arquivos de disco com o nome CACHE.DAT (somente um por diretório). Cada um destes arquivos contém numerosas globais (arrays multidimensionais). Dentro de um arquivo, o nome de cada global deve ser único, mas diferentes arquivos podem conter o mesmo nome de global. Estes arquivos podem ser livremente imaginados como bancos de dados.
Em vez de especificar qual arquivo de banco de dados usar, cada processo Caché usa um namespace para acessar os dados. Um namespace é um mapa lógico que traça os nomes dos arrays de globais multidimensionais e o código para bancos de dados. Se um banco de dados é movido de uma unidade de disco ou computador para outro, somente o mapa do namespace precisa ser atualizado. A própria aplicação permanece inalterada.
Normalmente, exceto por alguma informação de sistema, todo os dados para um namespace são armazenados em um único banco de dados. No entanto, os namespaces fornecem uma estrutura flexível que permite mapeamento arbitrário e não é incomum que um namespace mapeie o conteúdo de vários bancos de dados, incluindo alguns de outros computadores.

Acesso SQL

O SQL é a linguagem de consulta do Caché, e suporta um conjunto completo de capacidades dos bancos de dados relacionais – inclusive DDL, transações, integridade referencial, gatilhos, procedimentos armazenados e várias outras. O Caché suporta acesso por ODBC e JDBC (usando um driver baseado em Java puro). Os comandos e consultas do SQL também podem ser incorporados ao Caché ObjectScript e nos métodos de objetos.
O SQL acessa dados vistos como tabelas com linhas e colunas. Como os dados do Caché são, de fato, armazenados em estruturas multidimensionais, as aplicações que usam SQL obtêm desempenho melhor com o Caché do que quando rodam em bancos de dados relacionais tradicionais.
O Caché suporta, além da sintaxe SQL padrão, muitas das extensões geralmente usadas em outros bancos de dados, de forma que muitas aplicações baseadas em SQL podem rodar no Caché sem modificações – especialmente as escritas com ferramentas que independem de bancos de dados.
O Caché SQL inclui aprimoramentos relativos a objetos que tornam a codificação do SQL mais simples e sua leitura e escrita mais intuitiva, conforme o quadro A13.

Acessando Bancos de Dados Relacionais com o SQL Gateway Relacional do Caché

O Gateway Relacional do Caché permite que um pedido SQL que se origina no Caché seja enviado para outros bancos de dados (relacionais) para ser processado.
Usando o Gateway, uma aplicação Caché pode recuperar e atualizar dados armazenados na maioria dos bancos de dados relacionais.
Adicionalmente, se as classes do banco de dados Caché são compiladas usando a opção de CachéSQLStorage, o Gateway permite que as aplicações Caché usem,

transparentemente, bancos de dados relacionais. Porém, estas aplicações rodarão mais rapidamente e serão mais escalonáveis se acessarem o banco de dados Caché.

SQL TRADICIONAL	OBJECT EXTENDED SQL
SELECT SC.FullName, SM.Descr, MS.Value, SI.InvDate, SI.InvNumber FROM MainSales MS, SalesItem SI, SalesProduct SP, SalesCustomer SC, SalesMarket SM WHERE SI.SalesItemID *= MS.SalesItem AND SP.SalesProductID *= MS.Product AND SC.SalesCustomerID *= MS.Customer AND SM.SalesMarketID *= SC.SalesMarket AND SP.Descr = 'Hammer'	SELECT Customer->FullName, Customer->SalesMarket->Descr, Value, SalesItem->InvDate, SalesItem->InvNumber FROM MainSales WHERE Product->Descr = 'Hammer'

Quadro A13 – SQL tradicional e sua extensão no Caché.

OBJETOS DO CACHÉ

O estado dos objetos Caché é permanentemente mantido no Servidor de Aplicações do Caché. Quando um programa Java, C++, C#, Visual Basic, ou qualquer outro externo ao servidor de aplicações acessa um objeto Caché, ele chama um gabarito da classe na linguagem nativa. Aquele gabarito da classe (que é gerado automaticamente pelo Caché) se comunica com o servidor de aplicações para invocar métodos no servidor do Caché e acessar ou modificar propriedades. Para apressar a execução e reduzir o messaging, o Caché faz uma cópia em cache dos dados do objeto no cliente e faz as atualizações quando possível, voltando com outras mensagens. Para o programa do usuário, parece que o objeto é local; o Caché controla transparentemente todas as comunicações com o servidor. O gabarito Java e a biblioteca de suporte são completamente baseados em Java, assim eles podem ser usados na Web ou em dispositivos especializados Java.

O Caché Objects

O Caché Objects é o componente da estrutura do banco de dados Caché responsável por disponibilizar, simultaneamente, o desempenho e o poder de modelagem da estrutura multidimensional de dados associada às características da tecnologia de OO.

As operações fundamentais do Caché Objects estão baseadas na definição de classes de objetos e subseqüente criação, armazenamento, recuperação e manipulação de instâncias específicas dessas classes.

Principais componentes do Caché Objects:

- Class dictionary - um repositório de definições de classes (metadados), que descrevem classes específicas. Este repositório é armazenado com o banco de dados Caché. Este componente também é usado pelo Caché SQL Engine sendo responsável pela sincronização dos objetos e acesso relacional.
- Class Compiler - conjunto de programas que convertem a definição da classe em código executável.
- Object Runtime System - conjunto de características construídas na máquina virtual Caché que suporta operações de objetos em uma aplicação.
- Caché Class Library - conjunto das classes predefinidas que vêm com o Caché. Isto inclui as classes que são usadas para fornecer comportamentos para classes definidas pelo usuário (tais como tipos persistentes ou de dados) bem como classes que são pretendidas para o uso direto em aplicações.
- Language Bindings - uma combinação dos geradores de código e dos componentes runtime que fornecem o acesso externo ao Caché Objects. Estes bindings incluem o Caché Java Binding, Caché ActiveX binding e o Caché C++ Binding.
- Gateways - os componentes server-side que dão ao Caché objects acesso aos sistemas externos. Estes gateways incluem Caché SQL Gateway e o Caché Activate ActiveX Gateway.

Classes no Caché Objects

No Caché Objects as classes estão divididas em vários tipos. A divisão básica é a de Classes de Objetos (Object Classes) e Classes de Tipos de Dados (Data Type Classes). As Classes de Objetos representam entidades específicas e modelam como estas interagem com o meio externo. As Classes de Tipos de Dados representam valores literais, como inteiros, cadeias e datas para atributos de classes de objetos.

As classes do Caché Objects são formadas por um conjunto de parâmetros, métodos e propriedades. Os parâmetros das classes do Caché definem toda a parte comportamental da mesma quando da sua compilação. As propriedades modelam o conteúdo de uma classe. O valor das propriedades de uma classe define seu estado interno. As propriedades de uma classe podem ser tipos de dados simples (como inteiros e strings), referências a outros objetos (relacionamentos) ou objetos embutidos. Além disso, as propriedades podem tomar a forma de um valor atômico, um array ou uma lista. Os métodos definem como a classe se comunica com o meio externo. Eles podem ser de instância, de classe ou de consulta.

Estrutura de Objetos e Classes no Caché

A estrutura de Objetos e Classes do Caché consiste basicamente em visualizar o modelo de dados multidimensional na forma de objetos através de um dicionário de classes. No dicionário de Classes estão definidas as propriedades e os métodos das classes. Estas classes são compiladas dando origem a programas em COS que são responsáveis pela execução dos métodos e manipulação das propriedades da classe.

Herança

Três pontos devem ser levados em consideração quando se fala em herança no Caché. O primeiro diz respeito a herança de propriedades. O valor das propriedades herdadas é duplicado na estrutura multidimensional que armazenará o conteúdo da nova classe. No entanto, no dicionário de classes, as definições das propriedades não são duplicadas. O segundo ponto diz respeito a herança de métodos, que também são duplicados nos programas COS que manipularão a nova classe, mas mantém-se únicos na definição do dicionário de classes. O terceiro ponto que merece destaque diz respeito ao suporte a herança múltipla. Neste mecanismo, para métodos de mesmo nome, o Caché mantém o método pertencente a última classe da lista de classes pais.

Geradores de Métodos

O Caché inclui várias tecnologias de objeto exclusivas e avançadas – uma dos quais é o gerador de métodos. Um gerador de métodos é um método que se executa durante o tempo de compilação gerando código que pode rodar quando o programa é executado. Um gerador de métodos tem acesso a definições de classes, incluindo propriedades e definições de métodos e parâmetros, para permitir que ele gere um método que é customizado para a classe. Os geradores de métodos são particularmente poderosos em combinação com heranças múltiplas – podem definir funcionalidades dentro de uma classe multiplamente herdada que se customiza nas subclasses.

INDEXAÇÃO TRANSACIONAL POR BIT-MAP

O Caché fornece Indexação Transacional por Bit-Map que aumenta o desempenho de consultas complexas dando desempenho de fast data warehouse.

O desempenho dos bancos de dados é extremamente dependente da disponibilidade de índices das propriedades que são freqüentemente usadas nas buscas efetuadas no banco de dados. A maioria dos bancos de dados usa índices que, para cada valor possível da coluna ou propriedade, mantém uma lista dos IDs das linhas (ou IDs dos objetos) para as linhas/objetos que têm aquele valor.

Um índice por mapa de bits contém um Bit-Map separado para cada valor possível de uma coluna/propriedade, com um bit para cada linha/objeto armazenado. O bit 1 significa que a linha/objeto tem aquele valor para a coluna/propriedade.

A vantagem dos índices Bit-Map é que consultas complexas podem ser processadas executando operações Booleanas (E, OU) nos índices – o que permite determinar,

exatamente, quais instâncias (linhas) se ajustam às condições da consulta, sem procurar pelo banco de dados inteiro.
Freqüentemente, índices Bit-Map podem apresentar tempos de resposta para consultas que procuram grandes volumes de dados superiores aos demais por um fator de 100 vezes ou mais.
Os mapas de bits tradicionalmente sofrem de dois problemas: podem ser lentos para serem atualizados em bancos de dados relacionais e podem consumir muito armazenamento. Assim, nos bancos de dados relacionais, eles são raramente usados em aplicações que processam transações.
O Caché introduziu uma nova tecnologia – índices transacionais Bit-Map – que alavanca estrutura de dados multidimensionais para eliminar estes dois problemas. A atualização destes Bit-Maps é freqüentemente mais rápida que a dos índices tradicionais e eles utilizam sofisticadas técnicas de compressão para reduzirem o armazenamento. O resultado apresenta Bit-Maps que freqüentemente podem ser usados para pesquisar milhões de registros em uma fração de segundo em um banco de dados processando transações on-line. O Caché oferece tanto índices Bit-Map tradicionais como transacionais. O Caché também suporta índices multicoluna.

ENTERPRISE CACHÉ PROTOCOL PARA SISTEMAS DISTRIBUÍDOS
O protocolo ECP (Enterprise Caché Protocol) oferece um desempenho elevado e tecnologia escalonável que permitem aos computadores de um sistema distribuído usarem os bancos de dados uns dos outros. O uso do ECP não requer nenhuma mudança nas aplicações – as aplicações tratam o banco de dados simplesmente como se ele fosse local.
Cada Servidor de Aplicações inclui seu próprio Servidor de Dados que pode operar sobre dados que residem em seu próprio sistema de disco ou sobre blocos que lhe foram transferidos de outro Servidor de Dados pelo ECP. Quando um cliente fizer um pedido de informação, o Servidor de Aplicações do Caché tentará satisfazer o pedido a partir de seu cachê local. Se não puder fazê-lo, ele pedirá os dados necessários ao Servidor de Dados Caché remoto. A resposta inclui os blocos do banco de dados onde aqueles dados foram armazenados. Estes blocos ficam em cache no Servidor de Aplicações, disponíveis para todas as aplicações que rodam naquele servidor. O ECP cuida automaticamente de administrar a consistência do cache ao longo da rede e propagar as mudanças de volta aos servidores de dados. O caching reduz o tráfego de rede entre o banco de dados e os servidores de aplicação, de forma que uma dada rede pode suportar mais servidores e clientes.
O uso do ECP é transparente para as aplicações. As aplicações escritas para rodar em um único servidor rodam em um ambiente de múltiplos servidores sem modificações. Para usar o ECP, o gerente do sistema simplesmente identifica um ou mais servidores de dados, então usa o Namespace Mapping para indicar que as referências a alguma ou a todas as estruturas globais (ou partes das estruturas globais) se referem a algum servidor de dados remoto.
Cada sistema Caché tanto pode funcionar como um servidor de aplicações quanto como um servidor de dados para outros sistemas. O ECP suporta qualquer combinação de servidores de aplicações e servidores de dados e qualquer topologia ponto a ponto de até 255 sistemas.
Tolerância a falhas em sistemas distribuídos
Para sistemas distribuídos usando o Enterprise Caché Protocol, na eventualidade de uma falha temporária na rede ou colapso e reinicialização do Servidor de Dados, os servidores tentam se reconectar. Se uma reconexão tiver sucesso dentro de um período de tempo especificado, os Servidores de Aplicações reenviam qualquer solicitação não completada, e as operações continuam sem efeito observável para os usuários dos Servidores de Aplicações remotos além de uma breve pausa. Caso contrário, as transações retroagem, e é emitido um erro aos usuários dos processos.
Em algumas configurações, um servidor de dados pode ser configurado para falhar sobre um servidor de reserva (Shadow Server) ou membro do cluster, aumentando ainda mais a

confiabilidade. O novo servidor retoma o processamento do servidor de dados que falhou, permitindo que a operação seja ininterrupta.

Clusters

Os clusters de bancos de dados têm características de suportarem falhas automaticamente. Em um cluster de banco de dados, múltiplos computadores acessam as mesmas unidades de disco e usam sua capacidade de agrupamento para coordenarem acesso compartilhado ou exclusivo a blocos do disco. Se um computador falhar, seus processos estão perdidos, mas os outros computadores continuam funcionando. Os usuários do computador que falhou têm suas transações retroagidas e estes usuários podem então entrar em outro computador.

Freqüentemente, é usado balanceamento da carga para atribuir dinamicamente os usuários aos computadores agrupados.

Embora clusters de bancos de dados dêem flexibilidade operacional e aumentem a confiabilidade de um sistema, eles normalmente requerem mais administração de sistema do que outros sistemas, e requerem hardware especial e suporte ao sistema operacional.

Escalabilidade e disponibilidade

O Caché escala de forma quase linear tanto vertical como horizontalmente, isto é:

- Vertical: adicionando-se mais memória ou processadores. O Caché alocará os processos para os processadores disponíveis sem necessidade de configurações. Adicionando-se mais memória, o Caché poderá utilizá-la para trabalhar com um maior número de processos ou como cache para dados e rotinas.
- Horizontal: adicionando-se mais máquinas e utilizando-se o ECP (Enterprise Caché Protocol) é simples distribuir a carga através de várias máquinas e construir esquemas de cluster de load-balancing e failover sem deixar máquinas ociosas.

O Caché trabalha com qualquer storage disponível no mercado (NetApp, SUN, etc.) e pode ser utilizado nas mais diversas configurações de cluster em ambientes Linux, Windows, HP, e etc.

O Caché permite uma economia maior de hardware não só pelo fato de exigir menos como também pelo fato de poder utilizar as máquinas de stand by de clusters de failover como máquinas de processamento enquanto não ocorrerem falhas. Ou seja, nenhuma máquina fica ociosa.

Travamento Lógico

Em sistemas com milhares de usuários, a redução de conflitos entre processos concorrentes é crítica para conseguir alto fluxo de trabalho. Um dos maiores conflitos ocorre entre transações que desejam acessar os mesmos dados.

Os processos do Caché não travam páginas de dados inteiras enquanto fazem atualizações. Ao invés disto, como as transações requerem acessos freqüentes ou mudanças de pequenas quantidades de dados, o travamento do banco de dados no Caché é feito ao nível lógico. Conflitos no banco de dados são ainda mais reduzidos usando operações de adição e de subtração atômicas que não requerem travamento. Estas operações são particularmente úteis no incremento de contadores usados para alocar números de ID ou modificar contadores de estatísticas.

O modelo de dados multidimensional transacional permite escalar aplicações baseadas no Caché até milhares de clientes sem sacrificar seu alto desempenho. Isso pode ser feito porque o acesso aos dados em um modelo multidimensional não é significativamente afetado pelo tamanho ou complexidade do banco de dados, diferentemente do modelo relacional. As transações podem acessar os dados de que necessitam sem executar complicadas junções ou saltando de tabela em tabela. O fato de o Caché usar travamento lógico para atualizações em vez de fechar páginas físicas é outra contribuição importante para a concorrência, assim como seu sofisticado cache de dados dentro da rede.

GERENCIAMENTO DO SISTEMA / OPERAÇÃO DA BASE DE DADOS

O Caché foi projetado para sistemas que vão de 4 usuários a sistemas com dezenas de milhares de usuários. Não há no Caché nenhuma necessidade de executar operações como reconstruir periodicamente os índices para melhorar o desempenho ou recarregar um banco de dados quando uma nova versão é liberada, como é comum com alguns sistemas relacionais. O Caché inclui um complemento inteiro de utilidades de gerenciamento do sistema que podem ser inteiramente acessadas de forma remota. As principais funções de gerenciamento do sistema podem ser programadas para operação independente.

Backup

O Caché suporta backup completo, incremental, e incremental cumulativo. Estas cópias de reserva podem ser executadas enquanto o banco de dados está ligado e rodando, inclusive enquanto estão acontecendo atualizações do banco de dados. Os backups podem ser programados de forma a serem executados automaticamente sem operador.

Reconfigurações do hardware e do banco de dados

Em cada sistema, o Caché mantém mapas de namespace que especificam onde os dados e o código estão armazenados. Mudanças no sistema, até mesmo grandes mudanças como adicionar um servidor de banco de dados ou comutar para um servidor de reserva, podem ser realizadas revisando o mapa de namespace. Reconfigurações programadas podem ser feitas dinamicamente, de forma transparente para as aplicações. Podem ser configurados mapas de namespace de contingências para compensar uma variedade de cenários de falhas.

Mudanças no fonte de um sistema em funcionamento

Para introduzir uma mudança no CachéObjectScript ou Basic de um sistema em funcionamento, pode-se carregar a rotina modificada do fonte no único servidor de dados em que reside o código. O código será compilado automaticamente – não será requerido nenhum *link* – e os servidores de aplicações serão notificados de que precisam recarregar a rotina revisada.

O SERVIDOR DE APLICAÇÕES DO CACHÉ

O servidor de aplicações Caché oferece capacidade de programação por objetos, fornece caching de dados e integra fácil acesso a uma variedade de tecnologias. O servidor de aplicações Caché fornece:

- A Máquina Virtual Caché que roda duas linguagens de *scripting* embutidas – Caché ObjectScript e Basic.
- Acesso aos Servidores de Dados Multidimensionais Caché no mesmo e em outros computadores com roteamento transparente.
- Software de conectividade com caching no lado do cliente, para permitir rápido acesso a Objetos Caché de todas as tecnologias comumente usadas, incluindo Java, C++, C#, COM, .NET, Visual Basic e Delphi. O Caché executa automaticamente a gestão de redes entre o cliente e o servidor de aplicação.
- Compatibilidade com SOAP e XML.
- Acesso a SQL usando ODBC e JDBC, incluindo caching no cliente e no servidor de aplicações para obter alto desempenho.
- Acesso a bancos de dados relacionais.
- Caché Server Pages para aplicações Web de alto desempenho, fáceis de programar.
- Caché Studio – um IDE para desenvolver rapidamente e depurar aplicações com Caché.
- O código para as linguagens de *scripting* fica armazenado no banco de dados e pode ser mudado on-line, com mudanças que se propagam automaticamente a todos os servidores de aplicações.

A MÁQUINA VIRTUAL CACHÉ E AS LINGUAGENS DE PROGRAMAÇÃO

O núcleo do servidor de aplicações do Caché é a máquina virtual Caché, extremamente rápida, e capaz de suportar as duas linguagens de *scripting* do Caché – o CachéObjectScript e o Basic.

O CachéObjectScript é uma linguagem orientada a objetos, com estruturas de dados extremamente flexíveis. O Basic fornece um modo fácil para os programadores de Visual Basic começarem a usar o Caché. Semelhante ao VBScript, o Basic é estendido para ter acesso direto ao Caché Multidimensional Arrays.
Cada processo de usuário na máquina virtual Caché tem acesso direto às estruturas de dados multidimensionais fazendo ligações à memória compartilhada que acessa o cache do banco de dados compartilhado. Todas as outras tecnologias (Java, C++, ODBC, JDBC, etc.) se conectam pela máquina virtual Caché para acessar o banco de dados.

Interoperabilidade

Desde que o CachéObjectScript e o Basic estejam implementados na mesma máquina virtual Caché, eles são completamente interoperáveis:

- qualquer método de objeto pode ser escrito em qualquer linguagem – a mesma classe pode usar ambas as linguagens;
- cada linguagem pode fazer ligações para código escrito na outra linguagem;
- elas compartilham variáveis, arrays e objetos.

CACHÉOBJECTSCRIPT

O CachéObjectScript é uma poderosa linguagem de programação orientada a objetos projetada para rápido desenvolvimento de aplicações de bancos de dados. Algumas das características fundamentais da linguagem:

Estrutura Global

O CachéObjectScript é orientado por comandos; conseqüentemente tem sintaxe como:

```
set x=a+b
do rotate(a,3)
if (x>3)
```

Há um conjunto de funções de sistema incorporadas que são particularmente poderosas para manipulação de texto. Todos os seus nomes começam com um único caractere '$' para distingui-las de variáveis e nomes de arrays. Por exemplo:

```
$length(string)                 // determina o tamanho da string
```

Armazenamento Flexível de Dados

Uma das características mais exclusivas do CachéObjectScript é seu armazenamento de dados flexível e dinâmico. Com raras exceções, em qualquer lugar na linguagem onde pode ser usada uma variável, também poderia ser usado um array, uma propriedade de objeto, ou uma referência global.
Na maioria das linguagens de computador, os tipos de dados são uma extensão de conceitos de armazenamento de hardware (integer, float, char, etc.). Porém, o CachéObjectScript tem por filosofia que não se pensa usando tais tipos de armazenamento, e que estes tipos de dados centrados em computador simplesmente impedem o rápido desenvolvimento de aplicações. Requerendo declarações e informações de dimensões, elas introduzem muito mais erros do que ajudam a preveni-los (erros como um overflow de 2-bytes de um inteiro, ou quando um string excede sua alocação de memória e corrompe outro armazenamento). Porém, a tipificação de objetos, como Pessoa, Fatura ou Animal, é vista como altamente valiosa e consistente com o modo de pensar dos humanos.
Assim, no CachéObjectScript as propriedades dos objetos são fortemente tipificadas, mas os outros três tipos de armazenamento (variáveis, arrays e nodos globais) são completamente polimórficos, entidades sem tipo que não precisam ser declaradas ou definidas. Eles simplesmente passam a existir quando são usados e se moldam às necessidades dos dados que estão armazenando e à forma como estão sendo usados em uma expressão. Nem sequer os arrays precisam de qualquer especificação de tamanho, dimensão, tipo de subscritos ou dados.

Acesso Direto ao Banco de Dados

Uma referência direta ao banco de dados (uma referência global) é essencialmente uma referência a um array multidimensional precedida pelo caractere circunflexo '^'. Este caractere indica que esta é uma referência a dados armazenados no banco de dados em lugar de dados privados de processo temporário.

Da mesma forma que com os arrays multidimensionais e variáveis, não são requeridas declarações, definições, ou reservas de armazenamento para acessar ou armazenar dados no banco de dados; os dados globais simplesmente passam a existir quando os dados são armazenados.

Referências a Objetos

No CachéObjectScript um oref é usado para acessar um objeto (um oref é tipicamente uma variável cujo valor especifica qual objeto na memória está sendo referido). O oref é seguido de um ponto e então pelo nome de uma propriedade ou método. Podem ser usadas referências a objetos onde quer que uma expressão possa ser usada. Por exemplo:

```
set name=person.Name        // 'person' is a variable whose value is an oref
                            // the person's name is put into the variable 'name'
if (person.Age>x)           // see if the person's age is greater than 'x'
set money=invoice.Total()   // 'Total()' is a method that calculates the sum of
                            // all of the invoice's line items
```

Métodos podem também ser executados com um comando DO quando nenhum retorno de valor é necessário. Por exemplo:

```
do part.Increment()         //'Increment()' é um método cujo valor retornado,
                            // caso exista, não tem interesse
```

O oref não é o mesmo que um objeto de base de dados ID. O objeto ID é um valor que está permanentemente associado a um objeto do banco de dados; é usado para retroceder e armazenar um objeto do banco de dados. Uma vez que um objeto está na memória, ele é definido como um valor oref reusável que então é usado para acessar dados do objeto. Na próxima vez que que o mesmo objeto de banco de dados for trazido para a memória, ele provavelmente será definido com um valor oref diferente.

Código de Chamada

Em algumas linguagens de objetos, todo o código tem que fazer parte de algum método. O CachéObjectScript não tem esta restrição – o código pode ser chamado diretamente ou pode ser chamado pela sintaxe do objeto.

O código é freqüentemente chamado usando o comando DO.

```
do rotate(a,3)
```

O código que retorna um valor também pode ser chamado como uma função. Por exemplo, set x=a+$$insert(3,y) chama o procedimento escrito pelo programador ou a sub-rotina insert.

O código também pode ser invocado como um método do objeto.

```
set money=invoice.Total()   //Total() returns the invoice total amount do
part.Increment()            //'Íncrement()' is a method whose return value,
                            // if any, is not of interest
```

Tanto a chamada por valor (call by value) como a chamada por referência (call by reference) são suportadas por parâmetros.

Rotinas

O código do CachéObjectScript está fundamentalmente organizado em um conjunto de rotinas. Cada rotina (tipicamente de até 32kb em tamanho) é atômica no sentido de que pode ser editada, armazenada e compilada independentemente. As rotinas são ligadas dinamicamente durante a execução; não há nenhum passo separado de "linkagem" do programador. O código da rotina é armazenado no banco de dados; assim, as rotinas podem ser chamadas dinamicamente pela rede em vez de precisarem ser instaladas em cada computador.

Dentro de uma rotina, o código é organizado como um jogo de procedimentos e/ou sub-rotinas. (Um método de objeto é um procedimento, mas ele é acessado por uma sintaxe diferente). As rotinas podem ser editadas e compiladas pelo Caché Studio.

Métodos de Objetos

As definições de classe e o código de seu método são armazenados em arquivos de dados globais e o compilador de classe compila cada classe em uma ou mais rotinas. Cada método é simplesmente um procedimento em uma rotina, embora só possa ser invocado através da sintaxe de objeto. Por exemplo, se a classe Paciente definir um método Admit e a

variável Pat identifica um objeto Paciente específico, então chamamos o método Admit para aquele objeto com a seguinte sintaxe:

```
do Pat.Admit()          // Calls the admit method for Patient
set x = Pat.Admit()     // Calls the same method but uses the return value
```

Procedimentos e Variáveis Públicas/Privadas

Um procedimento é um bloco de código dentro de uma rotina que é semelhante a uma função em outras linguagens. Um procedimento consiste de um nome, uma lista de parâmetros formais, uma lista de variáveis públicas e um bloco de código delimitado por '{}'. Por exemplo: Admit(x,y)[name,recnum] {...código}.

No CachéObjectScript, algumas variáveis são comuns e outras são privadas de um determinado procedimento. Cada variável que é usada dentro de um procedimento é considerada privada daquele procedimento a menos que seja listada na lista pública. No exemplo anterior, name e recnum acessam as variáveis públicas por estes nomes, enquanto que todas as outras variáveis só existem para esta invocação deste procedimento. As variáveis cujos nomes começam por um caractere '%', implicitamente, são sempre públicas. Os procedimentos não podem ser aninhados, embora um procedimento possa conter sub-rotinas.

Sub-rotinas

As rotinas também podem conter sub-rotinas que são mais leves que os procedimentos. Uma sub-rotina pode conter uma lista de parâmetros e retornar um valor, mas ela não deve ter uma lista pública ou estrutura formal de bloco. As sub-rotinas podem ser embutidas dentro de procedimentos ou podem estar ao mesmo nível de um procedimento em uma rotina.

As sub-rotinas permitem a chamada de código usando o mesmo conjunto de variáveis públicas/privadas de quem chama e podem ser chamadas mais rapidamente.

Uma sub-rotina embutida dentro de um procedimento usa o mesmo escopo variável do procedimento e só pode ser chamada de dentro daquele procedimento. As referências feitas a variáveis para uma sub-rotina que não fazem parte de um procedimento serão todas para variáveis públicas.

INTEGRANDO O CACHÉ COM OUTRAS TECNOLOGIAS E FERRAMENTAS

Basic

No Caché, o Basic foi estendido para suportar acesso direto às estruturas de dados do core do servidor de dados – o array multidimensional –, bem como a outras características do servidor de aplicações do Caché. Ele suporta diretamente o Caché Object Model que usa a sintaxe do Visual Basic e roda na máquina virtual Caché.

O Basic pode ser usado como método de classes ou como rotinas de Caché. O Basic pode chamar o CachéObjectScript, e vice-versa, sendo que ambas as linguagens acessam as mesmas variáveis, arrays e objetos em processo na memória.

C++

Cada classe Caché pode ser projetada como uma classe C++, com métodos correspondentes para cada propriedade e método da classe Caché. Para os programas C++, estas classes se parecem exatamente como qualquer outra classe C++ local, e o Caché trata automaticamente todas as comunicações entre o cliente e servidor. As propriedades da classe são cacheadas no cliente, e as chamadas do método C++ invocam métodos correspondentes ao lado do servidor – incluindo métodos para armazenar um objeto no banco de dados e depois recuperá-lo.

Java

Java é uma tecnologia de programação cada vez mais popular, mas conectar aplicações Java a um banco de dados pode ser desafiante. Sua conexão a um banco de dados relacional requer extensa codificação SQL – que consome muito tempo e enfraquece muitas das vantagens da tecnologia de objetos Java.

Normalmente, é preferida a sintaxe de objetos para acessar bancos de dados de alguns usuários que gostam da funcionalidade do Enterprise Java Beans. Cada classe Caché pode ser projetada como uma classe Java (ou EJB), com métodos correspondentes a cada propriedade e método da classe Caché. Para os programas Java estas classes se parecem exatamente como qualquer outra classe Java local. Para fazer uso de classes Java a InterSystems disponibiliza uma biblioteca Java que trata de todas as comunicações entre cliente e servidor.
O estado de cada objeto Caché é mantido no servidor de aplicações Caché, embora as propriedades da classe também sejam cacheadas no cliente para melhorar o desempenho. As chamadas do método Java invocam chamadas de métodos correspondentes no servidor de aplicações Caché – incluindo métodos para armazenar um objeto no banco de dados e depois recuperá-lo. Para o cliente é transparente qual servidor de dados Caché contém os dados, até mesmo se os dados do objeto forem armazenados em um banco de dados relacional acessado pelo servidor de aplicações Caché.

Métodos Caché Escritos em Java

Os métodos de classes Caché podem ser escritos em Java usando o Caché Studio. Porém, diferentemente do CachéObjectScript e do Basic, os métodos Java não são executados pela máquina virtual Caché. Ao invés disto, eles são incluídos na classe Java gerada e executados em qualquer máquina virtual Java. Tal código não é acessível por métodos que não sejam Java.

Dando Persistência às Aplicações J2EE

Os desenvolvedores de aplicações J2EE que usam o Enterprise Java Beans (EJB), trabalham principalmente com objetos até o momento em que eles precisam acessar o banco de dados. Então, normalmente, eles são forçados a voltar a usar SQL. Através de sua *interface* JDBC, o Caché pode fornecer respostas SQL rápidas para tais aplicações. Porém, o acesso SQL não constitui, geralmente, a abordagem preferida.
Os bancos de dados orientados a objetos representam uma técnica de acesso mais natural para os programadores EJB. O Caché projeta classes Caché como EJBs, gerando automaticamente métodos de persistência de alto desempenho para Bean Managed Persistence (BMP). Isto evita a sobrecarga de SQL e mapeamento objeto/relacional – o resultado é maior escalabilidade para as aplicações J2EE.

Estratégias de Alto Desempenho com Java

Normalmente, o código de uma máquina virtual Caché executa mais rapidamente que o código de uma máquina virtual Java e, normalmente, é também muito mais rápido de se desenvolver. Embora seja possível construir aplicações completas em Java, a maioria dos clientes obtém os melhores resultados maximizando a quantidade de código que colocam nas linguagens de programação do servidor de aplicações Caché (Basic e CachéObjectScript), limitando a porção Java à *interface* da aplicação com o usuário.

.NET

Devido a seu acesso de dados aberto e flexível, o Caché trabalha em interligação com o .NET. Há muitos modos de conectá-los, incluindo objetos, SQL, XML, e SOAP.

COM & ADO

COM e ADO são ambas consideradas as tecnologias mais "antigas" da Microsoft, incluídas no .NET. O Caché interage com o COM – através de seu ObjectServer para ActiveX – expondo as classes Caché como classes compatíveis com COM. Em contraste, o ADO fornece wrappers para os objetos de dados relacionais e interage com o Caché pelo acesso de dados relacionais do Caché.

ADO.NET

O ADO.NET é uma nova geração do ADO, otimizada para uso na estrutura .NET. Pretende-se tornar as aplicações .NET independentes do banco de dados. Da mesma forma que o ADO, ele usa o acesso de dados relacionais do Caché para interagir com o Caché.

Web Services

Há dois modos de usar Web Services no .NET. Um é enviar documentos XML sobre HTTP. O outro é usar o protocolo SOAP para simplificar a troca de documentos XML. Em ambos os

casos, como o Caché pode expor dados como documentos XML ou como documentos SOAP, ele trabalha em perfeita interligação com o .NET Web Services.

XML

Da mesma maneira que o HTML é uma linguagem de mark-up compatível com a Internet para exibir dados, o XML é uma linguagem de mark-up para trocar dados entre aplicações.
Usando o XML, aplicações distintas (dentro de uma companhia, ou de diferentes empreendimentos) podem compartilhar dados sobre uma rede. A estrutura de dados do XML é hierárquica e multidimensional, tornando natural o ajuste para a máquina de dados multidimensionais do Caché.
O Caché fornece uma *interface* bidirecional, para XML que elimina a necessidade de os desenvolvedores criarem manualmente uma camada de mapeamento para o processamento de dados XML e o banco de dados.
Tudo o que é preciso para tornar uma classe Caché compatível com XML é tê-la herdado de uma classe do %XMLAdaptor que é incluída no Caché. Ela fornece todos os métodos necessários para:

- Criar um DTD (Document Type Definition) ou um Schema XML para a classe. DTDs e Schemas são documentos que definem a estrutura dos dados XML. O Caché gera automaticamente DTDs e Schemas XML, mas os desenvolvedores que desejam customizar a formatação XML de uma classe podem fazê-lo.
- Formatar dados automaticamente (instâncias de classes) como XML, conforme o DTD ou Schema definido.

O Caché vem com outras classes que fornecem métodos que permitem aos desenvolvedores:

- Importar Schemas XML e criar automaticamente as classes Caché correspondentes.
- Importar dados em documentos XML como instâncias de classes Caché, por meio de uma simples API.
- Dividir e validar documentos XML por meio do parser XML (SAX) incorporado.

Web Services

Os Web Services permitem o compartilhamento de funcionalidades de aplicações sobre a Internet como também dentro de uma organização ou sistema. Os Web Services têm uma *interface* descrita em WSDL (Web Service Definition Language) que retorna um documento XML formatado de acordo com o protocolo SOAP.
O Caché habilita qualquer método de classe, qualquer procedimento SQL armazenado e qualquer consulta a ser automaticamente exposta como um Web Service. O Caché gera o descritor WSDL para o serviço e, quando o serviço é invocado, envia a resposta, adequadamente formatada como SOAP/XML. O Caché também facilita o rápido desenvolvimento gerando automaticamente uma página Web para testar o serviço, sem a necessidade de construir uma aplicação cliente.

Rational Rose

O Caché RoseLink é um add-in fornecido pela InterSystems à popular ferramenta de modelagem de objetos Rational Rose. Ele fornece toda a *interface* de engenharia para ida e volta entre o Caché e o Rational Rose, permite que sejam projetados schemas em Rational Rose e exportados para implementação no Caché. De forma semelhante, os desenvolvedores podem importar schemas do Caché para o Rational Rose para completar o projeto ou a documentação.

Dreamweaver

O Caché vem com um add-in para o ambiente de projeto de páginas Web do Dreamweaver que dá aos usuários do Dreamweaver uma maneira fácil de criarem Caché Server Pages, e usarem o Caché Application Tags.

CACHÉ STUDIO

O Caché Studio é um poderoso ambiente de desenvolvimento integrado, que permite a criação, a depuração, e o teste de aplicações Caché. Suas características incluem:

Editores

Os programadores podem escrever código CachéObjectScript, Basic e Java bem como arquivos de definição de classes e páginas Web. Clicando-se sobre uma propriedade, método, etc. no editor exibem-se as características daquela propriedade, etc., e se permite que elas sejam editadas na janela de exibição.

Verificação Automática da Sintaxe

Sintaxe imprópria é automaticamente sinalizada no CachéObjectScript, Basic, HTML e Java.

Comando View Web Page

Os desenvolvedores podem testar páginas Web diretamente de dentro do ambiente de desenvolvimento.

Debugger Visual

Para compilar e depurar o código com um depurador visual que também pode anexar e depurar um processo Caché existente.

Import e Export

Importa automaticamente definições de classes Caché de definições de schema DDL relacionais, ou ferramentas de modelagem de objetos. A ferramenta Export permite criar arquivos de definições para o envio a outros computadores.

TIPOS DE DADOS

Caché suporta vários tipos de dados para variáveis em que cada tipo de dados é uma classe. Cada tipo de dados classe representa um tipo literal específico, como uma string ou integer. Estes tipos de dados classe determinam o comportamento das propriedades de objeto e campos em tabelas relacionais. Caché também permite criar tipos próprios de dados. O quadro A14 mostra os tipos de dados classe.

Nome da Classe	Tipo	tipo SQL
%Binary	binary data	BINARY, BINARY VARYING, RAW, VBINARY
%Boolean	a boolean value	N/A
%Currency	a currency value	MONEY, SMALLMONEY
%Date	a date	DATE
%Float	a floating point value	DOUBLE, DOUBLE PRECISION, FLOAT, REAL
%Integer	an integer	BIT, INT, INTEGER, SMALLINT, TINYINT
%List	data in $List format	N/A
%Name	a name in the form “Lastname,Firstname”	N/A
%Numeric	a numeric values of varying precision	DEC, DECIMAL, NUMBER, NUMERIC
%Status	an error status code	N/A
%String	a string	CHAR, CHAR VARYING, CHARACTER, CHARACTER VARYING, NATIONAL CHAR, NATIONAL CHAR VARYING, NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, NATIONAL VARCHAR, NCHAR, NVARCHAR, VARCHAR, VARCHAR2
%Time	a time value	TIME
%TimeStamp	a value for a time and date	TIMESTAMP

Quadro A14 - tipos de dados classe.

Tipos de dados classe possuem as seguintes características:

- fornecem interoperabilidade para SQL, ODBC, ActiveX e Java provendo operação lógica SQL, tipos de dados do cliente e informação de conversão;
- fornecem validação para valores de dados literais que podem ser estendidos ou cutomizados usando parâmetros de classes de tipos de dados;

- gerenciam a conversão de dados literais para seu armazenamento (em disco), lógico (em memória) e formatos de exibição;

Tipos de dados classe diferem de outras classes de vários modos:

- não podem ser instanciados ou armazenados independentemente;
- não podem conter propriedades;
- suportam um conjunto específico de métodos (chamados de tipos de dados *interface*).

Operações

A principal função de um tipo de dados classe é para especificar os tipos de propriedades em uma classe. Seu formato básico é:
Property City As %String;

Validação de funcionalidade

Qualquer propriedade que usa um tipo de dados de sistema (ou um tipo de dados derivado de um tipo de dados de sistema) suporta validação de dados através de um método gerador associado a propriedade.
O método de validação tem um nome no formato PropertyNameIsValidDT; por exemplo, uma propriedade Age de tipo %Integer tem um método de AgeIsValidDT associado. O método confere a validade de um valor especificado para a propriedade.

Parâmetros

Tipos de dados classe suportam parâmetros, que executam várias ações e variam de acordo com o tipo de dados. Estes parâmetros são apresentados no quadro A15.

Parâmetro	Descrição
COLLATION	Especifica a maneira pela qual os valores das propriedades são transformados para indexação
DISPLAYLIST	Usado em conjunto com o parâmetro VALUELIST para propriedades de enumeração (múltipla escolha). DISPLAYLIST, se não nulo, representa os valores de exibição para a propriedade que corresponde aos valores lógicos listados em VALUELIST. Os valores de exibição retornam pelo método LogicalToDisplay.
VALUELIST	Usado para propriedades de enumeração. VALUELIST pode ser uma string nulo ("") ou uma lista delimitada de valores lógicos (onde o delimitador é o primeiro caracter).
FORMATO	Especifica o formato para o valor de exibição do tipo de dados. O valor FORMAT corresponde à opção de formatação da função $FNUMBER.
INDEXSUBSCRIPTS	Se presente, especifica o número de subscritos usados pela propriedade nos índices, enquanto usando uma vírgula como um delimitador no valor da propriedade; a classe %CacheStorage usa este número.
MAXLEN	Especifica o número máximo de caracteres que a string pode conter.
MAXVAL	Especifica o valor lógico máximo permitido pelo tipo de dados.
MINLEN	Especifica o número mínimo de caracteres que a string pode conter.
MINVAL	Especifica o valor lógico máximo permitido pelo tipo de dados.
ODBCDELIMITER	Especifica o caracter delimitador usado para construir um valor %List quando for projetado via ODBC.
PATTERN	Especifica um padrão para iniciar a string.
SCALE	Especifica o número de dígitos após o ponto decimal.
TRUNCATE	Especifica se a string é truncado para MAXLEN, onde 1 é TRUE e 0 é FALSE.
XSDTYPE	Declara o tipo XSD usado ao projetar esquemas XML.

Quadro A15 – parâmetros para tipos de dados classe.

Os parâmetros suportados para cada tipo de dado de sistema são apresentados no quadro A16.

Tipo de dado classe	**Parâmetro**

%Binary	MAXLEN, MINLEN
%Boolean	
%Currency	DISPLAYLIST, FORMAT, MAXVAL, MINVAL, VALUELIST
%Date	DISPLAYLIST, FORMAT, MAXVAL, MINVAL, VALUELIST
%Float	DISPLAYLIST, FORMAT, MAXVAL, MINVAL, SCALE, VALUELIST, XSDTYPE
%Integer	DISPLAYLIST, FORMAT, MAXVAL, MINVAL, VALUELIST, XSDTYPE
%List	ODBCDELIMITER
%Name	COLLATION, INDEXSUBSCRIPTS, MAXLEN, XSDTYPE
%Numeric	DISPLAYLIST, FORMAT, MAXVAL, MINVAL, SCALE, VALUELIST
%Status	
%String	COLLATION, DISPLAYLIST, MAXLEN, MINLEN, PATTERN, TRUNCATE, VALUELIST, XSDTYPE
%Time	DISPLAYLIST, FORMAT, MAXVAL, MINVAL, VALUELIST
%TimeStamp	DISPLAYLIST, MAXVAL, MINVAL, VALUELIST

Quadro A16 – parâmetros para os tipos de dados classe.

Tipo de dados no cliente

Para usar dados do Caché com qualquer sistema cliente, como Java, por exemplo, os dados necessitam estar em um formato que o sistema do cliente possa entender. Para isso, o Caché fornece a palavra-chave de classe CLIENTDATATYPE, que especifica a informação do formato dos projetos Caché como uma propriedade para o cliente, como mostra o quadro A17.

Valor	Formato
BINARY	**%Binary** (or any property requiring that there is no Unicode conversion of data)
CURRENCY	**%Currency**
DATE	**%Date**
DOUBLE	**%Float**
INTEGER	**%Boolean** , **%Integer**
LIST	**%List**
NUMERIC	**%Numeric**
VARCHAR	**%Name** , **%String**
TIME	**%Time**
TIMESTAMP	**%TimeStamp**

Quadro A17 – Valores CLIENTDATATYPE

REFERÊNCIA OBJECTSCRIPT

Manipulando Objetos

Criar um novo objeto	set <oref> = ##class(<nomeClasse>).%New()
Abrir um objeto persistente	set <oref> = ##class(<nomeClasse >).%OpenId(<id>)
Verificar a existência do objeto	write ##class(<nomeClasse >).%ExistsId(<id>)
Salvar um objeto	set <st> = <oref>.%Save()
Mostrar erros de uma variável	%Status do $System.OBJ.DisplayError(<st>)
Fechar a referência do objeto	do <oref>.%Close()
Remover um objeto persistente	do ##class(<nomeClasse >).%DeleteId(<id>)
Remover todas as instâncias de uma classe	do ##class(<nomeClasse >).%KillExtent()
Mostrar a propriedade de um objeto	write <oref>.<nomeProp>
Atribuir valor a uma propriedade do objeto	set <oref>.<nomeProp> = <valor>
Fazer referência entre objetos	set <oref>.<nomeProp> = <oref2>
Criar instâncias para uma classe *	do ##class(<nomeClasse>).Populate(<num>) * (disponível apenas para subclasses de %Populate)
Fechar todos os objetos carregados em memória	do $System.OBJ.CloseObjects()
Listar todos os objetos abertos em	do $System.OBJ.ShowObjects()

memória	
Mostrar a estrutura e conteúdo de um objeto	do $System.OBJ.DumpObject(<oref>)
Compilar uma classe	set <st> = $System.OBJ.Compile("<nomeClasse>")
Compilar todas as classes do namespace	do $System.OBJ.CompileAll()

Quadro A18 - referência OBJECTSCRIPT para manipulação de objetos

Listas

Criar uma nova lista	set <lista> = ##class(%Library.ListOfDataTypes).%New()
Inserir elemento no final da lista	do <lista>.Insert(<elem>)
Inserir elemento na lista na posição i	do <lista>.InsertAt(i, <elem>)
Recuperar elemento da lista na posição i	set <oref> = <lista>.GetAt(i)
Mostrar o tamanho da lista	write <lista>.Count()
Remover elemento da lista na posição i	do <lista>.RemoveAt(i)
Remover todos os elementos da lista	do <lista>.Clear()

Quadro A19 - referência OBJECTSCRIPT para manipulação de listas

Arrays

Criar um novo array	set <array> = ##class(%Library.ArrayOfDataTypes).%New()
Inserir elemento no array	do <array>.SetAt(<elem>, <chave>")
Recuperar elemento do array	set <elem> = <array>.GetAt(<chave>)
Mostrar o tamanho do array	write <array>.Count()
Remover elemento do array	do <array>.RemoveAt(<chave>)
Remover todos os elementos do array	do <array>.Clear()

Quadro A20 - referência OBJECTSCRIPT para manipulação de arrays

APÊNDICES

APÊNDICE A - TESTE DE DESEMPENHO BDR x BDOO

APÊNDICE B - PUBLICAÇÕES

APÊNDICE A

TESTE DE DESEMPENHO BDR x BDOO

Introdução

Este apêndice registra uma aplicação prática de benchmark para banco de dados, visando comparar o desempenho de modelos multidimensionais em tecnologia de banco de dados relacional e orientada a objetos, utilizando o BDR Oracle e o BDOO Caché.

A medida do teste de benchmark baseia-se no desempenho de consultas, observando o objetivo do modelo multidimensional, voltado às exigências dos sistemas de apoio a decisão. O teste foi realizado com base no modelo utilizado na etapa de validação da proposta da metodologia (capítulo 6 do presente trabalho).

O modelo utilizou a classe Fatos - Posição Financeira - e as classes de Dimensão – Gerente, Tempo, Conta e Agencia, além da classe de associação ContaCliente, com sua respectiva classe Cliente.

Componentes do sistema

Para a realização dos testes foi selecionado o banco de dados relacional Oracle e o banco de dados orientado a objetos (pós-relacional) Caché. A versão utilizada do Oracle foi a 9i e a versão para o Caché 5.0. O quadro A21 mostra os componentes de sistema e hardware para a realização dos testes.

Componente	Identificação
Processador	Intel Pentium 4, 2.4 MHz, clock 1816 MHz
Memória	1 GB RAM, 512 cache
disco rígido	Seagate 60 GB, 7200 rpm, Controladora IDE Intel
Sistema Operacional	Microsoft Windowns XP Professional
Oracle	Oracle 9i
Caché	Caché 5.0

Quadro A21 - componentes de sistema e hardware.

O benchmark aqui proposto visa documentar medidas de tempo para a execução de uma determinada tarefa, no caso consultas às tabelas Fatos e Dimensão. Estas medidas são utilizadas para comparações entre diferentes sistemas de banco de dados: Oracle e Caché, com as mesmas condições e configurações de hardware e sistema.

As medidas de desempenho utilizadas baseiam-se no tempo de resposta (ou latência), definido como o intervalo de tempo entre uma requisição e a resposta do sistema.

As classes / tabelas foram criadas utilizando-se os *scripts* da figura A2 (para o Caché) e da figura A3 (para o Oracle). Foram utilizados os conceitos da OO na geração das classes no Caché. No Oracle foram utilizados os conceitos do modelo relacional através da geração de tabelas.

```
Class modelo.agencia Extends (%Persistent, %Populate) [ ClassType = persistent, ProcedureBlock ]
{
Property nomeagencia As %String;
Property agrupadoragencia As %Integer;
Property chaveagencia As %Integer;
Property enderecoagencia As %String;
Property formatoagencia As %String;
Property cidadeagencia As cidade;
Index nomeagenciaIndex On nomeagencia;
}
```

```
Class modelo.cidade Extends %SerialObject [ ClassType = serial, ProcedureBlock ]
{
Property nomecidade As %String;
Property descricaoregiao As %String(VALUELIST = ",NORTE,NORDESTE,SUL,SUDESTE,
CENTRO-OESTE");
Property ufcidade As %String(MAXLEN = 02);
}

Class modelo.cliente Extends modelo.pessoa [ ClassType = persistent, ProcedureBlock ]
{
Property cpfcliente As %Integer;
Property datanascimento As %Date;
Relationship asContasCliente As modelo.contacliente [ Cardinality = many, Inverse = oCliente ];
}

Class modelo.conta Extends (%Persistent, %Populate) [ ClassType = persistent, ProcedureBlock ]
{
Property dataabertura As %Date;
Property descricaocategoria As %String;
Property descricaotipoconta As %String;
Property numeroconta As %String;
Relationship asContasCliente As modelo.contacliente [ Cardinality = many, Inverse = aConta ];
Index numerocontaIndex On numeroconta;
}

Class modelo.contacliente Extends (%Persistent, %Populate) [ ClassType = persistent,
ProcedureBlock ]
{
Property TitularConta As %String(MAXLEN = 1) [ Required ];
Relationship aConta As modelo.conta [ Cardinality = one, Inverse = asContasCliente ];
Index aContaIndex On aConta;
Relationship oCliente As modelo.cliente [ Cardinality = one, Inverse = asContasCliente ];
Index oClienteIndex On oCliente;
}

Class modelo.gerente Extends modelo.pessoa [ ClassType = persistent, ProcedureBlock ]
{
Property gerentedepto As %String;
Property gerentedesde As %Date;
}

Class modelo.pessoa Extends (%Persistent, %Populate) [ ClassType = persistent, ProcedureBlock ]
{
Property nomepessoa As %String [ Required ];
Property chavepessoa As %Integer;
Property enderecopessoa As %String;
Property cidadepessoa As cidade [ Required ];
Index nomepessoaIndex On nomepessoa;
}

Class modelo.posicaoconta Extends (%Persistent, %Populate) [ ClassType = persistent,
ProcedureBlock ]
{
Property juroscobrados As %Integer;
Property jurospagos As %Integer;
Property numerotransacoes As %Integer;
Property saldodata As %Integer;
Property saldomediomensal As %Integer;
Property taxascobradas As %Integer;
Property totalcredito As %Integer;
```

```
Property totaldebito As %Integer;
Property tempo As modelo.tempo;
Property agencia As modelo.agencia;
Property gerente As modelo.gerente;
Property conta As modelo.conta;
Index oTempoIndex On tempo;
Index aAgenciaIndex On agencia;
Index oGerenteIndex On gerente;
Index aContaIndex On conta;
}

Class modelo.tempo Extends (%Persistent, %Populate) [ ClassType = persistent, ProcedureBlock ]
{
Property ano As %Integer;
Property chavedata As %Date;
Property chavemes As %Integer(MAXVAL = 12, MINVAL = 1);
Property diasdoano As %Integer;
Property diasemana As %Integer;
Property nomemes As %String(VALUELIST =
",JANEIRO,FEVEREIRO,MARÇO,ABRIL,MAIO,JUNHO,JULHO,AGOSTO,SETEMBRO,OUTUBRO,N
OVEMBRO,DEZEMBRO");
Index chavedataIndex On chavedata;
}
```

Figura A2 – *Script* da geração das classes no Caché.

```
create table gerente(
  oidpessoa number(12,0) not null primary key,
  nomepessoa varchar2(50),
  chavepessoa number(12,0),
  enderecopessoa varchar2(50),
  nomecidade varchar2(50),
  descricaoregiao varchar2(50),
  ufcidade    char(2),
  gerentedepto varchar(50),
  gerentedesde date);

create table cliente(
  oidpessoa number(12,0) not null primary key,
  nomepessoa varchar2(50),
  chavepessoa number(12,0),
  enderecopessoa varchar2(50),
  nomecidade varchar2(50),
  descricaoregiao varchar2(50),
  ufcidade    char(2),
  cpfcliente number(12,0),
  datanascimento date);

create table tempo(
  oiddata number(12,0) not null primary key,
  ano number(12,0),
  chavedata date,
  chavemes  number(12,0),
  diasdoano number(12,0),
  diasemana number(12,0),
  nomemes  varchar2(20) );

create table agencia(
  oidagencia number(12,0) not null primary key,
```

```
  nomeagencia varchar2(50),
  agrupadoragencia number(12,0),
  chaveagencia number(12,0),
  enderecoagencia varchar2(50),
  nomecidade varchar2(50),
  descricaoregiao varchar2(50),
  ufcidade    char(2),
  formatoagencia varchar2(50));

create table conta(
  oidconta number(12,0) not null primary key,
  dataabertura date,
  descricaocategoria varchar2(50),
  descricaotipoconta varchar2(50),
  numeroconta char(15)) ;

create table contacliente(
  oidcontacliente number(12,0) not null primary key,
  titularconta varchar2(50),
  conta number(12,0),
  cliente number(12,0)
  constraint fk_contacliente_conta foreign key (conta) references conta(oidconta),
  constraint fk_contacliente_cliente foreign key (cliente) references cliente(oidpessoa));

create table posicaoconta(
  oidposicaoconta number(12,0) not null primary key,
  juroscobrados number(16,2),
  jurospagos number(16,2),
  numerotransacoes number(12,0),
  saldodata number(16,2),
  saldomediomensal number(16,2),
  taxascobradas number(16,2),
  totalcredito number(16,2),
  totaldebito number(16,2),
  gerente number(12,0) not null,
  tempo number(12,0) not null,
  agencia number(12,0) not null,
  conta number(12,0) not null);
  constraint fk_posicaoconta_gerente foreign key (gerente) references gerente(oidpessoa),
  constraint fk_posicaoconta_tempo foreign key (tempo) references tempo(oiddata),
  constraint fk_posicaoconta_agencia foreign key (agencia) references agencia(oidagencia),
  constraint fk_posicaoconta_conta foreign key (conta) references conta(oidconta));
  create index idx_nome on agencia(nomeagencia);
  create index idx_numeroconta on conta(numeroconta);
  create index idx_cliente on contacliente(cliente);
  create index idx_nomegerente on gerente(nomepessoa);
  create index idx_nomecliente on cliente(nomepessoa);
  create index idx_data on tempo(chavedata);
```

Figura A3 – *Script* da geração das tabelas no Oracle.

O quadro A22 mostra o número de registros de cada classe / tabela utilizado para a realização do teste.

Tabela	**Nº de registros**
Fatos – PosicaoConta	2.000.000
Dimensão – Conta	200.000
Dimensão – Tempo	400
Dimensão – Agencia	20
Dimensão – Gerente	40

Dimensão – ContaCliente	240.000
Cliente	180.000

Quadro A22 – Número de registros nas dimensões e fatos.

Recuperação de Dados

Para a aplicação dos testes foram executadas dezessete consultas.

O quadro A23 mostra a referência das consultas de recuperação de dados executadas, com o número de registros encontrados em cada uma das consultas, e os respectivos tempos decorridos no Caché e no Oracle para sua execução. O quadro 04 registra a comparação dos tempos de testes.

Resultados

Os tempos de execução das tabelas mostra que o Caché apresenta-se ligeiramente superior nas consultas que fazem referência a objetos, pois o modelo relacional obrigatoriamente realiza joins para obter os dados resultantes. Nas consultas em que foi necessária a indicação das referências no modelo OO, o banco relacional demonstrou desempenho superior.

O quadro A24 mostra um resumo geral das consultas, identificando uma pequena diferença de desempenho superior ao modelo OO.

Portanto, o banco de dados Caché apresenta-se como uma solução viável para o modelo multidimensional, gerenciando um grande volume de dados nos limites esperados para um ambiente de DW.

	Caché		Oracle		
	Query	**Tempo (ms)**	**Query**	**Tempo (ms)**	**Nº de Linhas**
01	select agencia, tempo, avg(saldodata) as media_saldodata from modelo.posicaoconta group by agencia, tempo	61030	select agencia, tempo, avg(saldodata) as media_saldodata from posicaoconta group by agencia, tempo;	55080	8000
02	select tempo->nomemes, tempo->ano,agencia, agencia->nomeagencia, avg(saldodata) as media_saldodata from modelo.posicaoconta group by tempo->nomemes, tempo->ano,agencia, agencia->nomeagencia order by tempo->nomemes, tempo->ano	91030	select nomemes, ano,agencia, nomeagencia, avg(saldodata) as media_saldodata from posicaoconta, tempo, agencia where posicaoconta.agencia=agencia.oidagencia and posicaoconta.tempo = tempo.oiddata group by nomemes, ano,agencia, nomeagencia order by nomemes, ano;	44020	180
03	select gerente->nomepessoa, avg(juroscobrados) as media_juros_cobrados, avg(jurospagos) as media_juros_pagos from modelo.posicaoconta group by gerente->nomepessoa order by gerente->nomepessoa	80230	select nomepessoa, avg(juroscobrados) as media_juros_cobrados, avg(jurospagos) as media_juros_pagos from posicaoconta, gerente where gerente = oidpessoa group by nomepessoa order by nomepessoa;	22060	40
04	select ocliente->cpfcliente, ocliente->nomepessoa, avg(saldodata) as media_saldo from modelo.contacliente inner join modelo.posicaoconta on modelo.contacliente.aconta = modelo.posicaoconta.conta where tempo between 3001 and 3150 group by ocliente->cpfcliente, ocliente->nomepessoa having avg(saldodata) > 1000 order by ocliente->nomepessoa	92980	select cpfcliente, nomepessoa, avg(saldodata) as media_saldo from contacliente, posicaoconta, cliente where contacliente.conta = posicaoconta.conta and contacliente.cliente = cliente.oidpessoa and tempo between 3001 and 3150 group by cpfcliente, nomepessoa having avg(saldodata) > 1000 order by nomepessoa;	25040	312

05	select ocliente->cpfcliente, ocliente->nomepessoa, avg(saldodata) as media_saldo from modelo.contacliente inner join modelo.posicaoconta on modelo.contacliente.aconta = modelo.posicaoconta.conta group by ocliente->cpfcliente, ocliente->nomepessoa order by ocliente->nomepessoa	79870	select cpfcliente, nomepessoa, avg(saldodata) as media_saldo from contacliente inner join posicaoconta on contacliente.conta = posicaoconta.conta inner join cliente on cliente.oidpessoa = contacliente.cliente group by cpfcliente, nomepessoa order by nomepessoa;	55010	312
06	select agencia, tempo->nomemes, tempo->ano, avg(saldodata) as media_saldodata from modelo.posicaoconta where (agencia between 210 and 230) and (tempo between 3200 and 3300) group by agencia, tempo->nomemes, tempo->ano order by agencia, tempo->nomemes, tempo->ano	21070	select agencia, nomemes, ano, avg(saldodata) as media_saldodata from posicaoconta, tempo where (agencia between 210 and 230) and (tempo between 3200 and 3300) and posicaoconta.tempo = tempo.oiddata group by agencia, nomemes, ano order by agencia, nomemes, ano;	12060	22
07	select oCliente->cpfcliente, modelo.posicaoconta.conta, count(numerotransacoes) as transacoes from modelo.contacliente inner join modelo.posicaoconta on modelo.contacliente.aConta = modelo.posicaoconta.conta group by oCliente->cpfcliente, modelo.contacliente.aConta	73010	select cpfcliente, posicaoconta.conta, count(numerotransacoes) as transacoes from contacliente inner join posicaoconta on contacliente.Conta = posicaoconta.conta inner join cliente on cliente.oidpessoa = contacliente.cliente group by cpfcliente, posicaoconta.Conta;	57050	305
08	select oCliente->cpfcliente, oCliente->nomepessoa, avg(saldodata) as media_saldo from modelo.contacliente inner join modelo.posicaoconta on modelo.contacliente.aConta = modelo.posicaoconta.conta group by oCliente->cpfcliente, oCliente->nomepessoa order by oCliente->nomepessoa	79490	select cpfcliente, nomepessoa, avg(saldodata) as media_saldo from contacliente, posicaoconta, cliente, conta where posicaoconta.conta=conta.oidconta and conta.oidconta = contacliente.conta and contacliente.cliente = cliente.oidpessoa group by cpfcliente, nomepessoa order by nomepessoa;	73080	312
09	select agencia, tempo, avg(saldodata) as media_saldodata from modelo.posicaoconta	29160	select agencia, tempo, avg(saldodata) as media_saldodata from posicaoconta	23010	2010

	where (agencia between 200and 210) and (tempo between 3100 and 3300) group by agencia, tempo order by agencia, tempo		where (agencia between 200 and 210) and (tempo between 3100 and 3300) group by agencia, tempo order by agencia, tempo;		
10	select tempo->chavedata, agencia, conta->numeroconta, avg(saldodata) as media_saldodata from modelo.posicaoconta where (agencia between 200 and 210) and (tempo between 3100 and 3300) group by tempo->chavedata, agencia, conta->numeroconta order by tempo->chavedata, agencia, conta->numeroconta	64320	select chavedata, agencia, numeroconta, avg(saldodata) as media_saldodata from posicaoconta, tempo, conta where posicaoconta.tempo = tempo.oiddata and posicaoconta.conta = conta.oidconta and (agencia between 200 and 210) and (tempo between 3100 and 3300) group by chavedata, agencia, numeroconta order by chavedata, agencia, numeroconta;	143090	21370
11	select agencia, agencia->nomeagencia, tempo, tempo->nomemes, avg(saldodata) as media_saldodata from modelo.posicaoconta group by agencia, agencia->nomeagencia, tempo, tempo->nomemes order by agencia, tempo, tempo->nomemes	108340	select agencia, nomeagencia, tempo, nomemes, avg(saldodata) as media_saldodata from posicaoconta, agencia, tempo where posicaoconta.agencia=agencia.oidagencia and posicaoconta.tempo = tempo.oiddata group by agencia, nomeagencia, tempo, nomemes order by agencia, tempo, nomemes;	135070	8000
12	select agencia, agencia->nomeagencia, conta->numeroconta, avg(saldodata) as media_saldodata from modelo.posicaoconta group by agencia, agencia->nomeagencia, conta->numeroconta order by agencia, conta->numeroconta	115250	select agencia, agencia.nomeagencia, conta.numeroconta, avg(saldodata) as media_saldodata from posicaoconta, agencia, conta where posicaoconta.agencia = agencia.oidagencia and posicaoconta.conta = conta.oidconta group by agencia, nomeagencia, numeroconta order by agencia, numeroconta;	194030	9200
13	select agencia, agencia->nomeagencia, nomepessoa, avg(saldodata) as media_saldodata from modelo.posicaoconta inner join modelo.contacliente on modelo.contacliente.aconta = modelo.posicaoconta.conta inner join modelo.cliente on modelo.contacliente.ocliente = modelo.cliente.id group by agencia, agencia->nomeagencia, nomepessoa	130280	select agencia, nomeagencia, nomepessoa, avg(saldodata) as media_saldodata from posicaoconta inner join agencia on posicaoconta.agencia = agencia.oidagencia inner join conta on conta.oidconta = posicaoconta.conta inner join contacliente on contacliente.conta = conta.oidconta inner join cliente on contacliente.cliente = cliente.oidpessoa group by agencia, nomeagencia, nomepessoa order by agencia, nomepessoa;	222050	6220

	order by agencia, nomepessoa				
14	select agencia, agencia->nomeagencia, gerente->nomepessoa, avg(saldodata) as media_saldodata from modelo.posicaoconta group by agencia, agencia->nomeagencia, gerente->nomepessoa order by agencia, gerente->nomepessoa	99160	select agencia, nomeagencia, nomepessoa, avg(saldodata) as media_saldodata from posICAOconta, agencia, gerente where posicaoconta.agencia = agencia.oidagencia and posicaoconta.gerente = gerente.oidpessoa group by agencia, nomeagencia, nomepessoa order by agencia, nomepessoa;	308040	800
15	select oCliente->cpfcliente, oCliente->nomepessoa, agencia->nomeagencia, avg(saldodata) as media_saldo from modelo.posicaoconta inner join modelo.contacliente on modelo.posicaoconta.conta=modelo.contacliente.aConta group by oCliente->cpfcliente, oCliente->nomepessoa, agencia->nomeagencia order by oCliente->nomepessoa	110490	select cpfcliente, nomepessoa, nomeagencia, avg(saldodata) as media_saldo from contacliente, posicaoconta, cliente, conta, agencia where posicaoconta.conta=conta.oidconta and conta.oidconta = contacliente.conta and contacliente.cliente = cliente.oidpessoa and posicaoconta.agencia = agencia.oidagencia group by cpfcliente, nomepessoa, nomeagencia order by nomepessoa;	195040	6240
16	select oCliente->cpfcliente, oCliente->nomepessoa, agencia->nomeagencia, avg(saldodata) as media_saldo from modelo.posicaoconta inner join modelo.contacliente on modelo.posicaoconta.conta=modelo.contacliente.aConta where agencia->cidadeagencia_nomecidade = 'FLORIANOPOLIS' group by oCliente->cpfcliente, oCliente->nomepessoa, agencia->nomeagencia order by oCliente->nomepessoa	115100	select cpfcliente, nomepessoa, nomeagencia, avg(saldodata) as media_saldo from contacliente, posicaoconta, cliente, conta, agencia where posicaoconta.conta=conta.oidconta and conta.oidconta = contacliente.conta and contacliente.cliente = cliente.oidpessoa and posicaoconta.agencia = agencia.oidagencia and agencia.nomecidade = 'FLORIANOPOLIS' group by cpfcliente, nomepessoa, nomeagencia order by nomepessoa;	127040	2496
17	select oCliente->cpfcliente, oCliente->nomepessoa, tempo->chavedata, avg(saldodata) as media_saldo from modelo.posicaoconta inner join modelo.contacliente on modelo.posicaoconta.conta=modelo.contacliente.aConta where (tempo->chavedata between '01/01/2005' and '01/12/2005')	49070	select cpfcliente, nomepessoa, chavedata, avg(saldodata) as media_saldo from posicaoconta inner join contacliente on posicaoconta.conta=contacliente.Conta inner join cliente on contacliente.cliente = cliente.oidpessoa inner join tempo on posicaoconta.tempo = tempo.oiddata where (chavedata between '01/01/2005' and	84070	2808

	group by oCliente->cpfcliente, oCliente->nomepessoa, tempo->chavedata order by oCliente->nomepessoa, tempo->chavedata		'01/12/2005') group by cpfcliente, nomepessoa, chavedata order by nomepessoa, chavedata;		

Quadro A23 – tempos de execução das consultas nos BDs.

Consulta	Tempo Caché (ms)	Tempo Oracle (ms)	Caché:Oracle
01	61030	55080	1 : 1.1
02	91030	44.020	1 : 2
03	80230	22.060	1 : 3.6
04	92980	25040	1 : 3.7
05	79870	55010	1 : 1.5
06	21070	12060	1 : 1.7
07	73010	57050	1 : 1.2
08	79490	73080	1 : 1.1
09	29160	23010	1 : 1.2
10	64320	143090	2.2 : 1
11	108340	135070	1.2 : 1
12	115250	194030	1.6 : 1
13	130280	222050	1.7 : 1
14	99160	308040	3.1 : 1
15	110490	195040	1.7 : 1
16	115100	127040	1.1 : 1
17	49070	84070	1.7 : 1
Total	1399880	**1774840**	**1.26 : 1**

Quadro A24 – comparativo de tempos do Caché e Oracle

APÊNDICE B

PUBLICAÇÕES

BORBA, S. F. P. ; MORALES, A. B. T. Uma proposta de metodologia para implantação de modelos dimensionais em banco de dados orientado a objetos. In: 3º CONTECSI Congresso Internacional de Gestão de Tecnologia e Sistemas de Informação. São Paulo: FEA USP, 2006. Anais do 2º CONTECSI Congresso Internacional de Gestão de Tecnologia e Sistemas de Informação. São Paulo: FEA USP, 2006. v. 1. p. 47-47.

BORBA, S. F. P. ; MORALES, A. B. T. Extensão da UML para modelos dimensionais mapeados em banco de dados orientado a objetos. In: 2º CONTECSI Congresso Internacional de Gestão de Tecnologia e Sistemas de Informação, 2005, São Paulo. Anais do 2º CONTECSI Congresso Internacional de Gestão de Tecnologia e Sistemas de Informação. São Paulo: FEA USP, 2005. v. 1. p. 159-159.

BORBA, S. F. P. ; MORALES, A. B. T. Standard ODMG to dimensional models. In: IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005), 2005. Anais do IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005), 2005. v. 1. p. 299-306.

BORBA, S. F. P. ; MORALES, A. B. T. Extending the UML for dimensional models in object-oriented database. In: 16th International Conference on Database and Expert Systems Applications - DEXA '05, 2005, Copenhague. IEEE Computer Society Press, 2005.

BORBA, S. F. P. ; MORALES, A. B. T. Modelos dimensionales en Banco de Datos Orientado a Objetos. In: IV Congreso Internacional de Informatica Y Computacion de La ANIEI, 2005, Torreon. Anais do IV Congreso Internacional de Informatica Y Computacion de La ANIEI, 2005.