2º Contecsi – Congresso Internacional de Gestão da Tecnologia e Sistemas de Informação /
Internacional Conference on Information Systems and Technology Management
01-03 de Junho de 2005 São Paulo/SP Brasil

Extensão da UML para modelos dimensionais mapeados em banco de dados orientado a objetos

Sueli de Fatima Poppi Borba (Instituto Superior de Ciências Exatas, Agrárias, Tecnológicas e Geociências Universidade Paranaense – Unipar) - sueli@unipar.br
Aran Bey Tcholakian Morales (Departamento de Engenharia da Produção, Universidade Federal de Florianópolis) - aran@stela.ufsc.br

Este artigo insere-se no contexto atual do paradigma da orientação a objetos para a modelagem dimensional e banco de dados. O trabalho apresenta uma extensão da UML para o modelo dimensional, abordando os conceitos dessa linguagem e a utilização do diagrama de classes para a notação do modelo dimensional. A partir dessa representação, o modelo é mapeado para banco de dados orientado a objetos através do padrão ODMG e validado no banco de dados pós-relacional Caché. O trabalho busca apresentar a convergência do paradigma da orientação a objetos com a modelagem dimensional, utilizada na tecnologia de data warehouse. São apresentados os conceitos e objetivos da modelagem dimensional, incluindo os esquemas Star e Snowflake. Para a orientação a objetos, o trabalho aborda os conceitos aplicados a banco de dados, além de abranger a categorização de classes e seus relacionamentos, identificadores e descritores de objeto, através da representação no diagrama de classes da UML. O artigo relata a seqüência das etapas para o desenvolvimento do modelo dimensional, a partir do modelo operacional, sua posterior representação na extensão da UML e finalmente, seu mapeamento. Para validar as etapas do processo operacional ao mapeamento, o artigo utiliza um estudo de caso partindo de um modelo operacional, gerando o modelo dimensional através dos conceitos de hierarquias de classificação e desnormalização dos modelos Star e Snowflake. A complementação da validação é obtida através da utilização da ODL do padrão ODMG para gerar as classes, atributos e relacionamentos, para posterior implementação no banco de dados Caché.

Palavras-chave: Banco de dados, modelagem dimensional, orientação 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. Assim, alguns conceitos e ferramentas surgem com o objetivo de auxiliar esse processo, incluindo a tecnologia de Data Warehousing que é voltada para o estudo de técnicas e ferramentas utilizadas em aplicações que envolvam análise intensiva de dados, de forma a prover flexibilidade e agilidade em sua gerência (Inmon, 2001).

Muitos estudos e pesquisas voltam seus objetivos para a modelagem multidimensional, e alguns novos modelos têm sido propostos. Nos últimos anos, observa-se que a comunidade técnico-científica está direcionando seus trabalhos para um paradigma que consolida-se, gradativamente, no processo conceitual de desenvolvimento de sistemas, a *Unified Modeling Language* (UML) e a orientação a objetos. A realidade dos bancos de

dados orientados a objetos vem de encontro a essas 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. Os objetos podem ser reutilizados e compartilhados entre aplicações, representando significativamente a realidade.
Porém, a modelagem multidimensional, aliada às novas tecnologias de banco de dados orientados a objetos, não possui uma metodologia padrão de implementação, considerando que praticamente todas as abordagens consideram a modelagem multidimensional aplicada ao modelo relacional.

1.1 Justificativas e Relevância do Estudo

Nesse 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 Sistemas Gerenciadores de Banco de Dados (SGBD), que permitem o tratamento de novos tipos de dados. O modelo orientado a objetos implementa requisitos da abordagem Orientada a Objetos (OO), observando que anteriormente, esses requisitos eram apenas conceituados, mas não implementados em sistemas de banco de dados.
Portanto, aliar a tecnologia de Sistemas Gerenciadores de Banco de Dados Orientados a Objetos (SGBDOO) à tecnologia de data warehouse (DW) mostra-se como uma tendência 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.
Muitos trabalhos desenvolvidos na área de banco de dados estão direcionados à modelagem multidimensional. Alguns desses trabalhos voltam-se à modelagem multidimensional no 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 *Star* e buscam seu mapeamento a partir do modelo Entidade-Relacionamento (ER) (Monteiro, 1998). Outros trabalhos ainda voltam-se para a representação das principais propriedades da modelagem dimensional no nível conceitual, gerando notações específicas (Golfarelli et all, 1998; Sapia, 1998; Hüsemann et al., 2000).
Luján-Mora et al. (2002, 2004) observam que a UML tem sido aceita como a linguagem padrão para modelagem orientada a objetos e que alguns poucos esforços buscam estender a modelagem multidimensional a partir dessa concepção, verificando a aplicabilidade desse novo paradigma. Os trabalhos voltados para essa nova concepção buscam minimizar o trabalho dos desenvolvedores no sentido de conhecerem novas metodologias e nomenclaturas para cada novo projeto a ser modelado. Porém, muitos dos trabalhos encontrados nesse contexto utilizam os conceitos da orientação a objetos para a representação de seus diagramas (Nguyen et al., 2000; Trujillo, 2000), enquanto apenas alguns abordam propriedades dinâmicas da modelagem dimensional (Luján-Mora et al., 2002). Embora abordem os aspectos da orientação a objetos, observa-se que os trabalhos não oferecem o mapeamento para o modelo de banco de dados orientado a objetos.
Portanto, o objetivo desse trabalho é definir um mapeamento do modelo dimensional para um banco de dados orientado a objetos, baseando-se na modelagem gerada através da UML, aplicado na emergente tecnologia de banco de dados, fundamentada na

persistência de objetos.
O presente trabalho está organizado em 5 seções. A seção 2 aborda aspectos conceituais do modelo de dados multidimensional, relacionando seus elementos básicos e abordando a modelagem segundo o paradigma da orientação a objetos, suas principais características e requisitos. A seção 3 apresenta a utilização da linguagem UML no modelo multidimensional. A seção 4 apresenta o mapeamento do modelo em banco de dados orientado a objetos. Finalmente, na seção 5 são apresentadas as conclusões.

2 MODELAGEM DIMENSIONAL

O processo de modelagem de dados busca transformar modelos de dados orientados a processos, considerados modelos funcionais, em modelos de dados orientados a negócios, os modelos dimensionais. 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 dimensional.
Singh (2001) lembra que "em contraste com os sistemas OLTP (*On Line Transaction Processing*) 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". Kimball (1998) afirma que todo modelo dimensional é composto de uma tabela com uma chave composta de várias partes, chamada Tabela de Fatos e um grupo de tabelas menores, chamadas Tabelas de Dimensão. Cada tabela de Dimensão possui uma Pk - *Primary Key* de parte única, que corresponde exatamente a uma das partes da chave da tabela de Fatos.
A modelagem multidimensional apresenta os seguintes elementos básicos (Kimball, 2002, Machado, 2000 e Singh, 2001):

- 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 dimensional;
- 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.

O objetivo da modelagem dimensional 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, reduzindo a complexidade do banco de dados e minimizando o número de junções exigidas em consultas do usuário.
A representação do modelo dimensional através do *Star Schema* (Kimball, 1997) 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 dessas tabelas possui apenas uma junção com a tabela central. 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. O *Snowflake Schema* é a decomposição de um *Star Schema*, de uma ou mais dimensões, que possuem hierarquias entre seus membros. Os relacionamentos nesse modelo são de muitos-para-um entre os membros de uma dimensão, formando, através dos relacionamentos entre tabelas de dimensão, uma hierarquia.

2.1 Orientação a objetos

Trujillo e Palomar (1998) afirmam que o paradigma da orientação a objetos 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.

A orientação a objetos é um paradigma de linguagens e de 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. Também pode ser definido como a propriedade pela qual uma variável pode conter um apontador para diferentes classes, em diferentes instantes, assumindo formas distintas.
- 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.

Os conceitos da orientação a objetos são implementados pelas classes e objetos, que são os principais componentes da orientação a objetos, além de seus atributos, incluindo o identificador de objetos, os relacionamentos e as operações.

Booch et al. (2000) definem classe como "uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamento e semântica". Uma classe existe como um encapsulamento de atributos e métodos.

Os objetos têm uma identidade, um estado e um comportamento, são instâncias das classes. Um objeto possui um estado interno e um comportamento. O estado interno é composto por variáveis que podem armazenar dados e o comportamento, que 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).

Um atributo é um valor de dados guardado pelos objetos de uma classe. Cada atributo possui um valor para cada instância dos objetos (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 (Booch et al., 2000). Quando um objeto é criado, ele recebe um identificador – OID (Identificador de objetos), gerando a característica de unicidade para aquele objeto, diferenciado-o de todos os demais (Muller, 2002).

O modelo de objetos, além de apresentar os objetos individuais também mostra como os

objetos se relacionam entre si. Booch et al. (2000) relacionam três tipos principais de relacionamentos na UML: dependência, associação e generalização.
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, conhecidas como relacionamentos subclasse / superclasse. Esses relacionamentos são itens estáticos, representados, normalmente, em diagramas de classe da UML.
A associação é um relacionamento em que se representa a conexão de um objeto a outro. No entanto, outro tipo de associação é a agregação, que segundo Booch et al. (2000), 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. Uma agregação é um objeto que é composto de outros objetos.
A ligação da Generalização/Especialização relaciona duas classes (ou metaclasses). A classe mais geral é chamada superclasse e a mais específica, referenciada como subclasse. Como conseqüência deste tipo de ligação, obtém-se a herança. Se for permitido ter mais de uma superclasse, pode-se ganhar heranças múltiplas. 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.
Uma outra forma de interação entre os objetos é quando, através de um estímulo, um determinado objeto provoca uma reação em outro objeto (Giovinazzo, 2000). Uma classe aceita certos tipos de mensagens das instâncias de outras classes, que ativam a execução de métodos (consultas, atualizações, cálculos, etc.). A inclusão de métodos no modelo de dados ajuda modelar o comportamento junto com os dados. Métodos facilitam a implementação de funções de agregação complexas (Abelló et al., 2000). Booch et al. (2000) lembram que 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.

2.1.1 A UML no modelo dimensional

Trujillo et al. (2001) acreditam que a orientação a objeto com UML pode fornecer uma notação adequada para modelar todos os aspectos de um sistema de data warehouse, desde os requisitos de usuário até a implementação.
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).
Considerando o objetivo do trabalho proposto e suas etapas, a abordagem utiliza o diagrama de classes para representação da etapa da modelagem dimensional utilizando uma extensão da UML.
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).
Trujillo e Palomar (1998) estudam 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 de Dimensão e classes de Fatos. Classe Dimensão deve conter objetos Dimensão que tem

características dos dados efetivos, enquanto Classe Fatos possui objetos Fatos. Classe Fatos é construída a partir da classe Dimensão.

3 REPRESENTAÇÃO DA UML NO MODELO MULTIDIMENSIONAL

Kimball (1997) sugere a geração do modelo dimensional a partir do modelo ER, através das seguintes etapas:

- converter um diagrama ER em um grupo de diagramas dimensionais, separando-os nos diversos processos de negócios e modelando-os separadamente;
- selecionar os relacionamentos muitos-para-muitos do modelo ER contendo fatos não chave numéricos e aditivos e designá-los como uma tabela Fatos.
- desnormalizar todas as tabelas restantes em tabelas com chaves simples que se conectem diretamente com a tabela Fatos. Essas tabelas tornam-se tabelas Dimensão.

Considerando que em uma análise do modelo dimensional, as medidas são critérios de avaliação e as dimensões são o que está sendo analisado, deve-se identificar as medidas e dimensões de acordo com os requisitos estabelecidos pelo usuário. 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. Observando tais considerações, a partir de um modelo exemplo Entidade-Relacionamento de Pedidos de Clientes, representado na figura 1, define-se o modelo dimensional correspondente.

3.1 Categorizando Dimensões

A partir do modelo dimensional proposto, 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 e proposto por Luján-Mora et al. (2002). Para exemplificar a extensão da UML para a modelagem dimensional a partir do modelo da figura 1, a entidade de transação é definida, ou seja, a tabela Fatos, com base nos pedidos de produtos de clientes, identificada como a questão de negócio a ser analisada. A tabela fatos Pedido é gerada a partir das entidades Pedido e Item_Pedido do ambiente operacional.

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 Vendedor e Cliente.

3.2 Identificando Hierarquias de classificação

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.

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 Graph*) na classe dimensão. A estrutura DAG pode representar ou um caminho alternativo ou múltiplas hierarquias de classificação.

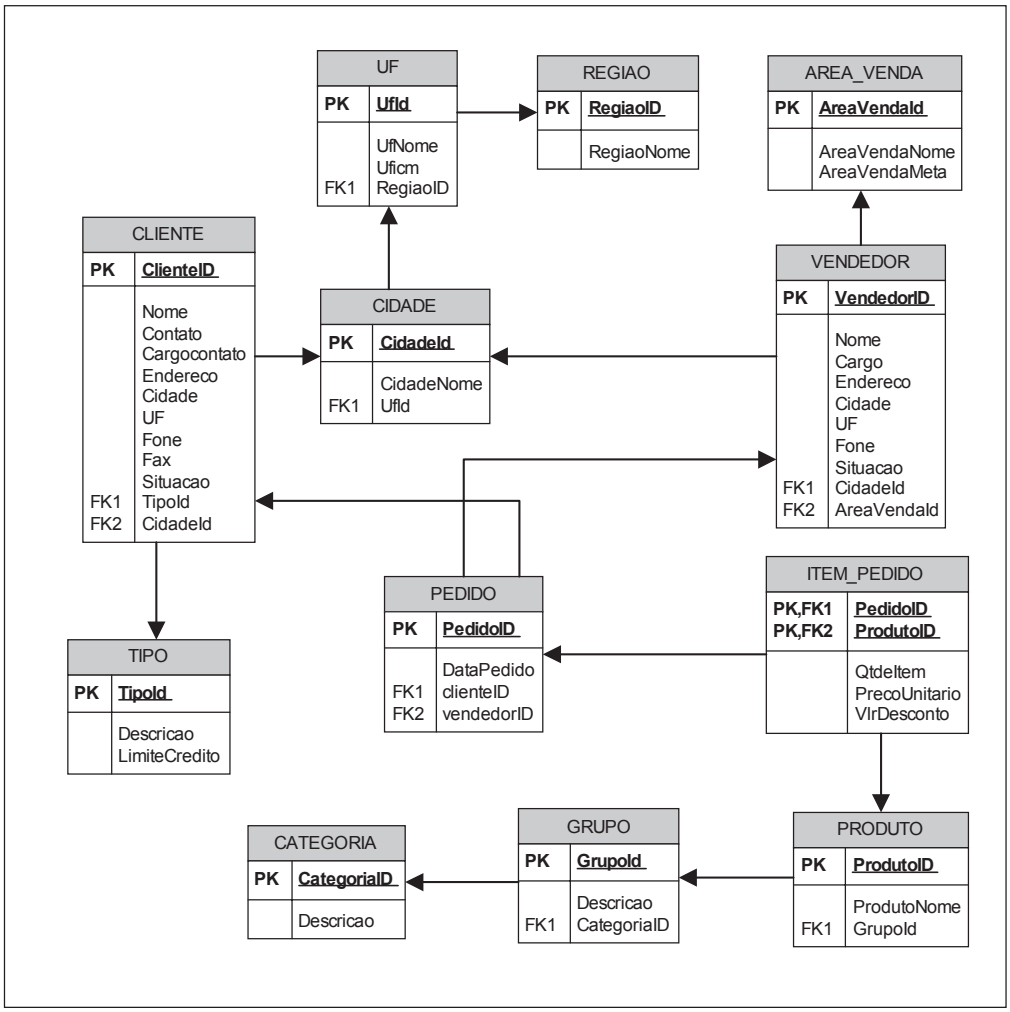

Figura 1. Modelo Entidade-Relacionamento de Pedidos de Clientes.

O modelo da figura 1 apresenta as hierarquias explicitamente embutidas de cidade-uf-região como sendo o enquadramento da cada cidade em sua respectiva região e a hierarquia entre produto-grupo-categoria.

Considerando que para uma Dimensão, uma classe básica representa todos os níveis de hierarquia de classificação, as Classes Vendedor, Cliente e Produto encontram-se devidamente definidas. As classes dimensão Vendedor e Cliente utilizam a hierarquia de classificação da classe Cidade e definem individualmente seu DAG. A estrutura do DAG representa um caminho alternativo e múltiplas hierarquias de classificação. Assim, a classe dimensão Cidade é uma associação com as classes dimensão Vendedor e Cliente, que compartilham um caminho de hierarquia de classificação.
A principal característica que diferencia o modelo ER do modelo dimensional é a desnormalização do modelo dimensional. Para gerar o modelo dimensional, portanto, faz-se necessário desmontar hierarquias, anexando suas informações relevantes às tabelas Fatos ou Dimensão correspondente. Para tanto, são classificadas as entidades no modelo e suas hierarquias. Essas hierarquias são separadas e finalmente, são agregados os dados de transação. No exemplo, a entidade Cidade é desmontada nas entidades Vendedor e Cliente, agregando atributos descritivos de Cidade, assim como as entidades Categoria e Grupo são desmontadas na entidade Produto.

3.3 Identificadores

Nassu e Setzer (1999) lembram que quando se cria um objeto, ele recebe um identificador (OID), que deve ser único para estabelecer relacionamentos entre objetos, oferecendo uma maneira de se recuperar os objetos do banco de dados. As classes básicas, inclusive a classe dimensão, que pertencem à hierarquia de classificação devem conter um atributo identificador explicitamente definido. Para isso, coloca-se a restrição {OID} próximo a um atributo em cada classe.

Além dos identificadores de objeto, podemos associar também os descritores a esses objetos. Esses atributos descritores, com a restrição {D} especificada, além de serem 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.

No modelo proposto, todas as classes envolvidas apresentam um identificador. A classe fatos Pedido e a classe dimensão Tempo, no exemplo, não apresentam atributos descritores. O quadro 1 referencia os atributos identificadores e descritores existentes nas classes.

Quadro 1. Identificadores e descritores das classes do modelo de Pedidos de clientes

Classe	Identificador (OID)	Descritor (D)
Pedido	PedidoId	-
Tempo	DataId	-
Vendedor	VendedorId	VendedorNome
Cliente	ClienteId	ClienteNome
Produto	ProdutoId	ProdutoNome

3.4 Granularidade e medidas de derivação

A granularidade do fato é o nível de detalhe com que é gravado. Se os dados são efetivamente analisados, devem estar no mesmo nível de granularidade. Como uma regra geral, os dados poderiam ser mantidos com o maior nível de detalhamento, considerando que a partir desse detalhamento, os dados podem ser sumarizados, gerando um nível mais baixo de granularidade.

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 fatos.

Trujillo et all. (2001) consideram aplicar aditividade ou sumarização para medidas ao longo de dimensões. Uma medida é aditivada ao longo de uma dimensão se for possível usar o operador SUM para agregar valores de atributo ao longo de todas as hierarquias definidas naquela dimensão.

O quadro 2 mostra as medidas derivadas: VlrTotal, QtdeTotal, DescontoTotal e NumeroItens, e suas regras de derivação, geradas a partir das tabelas de dimensão e implementadas na classe fatos. As medidas derivadas referenciadas no exemplo são aditivadas através da utilização dos operadores SUM e COUNT na agregação dos valores de medidas de dimensões.

Quadro 2. Regras de derivação e medidas derivadas para o modelo de Pedidos de clientes

Atributo(s) origem	Regra de derivação	Medida derivada
QtdeItem, PrecoUnitario	SUM(QtdeItem * PrecoUnitario)	VlrTotal
QtdeItem	SUM(QtdeItem)	QtdeTotal
VlrDesconto	SUM(VlrDesconto)	DescontoTotal

PedidoID	COUNT(PedidoID)	NumeroItens

3.5 Relacionamentos

A entidade de classificação do modelo caracteriza os relacionamentos muitos-para-muitos. Segundo o modelo proposto de Pedidos de Clientes, esses relacionamentos são identificados entre as entidades Pedido e Produto (entidade associativa Item_Pedido).

A flexibilidade da agregação compartilhada na UML permite representar o relacionamento muitos-para-muitos entre a classe fatos Pedido e a classe dimensão Produto, registrando que uma instância do objeto Pedido pode ser relacionada a uma ou mais instâncias do objeto Produto. Os relacionamentos entre a classe fatos Pedido e as classes dimensão Vendedor, Cliente e Tempo são associações de exatidão, especificando que para cada objeto na classe fatos existe apenas um objeto nas classes dimensão.

Portanto, classes de fatos são consideradas como classes compostas de um relacionamento de agregação compartilhada de n classes de dimensão. 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).

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

Para o modelo proposto as classes AreaVenda e Vendedor formam uma associação de não exatidão pois um objeto instanciado da AreaVenda pode referenciar-se a mais de uma instância do objeto Vendedor. Se a multiplicidade entre AreaVenda e Vendedor for * e não 1..*, AreaVenda pode estar relacionado com nenhum Vendedor, formando um relacionamento sem correspondência.

3.6 Refinamento do modelo

A partir das considerações acima referenciadas sobre as características conceituais da UML, e considerando a perspectiva do modelo dimensional, em que o fator tempo é essencialmente importante, o modelo proposto é finalizado com a dimensão Tempo, visando atender os requisitos de pesquisa em determinados períodos da informação. A dimensão Tempo é caracterizada em alguns aspectos pela exatidão e completeza, como por exemplo, um relacionamento de não exatidão entre as Classes Mês e Estação, pois um mês pode pertencer a mais de uma estação. Também pode ser representada a completeza nesse relacionamento, em que todos os registros de estação formam um ano e todas as estações que formam o ano devem ser registradas.

Observando as considerações de Trujillo et al. (2001) quanto a categorização de dimensões para modelar características adicionais para subtipos de entidades no modelo conceitual multidimensional, o modelo proposto utiliza o relacionamento do tipo generalização/especialização, seguindo a restrição de que apenas classes de dimensão podem pertencer a uma hierarquia de classificação e especialização ao mesmo tempo.

O modelo agrega as funcionalidades descritas. O diagrama representa a especialização

da classe Pessoa para as classes dimensão Vendedor e Cliente, sob a análise dos atributos diferenciados para as classes especializadas. A dimensão Tempo engloba os atributos dos relacionamentos de exatidão e completeza das classes Mês, Ano e Estação. Finalmente, as hierarquias de classificação de Cidade-Uf-Região têm seus atributos agregados a classe Pessoa e Cliente.
O diagrama de Classes de Pedidos de Clientes da representação da UML para o modelo dimensional é, portanto, apresentado na figura 2.

4 MAPEAMENTO DO MODELO MULTIDIMENSIONAL PARA BANCO DE DADOS ORIENTADO A OBJETOS

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. São armazenados o modelo de dados, o metadados, regras e visualização do negócio modelado.
Muller (2002) observa que o mapeamento de um modelo de dados UML para um esquema orientado a objetos poderia ser direto, se os fornecedores de sistemas gerenciadores de banco de dados orientado a objetos aderissem a um padrão de definição de esquemas. Questões surgem da natureza inerente do projeto de objetos persistentes, aparecendo nos produtos SGBDOO e no padrão ODMG (*Object Data Management Group*). O autor observa ainda que o padrão ODMG abrange apenas a linguagem de consulta OQL e os recursos básicos que um SGBDOO deve oferecer, não oferecendo interoperabilidade. Porém, considerando-o como o atual padrão existente, as especificações dessa seção serão definidas observando os padrões estabelecidos pela ODMG.

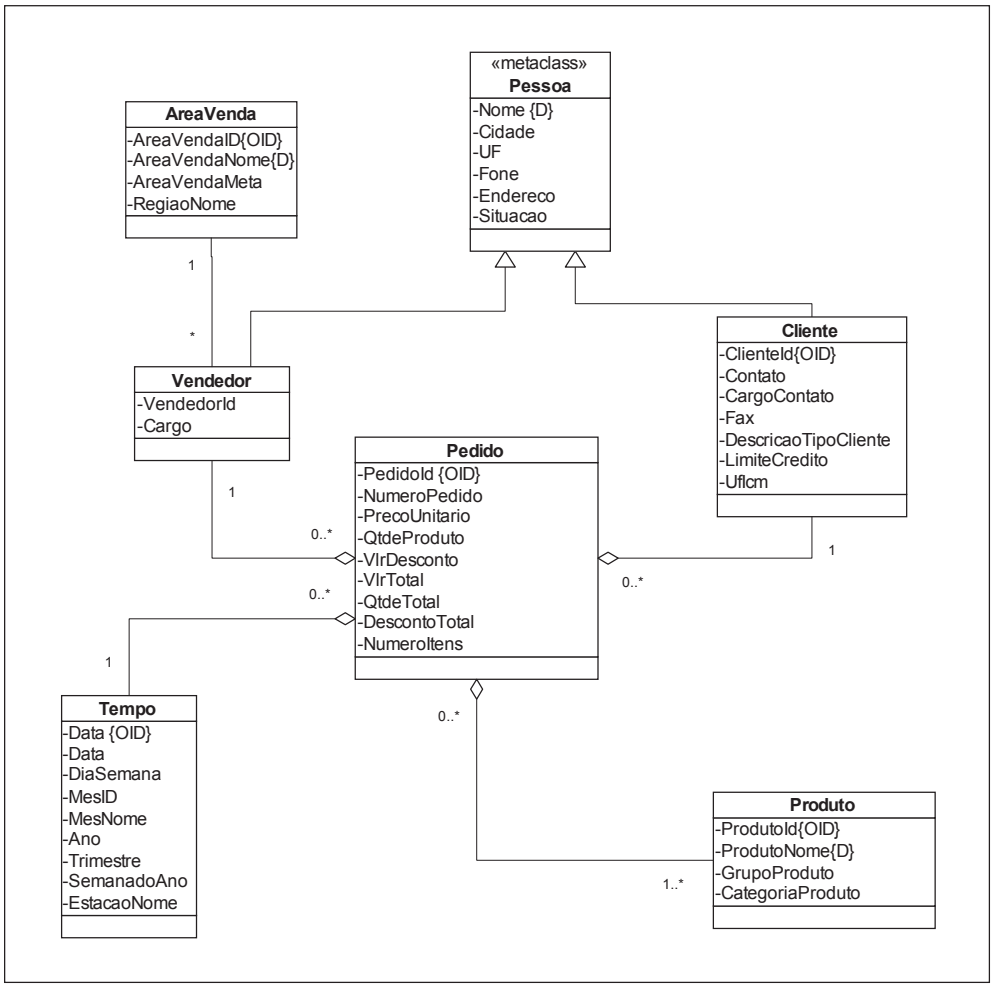

Figura 2. Utilização da UML para o modelo dimensional de Pedidos de Clientes.

A ODMG definiu uma linguagem de especificação usada para representar objetos em sistemas de banco de dados. Essa 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, 2004).

Os DBMSs usam a linguagem de definição de dados (DDL) e a linguagem de manipulação de dados (DML). 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 tipos objetos, definindo as características dos tipos, incluindo suas propriedades e operações (ODMG, 2004). Esse padrão é utilizado na seção de mapeamento do modelo dimensional proposto para o paradigma de banco de dados orientado a objetos.

Considerando a proposta do mapeamento do modelo de Pedidos de Clientes, os primeiros elementos a serem mapeados são as dimensões. Observando o trabalho dos autores que utilizaram a UML para a modelagem multidimensional (Trujillo e Palomar, 1998; Trujillo et al., 2001; Luján-Mora et al., 2004), e considerando o modelo proposto anteriormente, cada dimensão é mapeada para uma classe, em que cada elemento dimensional é mapeado para um atributo. São 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):

- classes dimensão não associativas: são as tabelas dimensão que não possuem informação adicional sobre os elementos dimensionais, como em um esquema *Star* simples;
- classes dimensão associativas: são aquelas tabelas dimensão que tem informação hierárquica ou informações adicionais sobre os elementos dimensionais, como em um esquema *Snowflake*.

4.1 Mapeando Classes em ODL

Observando essa classificação proposta por Buzydlowski et al. (1998), são mapeadas as dimensões definidas como não associativas, em que elementos dimensionais são considerados atributos simples. 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. As dimensões Cliente, Tempo e Produto são classificadas como não associativas e a dimensão Vendedor como uma associativa.

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 Vendedor e Cliente, com atributos gerais para as classes e específicos para cada uma delas separadamente. Portanto, os atributos generalizados são gerados na superclasse Pessoa.

A ODL versão 3.0 supõe que as classes são persistentes e a sintaxe que define essa persistência é específica de cada fabricante (Cattell, 2000). Portanto, para gerar a superclasse Pessoa através da ODL:

```
class Pessoa (extent Pessoas)
{
attribute string Nome;
...
}
```

A referência do objeto é analisada através do identificador de objeto, 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. A ODMG oferece um meio de especificar os valores das chaves por intermédio da cláusula key e um SGBDOO oferece algum tipo de indexação singular. O script abaixo em ODL mostra essa característica:

```
class Cliente extends Pessoa
          (extent clientes key ClienteId)
{ ...
attribute unsigned long ClienteId;
...
}
```

4.2 Mapeando Relacionamentos em ODL

A partir da superclasse definida através da geração da classe Pessoa, são geradas as classes especializadas de Vendedor e Cliente, com os atributos correspondentes. As classes herdam as características da classe Pessoa, e agregam 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, e em um banco de

dados orientado a objetos, 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 abaixo da classe Cliente:

```
class Cliente extends Pessoa
{
...
}
```

Através da declaração set pode-se indicar uma coleção não-ordenada de elementos sem duplicatas, usado para definir relacionamentos entre classes. No exemplo abaixo, o tipo de coleção literal set< > indica que cada cliente pode estar em zero ou mais pedidos, definido em ODL:

```
class Cliente extends Pessoa
{ ...
Relationship set<Pedido> PedidoCliente inverse
Cliente::pedidos;
...
}
```

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. No relacionamento entre as classes Vendedor e AreaVenda, os atributos representando a área de venda estão na classe Vendedor com a declaração no trecho ODL abaixo:

```
Class Vendedor extends Pessoa
 (extent vendedores key VendedorId)
{
attribute AreaVenda AreaVendedor;
...
}
```

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.
No modelo proposto, observa-se que cada pedido pode ser composto de vários produtos, sem a definição de um número exato de ocorrências para um mesmo pedido. Portanto, para a solução desse relacionamento, o modelo apresenta a proposta de Kimball (1998) e que possui variações na proposta de Giovinazzo (2000), incluindo-se uma tabela *bridge* entre as tabelas Fato e dimensão Produto. Rowen (2000) apresenta uma análise comparativa de propostas para os relacionamentos muitos-para-muitos entre tabelas fato e dimensão no modelo dimensional, incluindo a solução proposta por Kimball (1998).
Muller (2002) lembra que os SGBDOOs não suportam associações n-árias como relacionamentos de primeira classe, e que, portanto, o modelo de objetos ODMG não oferece uma maneira de declarar tais associações e que a solução desse problema consiste na criação de classes de associação. Considerando a solução proposta por Kimball (1998) definida acima, em ODL a geração da classe PedidoItem:

```
 Class PedidoItem (extent PedidoProduto)
{
relationship set<Pedidos> Itempedidos inverse
Pedidos::pedidos;
```

```
relationship set<Produtos> Itemprodutos inverse
Produtos::produtos;
attribute ...
}
```

Nas classes Pedido e Produto são gerados os relacionamentos correspondentes, em ODL:

```
Class Pedido (extent Pedidos)
{
relationship set<PedidoProduto> pedidos inverse
PedidoProduto::Itempedidos;
attribute ...
}

Class Produto (extent Produtos)
{
relationship set<PedidoProduto> produtos inverse
PedidoProduto::Itemprodutos;
attribute ...
}
```

4.3 Validação da proposta de mapeamento

A partir do modelo proposto, e da geração do mapeamento para banco de dados orientado a objetos utilizando o padrão ODMG, o modelo é implementado no banco de dados Caché.

4.3.1 Mapeando Classes no Caché

Os scripts do modelo exemplo proposto são gerados através do Caché Studio, que é a interface de desenvolvimento do banco de dados Caché, que permite a criação de classes, métodos, etc. Os exemplos são alocados no pacote denominado Modelo.

A classe dimensão Produto é gerada com os atributos como propriedades. Observando que a classe é definida como persistente, para armazenamento definitivo dos dados, utilizam-se conceitos de herança, herdando os métodos da classe Persistent. Os identificadores são gerados automaticamente pelo banco de dados e armazenado na estrutura e no metadados. Abaixo o script de geração da classe Produto no banco de dados orientado a objetos Caché:

```
Class modelo.Produto Extends %Persistent [ ClassType =
persistent, ProcedureBlock ]
{
Relationship ItemProduto As modelo.PedidoItem [ Cardinality
= many, Inverse = Produto ];
Property ProdutoID As %Integer [ Required ];
Index ProdutoIDIndex On ProdutoID [ Unique ];
...
}
```

4.3.2 Mapeando 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.

Os relacionamentos no Caché têm as seguintes características:

- relacionamentos são binários, em que o relacionamento é definido entre duas, e apenas duas classes, ou com uma classe e ela mesma;
- relacionamentos são definidos apenas para classes persistentes;
- relacionamentos devem ser bidirecionais, ou seja, ambos os lados do relacionamento devem ser definidos;
- relacionamentos administram automaticamente seu comportamento na memória e no disco;
- relacionamentos provêem escalamento superior e concorrência sobre coleções de objeto;
- relacionamentos são visíveis para o SQL como chaves estrangeiras.

Na versão 5.0, o Caché suporta dois tipos de relacionamento: um-para-muitos (independente) e pai-filho (dependente). 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.
Considerando a superclasse Pessoa uma generalização das classes Vendedor e Cliente, com atributos gerais para a superclasse e específicos para as classes especializadas, o script abaixo define tais características:

```
Class modelo.Pessoa Extends (%Persistent) [ ClassType =
persistent, ProcedureBlock ]
{
Property Nome As %String [ Required ];
...
}
```

A partir da superclasse definida através da geração da classe Pessoa, são geradas as classes especializadas de Vendedor e Cliente, com os atributos correspondentes. As classes herdam as características da classe Pessoa, e agregam suas próprias características, segundo o script abaixo:

```
Class modelo.Vendedor Extends modelo.Pessoa [ ClassType =
persistent, ProcedureBlock ]
{
Relationship PedidoVendedor As modelo.Pedido [ Cardinality
= many, Inverse = Vendedor ];
Property VendedorId As %Integer;
Property Cargo As %String;
Property AreaVenda As AreaVenda [ Required ];
Index VendedorIdIndex On VendedorId [ Unique ];
}
```

O tipo de classe serial implementa o relacionamento entre as classes Vendedor e AreaVenda, no qual os atributos representando a área de venda estão na classe Vendedor:

```
Class modelo.AreaVenda Extends %SerialObject [ ClassType =
serial, ProcedureBlock]
{
Property AreaVendaId As %Integer [ Required ];
...
}
```

Analisando a natureza da classe fatos, cada fato é uma classe associativa, de acordo com a granularidade dos dados, mapeados para classes. Como definido na seção anterior, o relacionamento entre a classe fatos Pedidos e a classe dimensão Produto caracteriza um relacionamento muitos-para-muitos, gerando a classe *bridge* de associação PedidoItem. Portanto, a classe Fatos contém os valores sumarizados do Pedido e a classe de associação contém os valores de cada produto, conforme o script gerado abaixo para as classes Pedido, Produto e PedidoItem, respectivamente:

```
Class modelo.Pedido Extends %Persistent [ ClassType =
persistent, ProcedureBlock ]
{
Relationship Item As modelo.PedidoItem [ Cardinality =
children, Inverse = Pedido ];
Relationship Cliente As modelo.Cliente [ Cardinality = one,
Inverse = PedidoCliente ];
Index ClienteIndex On Cliente;
Index NumeroPedidoIndex On NumeroPedido [ Unique ];
Property ...
}

Class modelo.PedidoItem Extends %Persistent [ ClassType =
persistent, ProcedureBlock ]
{
Relationship Pedido As modelo.Pedido [ Cardinality =
parent, Inverse = Item ];
Relationship Produto As modelo.Produto [ Cardinality = one,
Inverse = ItemProduto ];
Index ProdutoIndex On Produto;
...}
```

4.3.3 Mapeando Objetos

As classes do modelo são povoadas através do instanciamento de objetos. Através do 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.

Para criar um novo objeto da classe Produto e instanciá-lo, através do Caché Terminal:

```
Set objPro = ##class(modelo.Produto).%New()
```

Após a atribuição dos valores que serão armazenados no objeto, para torná-lo persistente:

```
Set ok = objPro.%Save()
```

5 CONCLUSÕES

Observa-se que a modelagem dimensional apresenta-se atualmente como um importante modelo para representar as regras de negócio das empresas. A utilização da notação UML, através do diagrama de classes para representar tais procedimentos auxilia o entendimento do desenvolver, facilitando a compreensão do modelo real e sua posterior implementação.

Porém, observa-se que quando esse modelo é mapeado para um modelo de banco de dados relacional, as características da orientação a objetos são desvinculadas e a geração dos dados difere do modelo proposto. Portanto, a contribuição do trabalho centra-se na

proposta de mapear o modelo dimensional modelado através da UML para um banco de dados orientado a objetos, fazendo sua validação através da implementação de um modelo, utilizando o banco de dados pós-relacional Caché, para manter a persistência dos objetos modelados em suas respectivas classes.

A tecnologia de banco de dados orientado a objetos é relativamente nova e são poucos os produtos comerciais que implementam efetivamente as características da orientação a objetos. Utilizando-se um modelo de SGBDOO, as características e conceitos da orientação são preservados e aplicados segundo sua definição e modelagem.

Portanto, a aplicação da UML, especificamente através do diagrama de classes para representar o modelo dimensional, torna-se mais eficiente e representativa através da implementação do modelo na tecnologia orientada a objetos para o armazenamento dos dados, preservando as características e conceitos definidos no modelo lógico.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ABELLÓ, A.; SANTOS, J.; SALTOR, F. 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, 2000.

AMBLER, S. W. UML 2 Class Diagrams. 2005. Disponível em: <http://www.agilemodeling.com/artifacts>.

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.

CATTELL, R. G. G.; BARRY, D. K. The Object Data Standard: ODMG 3.0. San Francisco: Morgan Kaufmann Publishers, 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), 1998.

HÜSEMANN, B.; LECHTENBÓRGER, J.; GOTTFRIED, V. Conceptual Data Warehouse Design. 2nd Int. Workshop on Design and Management of DW (DMDW). Stockholm, Sweden, 2000.

INMON. W. H. Data Warehousing – como transformar informações em oportunidades de negócios. São Paulo: Berkeley, 2001.

KIMBALL, R. A dimensional modeling manifesto. 1997. Disponível em: <http://www.rkimball.com/html/articlesArchitecture.html>.

___. 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.

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, Porto (Portugal), 2004.

MACHADO, F. N. R. Projeto de data warehouse: uma visão multidimensional. São Paulo: Érica, 2000.

MONTEIRO NETO, R. R. Mapeamento entre os modelos ER e Star. 1998. Disponível

em: http://genesis.nce.ufrj.br/dataware/.
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. 2004. Disponível em <http://www.odmg.org>.
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. 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, 1998.
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. et al. Designing Data warehouse with OO conceptual models. IEEE – Institute of Electrical and Electronics Engineer. Vol 34, nº 12, p. 66-75, 2001.