JOÃO ALEXANDRE BONIN DE MELLO

UMA PROPOSTA DE MODELO DE DADOS PARA SUPORTE AO PROCESSAMENTO TRANSACIONAL E DE APOIO INFORMACIONAL SIMULTANEAMENTE

Dissertação apresentada ao Programa de Pós-graduação em Engenharia de Produção da Universidade Federal de Santa Catarina como requisito parcial para obtenção do grau de Mestre em Engenharia de Produção

Orientador: Prof. Aran Bey Tcholakian Morales, Dr.

FLORIANÓPOLIS

2002

JOÃO ALEXANDRE BONIN DE MELLO

UMA PROPOSTA DE MODELO DE DADOS PARA SUPORTE AO PROCESSAMENTO TRANSACIONAL E DE DATA WAREHOUSE SIMULTANEAMENTE

Esta dissertação foi julgada aprovada para obtenção de grau de **Mestre em Engenharia de Produção** no **Programa de Pós-graduação em Engenharia de Produção** da Universidade Federal de Santa Catarina

Florianópolis, 23 de março de 2002.

_____

Prof. Ricardo Miranda Barcia, Ph.D.

Coordenador do Programa

Banca Examinadora

Prof. Aran Bey Tcholakian Morales, Dr. Universidade Federal de Santa Catarina **Orientador**	Prof. Denilson Sell, Ms. Universidade Federal de Santa Catarina
Prof. José Leomar Todesco, Dr Universidade Federal de Santa Catarina	Prof. Oscar Ciro Lopez Vaca, Dr Universidade Federal de Santa Catarina

A minha esposa Lourdes.

Aos meus filhos Victor e Murilo.

Agradecimentos

À Universidade Federal de Santa Catarina.

Ao departamento de Engenharia de Produção e Sistemas

Ao prof. Aran Bey Tcholakian Morales,

pelo competente trabalho de orientação.

À Universidade Paranaense pelo apoio financeiro.

Ao prof. José Leomar Todesco, prof. da disciplina de Data Warehouse, por ter despertado o interesse pelo tema.

A Denilson Sell, pelas valiosas contribuições.

Aos professores Sérgio Rodrigues e Divair Maria Terna Gomes.

Aos professores do programa de pós-graduação em Engenharia de Produção e Sistemas.

A todos que contribuíram, direta ou indiretamente, para a realização desta pesquisa.

Resumo

MELLO, João Alexandre Bonin de. **Uma proposta de modelo de dados para suporte ao processamento transacional e de data warehouse simultaneamente**. 2001, 101f. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-graduação em engenharia de produção, UFSC, Florianópolis.

Este trabalho propõe um modelo de dados que permita o suporte ao processamento transacional e informacional simultaneamente. Este modelo é voltado à organizações que não dispõe de recursos para investimento em ambientes separados para aplicações de processamento transacional e para aplicações informacionais. No trabalho são descritos conceitos de sistemas de informação, sua classificação de acordo com a finalidade, o modelo entidade/relacionamento, o modelo dimensional e o data warehouse. A fim de se verificar a viabilidade do modelo proposto são construídos protótipos de sistemas baseados no modelo proposto, no modelo relacional e no modelo dimensional. Estes protótipos são comparados entre si de forma qualitativa e quantitativa. A análise dos resultados dos testes demonstrou a viabilidade do modelo na situação testada, mostrando que o modelo proposto é capaz de suportar tanto o processamento transacional quanto informacional e gerar uma redução de custos pela unificação dos dois ambientes de dados (transacional e informacional) e pela eliminação da camada de data staging, necessária à utilização de data warehouses para o processamento informacional.

Palavras-chave: Data Warehouse; Modelagem de Dados; Sistemas de Apoio à Decisão.

Abstract

MELLO, João Alexandre Bonin de. **Uma proposta de modelo de dados para suporte ao processamento transacional e de informacional simultaneamente**. 2001, 101f. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-graduação em engenharia de produção, UFSC, Florianópolis.

This paperwork has the purpose provide a data model that could support the transactional and informational process at the same time. This related to some organizations that don't have economical resources to invest ins separated environments to transactional and informational applications. In the paperwork are described concepts of information systems, it classification is according with this purpose, the entity relationship model, the dimensional model and data warehouse. In order to verify the viability of it was built some prototypes based on the model purposed, on the relational model and on the dimensional model. These prototypes are compared each other, qualitative and quantitative. The results analyzed from the tests showed the viability in that situation, proving that the sample is able to support as the transactional and informational processing and that it generates an expenses reduction through the unifying the both data environment (transactional and informational) and through the data staging elimination, necessary to the usage of data warehouse in the informational processing.

Keywords: Data Warehouse; Data Modeling; Decision Suport Systems.

Sumário

Lista de Figuras

Lista de Quadros

Lista de reduções

DBMS	*Database Management System.* Sistema de gerenciamento de bancos de dados.
EIS	*Executive Information System.* Sistema de Informação executiva
OLTP	*On-line transaction processing.* Sistemas de processamento de transações.
SAD	Sistema de apoio à decisão
SGBD	Sistema de gerenciamento de banco de dados
SIG	Sistema de Informação gerencial
SPT	Sistema de processamento transacional

1. INTRODUÇÃO

1.1 Apresentação

O modelo entidade/relacionamento trouxe agilidade, exatidão e economia de espaço de armazenamento de dados para os sistemas de processamento transacional. Esta técnica, entretanto, revelou-se ineficiente para a implementação de sistemas informacionais, tais como de apoio à decisão e sistemas de informação executiva que necessitem de grandes bases de dados. Isso ocorre devido ao grande número de junções entre tabelas necessárias à construção de consultas mais complexas, o que degrada a performance do sistema, e à complexidade do modelo para usuários finais (geralmente tomadores de decisão) que constroem suas próprias consultas utilizando-se de ferramentas apropriadas. Esta deficiência foi suprida com a utilização de *data warehouses* baseados no modelo dimensional. Este modelo viabilizou a utilização o processo de descoberta do conhecimento em grandes bases de dados. O modelo dimensional proporciona uma forma simples e eficiente de armazenamento de grandes volumes de dados para alimentação de sistemas de apoio à decisão. Este modelo, entretanto, não se propõe a automatizar o processamento transacional das organizações.

A utilização do modelo entidade/relacionamento para os sistemas transacionais e do modelo dimensional para o data warehouse, destinado a suportar os sistemas informacionais, implica em duplicar a infra-estrutura de processamento de sistemas de informação, uma vez que são necessários dois ambientes de armazenamento de dados e estes precisam ser migrados periodicamente do ambiente transacional para o data warehouse, através de processos de extração, transformação e limpeza de dados. Este processo envolve custos, tornando proibitiva a sua utilização por aquela porção das organizações que não dispõe de recursos suficientes para realizar tais investimentos.

O presente trabalho propõe a utilização de um modelo de dados que possa unir as características do modelo entidade/relacionamento às características do modelo dimensional, possibilitando que as empresas retirem suas informações estratégicas diretamente do banco de dados operacional.

1.2. Objetivos

1.2.1. Objetivo Geral

Investigar a viabilidade da modelagem de dados utilizando um modelo que possa unir as características do modelo dimensional às do modelo entidade/relacionamento, criando um modelo que suporte o processamento transacional e informacional.

1.2.2. Objetivos Específicos

- Estudar detalhadamente as técnicas de modelagem de dados baseadas nos modelos entidade/relacionamento e dimensional, enumerando as vantagens, desvantagens e aplicabilidade da cada um.
- Propor um modelo de dados que possibilite extrair informações estratégicas, de modo eficiente, diretamente do banco de dados transacional.
- Validar a aplicabilidade deste modelo através do desenvolvimento de protótipos.

1.3 Justificativa

Em um ambiente globalizado, cada vez mais as empresas necessitam de informações para reagir aos problemas e oportunidades de negócio que surgem no horizonte (LAUDON e LAUDON, 1999), independentemente da área de negócio ou do porte das mesmas. Para HARRISON (1998, p 3) “aumentar o capital intelectual das empresas é uma necessidade competitiva (...). As organizações que usam com eficácia a tecnologia de informações adquirem conhecimento e velocidade para alcançar uma esmagadora superioridade nos mercados em que atuam”.

Os custos gerados pela duplicação dos ambientes de dados, um para o processamento transacional e outro para a extração de informações gerenciais, impossibilita uma grande quantidade de pequenas e médias empresas de se beneficiar de sistemas de apoio informacional, suportados por data warehouses. O modelo de dados proposto neste trabalho oferece uma tecnologia de baixo custo

para a criação de modelos de dados para as organizações de pequeno porte, que não dispõem de recursos para investir em dois ambientes de dados, um para os sistemas de apoio à decisão e de informações gerenciais e outro para processamento de transações.

O modelo proposto pressupõe a utilização de um único ambiente de dados, unindo características do modelo entidade-relacionamento ao modelo relacional, possibilitando o processamento de transações e a extração de informações de apoio à decisão de uma única base de dados, capaz de suportar aplicações de organizações de pequeno porte, que não necessitam de grandes volumes de informação e não dispõe de recursos para a duplicação de ambientes.

1.4 Estrutura do Trabalho

A dissertação está dividida em cinco partes. A primeira trata do referencial teórico necessário para a proposta do modelo híbrido de dados. Nesta parte é estudada a classificação dos sistemas de informação segundo sua finalidade, o modelo entidade/relacionamento, o modelo dimensional como técnicas para modelagem de sistemas transacionais e de apoio informacional. A segunda expõe a metodologia adotada para a execução do trabalho. Na terceira parte é proposto um modelo de dados que possibilite extrair informações estratégicas diretamente da base de dados transacional. A quarta parte apresenta os protótipos desenvolvidos e a análise dos resultados dos testes efetuados. Finalmente são tecidas as conclusões a respeito do trabalho e recomendados estudos mais aprofundados.

2. REVISÃO DA LITERATURA

2.1 Introdução

Este capítulo traça um perfil dos sistemas de informação, classificando-os de acordo com sua finalidade. Uma ênfase especial é dada aos sistemas de processamento de transação e ao modelo entidade/relacionamento, destinado a suportá-lo, ao ambiente de data warehouse e ao modelo dimensional.

Dado o objetivo principal deste trabalho, que é propor um modelo de dados capaz de suportar aplicações de processamento transacional e informacional, as técnicas de modelagem de dados, tanto para o modelo entidade/relacionamento quanto para o modelo dimensional são discutidas em detalhe, a fim de subsidiar o modelo proposto.

2.2 Sistemas de Informação

Um sistema de informações é composto de um conjunto interdependente de pessoas, estruturas organizacionais, software, hardware, processos e métodos interligados com o objetivo de facilitar o planejamento e o controle em empresas e outras organizações, organizando informações de forma que estas se tornem utilizáveis na coordenação do fluxo de trabalho de uma empresa. (LAUDON e LAUDON, 1998). Um sistema de informações tem a função de coletar, manipular, armazenar e disseminar dados e informações pela empresa (STAIR, 1998).

Os sistemas de informação transformam a informação em uma forma utilizável, contendo informação a respeito do ambiente interno e externo da organização, ajudando os membros da mesma a tomar decisões, possibilitando múltiplas análises e visualizações de informações a respeito de situações complexas (LAUDON e LAUDON, 1998).

2.2.1 Classificação dos Sistemas de Informação

Os sistemas de informação podem ser classificados de acordo com sua finalidade. Estes podem ser classificados em sistemas de processamento transacional, sistemas de informações gerenciais, sistemas de apoio à decisão, sistemas especialistas e sistemas de informação executiva (FREITAS e

POZZEBOM, 2000 e FURLAN, 1994, STAIR, 1998, FALSARELLA e CHAVES, 2001).

BOAR (2001), classifica os sistemas de informação em dois grandes grupos: os aplicativos do negócio e os aplicativos que analisam informações a respeito do negócio. O primeiro grupo corresponde àquelas aplicações que processam as transações diárias da organização, os sistemas de processamento transacional. Para este tipo de aplicação, integridade e performance a nível transacional e disponibilidade são requisitos imperativos. O segundo grupo são as aplicações que analisam informações a respeito do negócio, os sistemas informacionais. Estas aplicações se caracterizam por bases de dados estáticas ou de modificações lentas, armazenamento de dados por um longo período de tempo e facilidade e performance no processo de extração de informações. Os sistemas de processamento transacional estão enquadrados no primeiro grupo de aplicações, enquanto que os sistemas de informação gerencial, sistemas de suporte à decisão e sistemas de informação executiva se encaixam no segundo grupo.

Neste trabalho serão apresentadas as definições para sistemas de processamento transacional, sistemas de informação gerencial, sistemas de apoio à decisão e sistemas de informação executiva.

2.2.2 Sistemas de Processamento Transacional

Os sistemas de processamento transacional (SPT), atualmente chamados de OLTP (On-line transaction processing) (KIMBALL, 1998a), foram os primeiros aplicativos de computador a serem desenvolvidos na maioria das organizações. Eles têm a tarefa de monitorar e processar as funções básicas e rotineiras de uma organização, tais como processamento da folha de pagamento, faturamento etc. (STAIR, 1998, LAUDON e LAUDON, 1998).

Estes sistemas possuem algumas características próprias, tais como computação simples, grandes volumes de dados de entrada e saída, alto grau de repetição (LAUDON e LAUDON, 1998, STAIR, 1998), disponibilidade necessária em 100% do tempo de funcionamento da organização, altas taxas de performance no processamento de transações e grande complexidade no esquema conceitual da

base de dados, devido ao grande número de entidades e relacionamentos (BOAR, 2001).

Como os sistemas de processamento transacional operam as tarefas básicas e rotineiras da organização e o relacionamento com fornecedores, clientes e governo, os fatores críticos de sucesso para este tipo de sistema são: alto grau de precisão, integridade a nível transacional e produção de documentos em tempo hábil (STAIR, 1998, KIMBAL, 1998).

Sistemas de processamento transacional podem ser utilizados para a obtenção de vantagem competitiva, isto pode ser obtido utilizando-se este tipo de sistema para oferecer serviços e produtos de qualidade superior aos clientes, melhor agrupamento de informações e aperfeiçoamento do processo de planejamento (STAIR, 1998).

2.2.3 Sistemas de Informação Gerencial

Um sistema de informações gerenciais (SIG) tem o propósito de fornecer informações necessárias à medição da eficiência operacional da organização, dando ênfase às necessidades gerenciais através de informações resumidas, obtidas a partir da filtragem e análise de dados altamente detalhados, extraídos das bases de dados dos sistemas de processamento transacionais e de fontes externas (STAIR, 1998, FALSARELLA e CHAVES, 2001).

2.2.4 Sistemas de Apoio à Decisão

Um sistema de apoio à decisão (SAD) tem por finalidade dar embasamento à tomada de decisões referentes a problemas específicos. Este tipo de sistema é geralmente utilizado quando a tomada de decisão se torna difícil sem o apoio de um sistema que forneça informações específicas que possam dar o devido suporte à tomada de decisão. Um SAD deve reconhecer que existem estilos gerenciais diferentes e tipos de decisões diferentes. Isso implica em uma grande participação do usuário do SAD no seu desenvolvimento (STAIR, 1998; SPRAGHE e WATSON, 1991).

De acordo com STAIR (1998) um SAD deve ser capaz de manipular grandes volumes de dados, obter e processar informações de diferentes fontes, proporcionar flexibilidade de relatórios e apresentação, possuir orientação tanto textual quanto gráfica e executar análises e comparações complexas e sofisticadas.

Adicionalmente, um SAD pode também tem a capacidade de encontrar soluções para problemas menores através de abordagens de otimização e heurísticas e ainda executar análises de atendimento de metas (STAIR, 1988).

2.2.5 Sistemas de Informação Executiva

Os sistemas de informação executiva (*executive information systems* – EIS) oferecem um meio prático para que executivos acessem informações estratégicas da empresa de um modo agrupado e simplificado de acordo com suas necessidades, ajustadas ao seu estilo de trabalho (FURLAN, IVO e AMARAL, 1994).

Um EIS é um tipo especial de SAD destinado à tomada de decisões de alto nível da organização, oferecendo informações estruturadas, tanto internas quanto externas a respeito de aspectos da organização considerados fatores críticos de sucesso para a mesma (FALSARELLA e CHAVES, 2001, STAIR, 1988). Enquanto um SAD oferece apoio para decisões focadas em uma área específica, geralmente oferecendo suporte à média e baixa gerência, um EIS oferece informações consolidadas de alto nível e análises multidimensionais a executivos de alto nível (GUPTA, 2001). FALSARELLA e CHAVES (2001) acrescentam ainda que um EIS deve fornecer também informações detalhadas a respeito do passado, presente e perspectivas futuras a fim de auxiliar o processo de planejamento e controle da organização, permitindo aos seus usuários navegar por diversos níveis de detalhamento da informação.

O sucesso de um EIS depende não só da disponibilidade das informações necessárias ao processo decisório, como também a forma de extração e exploração destas informações utilizadas no momento da tomada de decisão, uma vez que diferentes tomadores de decisão necessitam de diferentes informações e possuem estilos decisórios diferentes (FREITAS e POZZEBOM, 2000).

STAIR (1998) resume algumas características importantes para um Sistema de Informações Executivas. Estas características são: facilidade de uso, manipulação de dados externos e internos, quantitativos e qualitativos, execução de sofisticadas análises de dados, alto grau de especialização, flexibilidade e suporte a todos os aspectos da tomada de decisões (uma vez que decisões que utilizam fontes externas de informações podem ter um alto grau de incerteza). Um EIS também deve oferecer recursos abrangentes de comunicação, uma vez que executivos necessitam de comunicação instantânea com o ambiente interno e externo da organização. FREITAS e POZZEBOM (2000) defendem uma mudança do foco dos sistemas de informações executivas para sistemas de informação empresarial, onde os tomadores de decisão de todos os níveis possam se beneficiar da oferta de informações. VOLOLINO, WATSON e ROBINSON *apud* FREITAS e POZZEBON (2000) definem EIS como uma tecnologia de informação para todos os usuários finais da organização, sendo os executivos um subconjunto destes usuários, onde o suporte é ajustado de acordo com as necessidades de cada classe de usuários.

2.3 O Modelo Entidade/Relacionamento e os Sistemas Transacionais

2.3.1 Introdução

Sistemas OLTP processam uma grande quantidade de transações diariamente. Consistência transacional e velocidade são requisitos imperativos neste tipo de sistema. Eles operam as atividades essenciais de uma organização (KIMBALL, 1998a).

O modelo entidade/relacionamento se aplica perfeitamente a este tipo de sistema, pois divide os dados em várias entidades distintas, cada uma implementada em uma tabela no banco de dados (KIMBALL, 1998a). Neste tipo de modelo, as informações são vistas sob a perspectiva de transações atômicas (SINGH, 2001). Técnicas de normalização eliminam completamente a redundância de dados, reduzindo significativamente a probabilidade de inconsistência e aumentando a velocidade de processamento transacional, uma vez que as atualizações do banco de dados a cada transação se resumem a uma pequena quantidade de registros por vez (KIMBALL, 1998a).

Este modelo foi proposto primeiramente por Peter Chen em 1976 e a partir daí foi aprimorado por ele mesmo e por outros. Este modelo foi adotado por várias ferramentas CASE, que também o modificaram. Atualmente não existe um padrão unificado para este modelo. O que existe é um conjunto de conceitos universalmente aceitos, dos quais se originam a maioria das variações de modelo entidade/relacionamento (KROENKE, 1999).

2.3.2 Elementos do modelo entidade/relacionamento

Os principais elementos do modelo entidade/relacionamento são entidades, atributos, identificadores e relacionamentos (KROENKE, 1999):

a) Entidades

Entidade é algo que pode ser identificado no ambiente de trabalho do usuário, algo que o mesmo queira localizar (KROENKE, 1999). Entidades representam classes de objetos do mundo real que podem ser observadas e classificadas segundo suas propriedades (BALLARD et al. 2001). Entidades podem ser definidas como “qualquer objeto distinguível em um banco de dados” (DATE, 2000). Entidades de um mesmo tipo são agrupadas em classes de entidade. Uma ocorrência de uma entidade é chamada de instância de entidade. Assim, o conjunto formado por todos os funcionários de uma organização forma a classe de entidades FUNCIONÁRIO (KROENKE, 1999).

b) Atributos

Atributos descrevem características das entidades. Estes atributos também são chamados de propriedades. O nome do funcionário é um atributo da entidade FUNCIONÁRIO. De acordo com a definição do modelo entidade/relacionamento, todas as instâncias de uma mesma classe de entidade possuem os mesmos atributos (KROENKE, 1999).

c) Identificadores

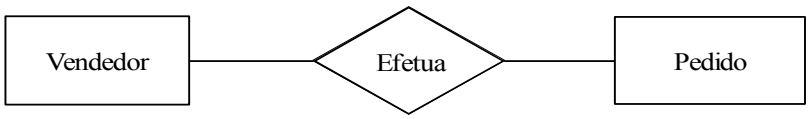
Identificadores são atributos que identificam uma determinada instância de entidade. Identificadores podem ser únicos ou não únicos, se forem únicos

identificarão uma, e somente uma instância de entidade. Se for não único identificará um conjunto de instâncias de entidade (KROENKE, 1999).

d) Relacionamentos

Relacionamentos descrevem a interação estrutural e a associação entre as entidades de um modelo (BALLARD et al. 2001). O modelo entidade/relacionamento permite associações ente instâncias de entidade (relacionamentos entre instâncias) e associações entre classes de entidade (relacionamentos entre classes). Relacionamentos podem possuir atributos próprios. Um atributo de relacionamento é um tipo de atributo que só existe se houver um relacionamento entre entidades (KROENKE, 1999). A Figura 1 ilustra um relacionamento entre as entidades VENDEDOR e PEDIDO.

Figura 1 - Relacionamento entre entidades



Fonte: Adaptado de KROENKE, D. **Banco de Dados:** Fundamentos, projeto e implementação. Rio de Janeiro: Livros Técnicos e Científicos, 1999.

2.3.3 Conceitos Adicionais à Respeito do Modelo Entidade/Relacionamento

Alguns conceitos adicionais são importantes para a compreensão do modelo entidade/relacionamento: entidades fracas, generalização e especialização. Estes conceitos serão discutidos a seguir.

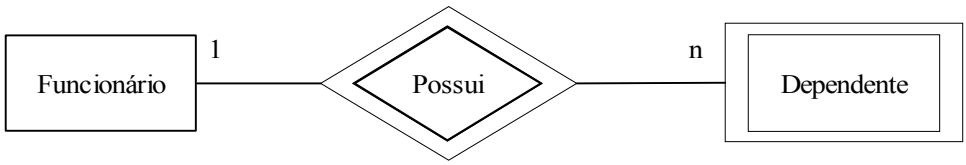
O modelo entidade/relacionamento define um tipo especial de entidade chamado de entidade fraca (Figura 2). A caracterização de uma entidade fraca se dá através da análise de duas características (COUGO, 1997, KROENKE, 1999):

a) dependência de existência;

b) dependência de identificador

A dependência de existência se dá quando a existência de uma entidade A só é possível se existir uma entidade B. Dizemos então que B é uma entidade forte e A é uma entidade fraca, enquanto que a dependência de identificador se dá quando o atributo identificador da entidade fraca inclui o atributo identificador da entidade forte à qual o mesmo está relacionado (COUGO, 1997, KROENKE, 1999). Um relacionamento entre uma entidade forte e uma fraca é sempre um relacionamento de um para muitos (DATE, 2000). Estes critérios podem ser dispensados do ponto de vista conceitual, sendo importantes no projeto lógico, onde os atributos identificadores passam a ter papel importante no endereçamento de uma instância de entidade (COUGO, 1997).

Figura 2 - Entidade forte e entidade fraca



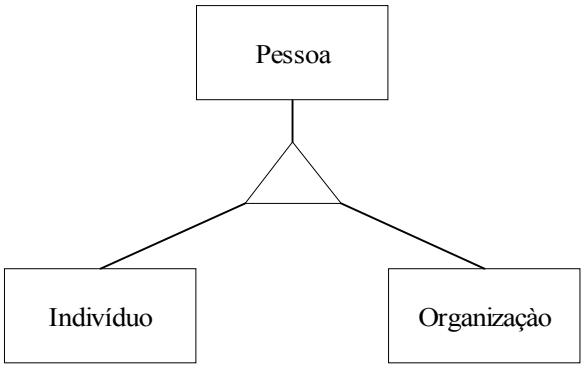
Fonte: Adaptado de KROENKE, D. Banco de Dados: Fundamentos, Projeto e Implementação. 6ª ed. Rio de Janeiro: Livros Técnicos e Científicos, 1999.

Quando são encontradas entidades que possuem um conjunto de atributos comum para descrevê-las, podemos generalizá-las em uma única entidade. A este processo chama-se generalização. De outro modo, quando encontramos um grupo de atributos que não são comuns ao grupo de entidades, podemos criar diversas entidades especializadas para conter estes atributos. Este processo é chamado de especialização (MACHADO, 1996). A uma entidade generalizada, dá-se o nome de supertipo de entidade, enquanto que a uma especialização de entidade dá se o nome de subtipo de entidade (DATE, 2000, KROENKE, 1999). A Figura 3 ilustra uma estrutura de generalização e especialização.

A derivação de uma estrutura conceitual de generalização e especialização não nos levará diretamente ao modelo lógico. Caso todos os atributos sejam comuns, pode-se optar por implementar todas as entidades em uma única entidade generalizada, acrescentando-se um atributo identificador. No extremo oposto, podem existir atributos que sejam específicos de cada uma das entidades especializadas. Na Figura 3, apenas a entidade PESSOA, possui os atributos Nome e Endereço, entretanto apenas a entidade INDIVÍDUO possui os atributos Documento de Identidade e CPF e apenas a entidade ORGANIZAÇÃO possui o atributo CNPJ,

neste caso, para evitar atributos com valores nulos, é aconselhável implementar cada entidade em uma tabela lógica separada. Existem casos em que se pode adotar uma solução combinada, adotando-se as duas soluções ao mesmo tempo (COUGO, 1997).

Figura 3 - Generalização e Especialização



2.3.4 Considerações Finais à Respeito do Modelo Entidade/Relacionamento

O modelo entidade/relacionamento é destinado ao processamento de transações. Técnicas de normalização eliminam completamente a redundância de dados (KROENKE, 1999, DATE, 2000), reduzindo significativamente a probabilidade de inconsistência e aumentando a velocidade de processamento transacional, uma vez que as atualizações do banco de dados a cada transação se resumem a uma pequena quantidade de registros por vez (KIMBALL, 1998a).

O modelo relacional tem sido utilizado como ferramenta de modelagem de dados, por fornecer uma visão dos requisitos de negócio e por servir de base para a implementação da estrutura de dados resultante (BALLARD et al. 2001).

Apesar do modelo entidade/relacionamento se adaptar perfeitamente ao processamento de transações, este modelo mostrou-se ineficiente para o processamento de sistemas de apoio à decisão (KIMBALL, 1998a).

Um diagrama entidade/relacionamento é bastante simétrico, não sendo possível a percepção de quais entidades são mais importantes e quais possuem atributos

numéricos que registram as medições de fatos. Freqüentemente, os diagramas possuem muitas entidades, tornando seu entendimento bastante difícil, tanto por parte de profissionais de desenvolvimento de aplicativos quanto por usuários finais. Esta dificuldade no entendimento do modelo informacional que o diagrama representa. Isso acaba por dificultar a construção de consultas personalizadas por parte dos usuários finais (KIMBALL, 1998a).

Modelos entidade/relacionamento, apesar de serem eficientes para processamento transacional são desastrosos como base para aplicações informacionais, porque são difíceis de serem entendidos pelo usuário final nem podem ser navegados de forma útil pelo gerenciador de bancos de dados (KIMBALL, 1998a).

2.4 O Modelo Dimensional e o Data Warehouse

2.4.1 Motivação

A origem do conceito de data warehouse remonta ao início dos anos 80, quando os bancos de dados relacionais tornaram-se produtos comerciais. As facilidades de extração de dados proporcionadas pela linguagem SQL proporcionaram um aumento do interesse por ferramentas de extração de dados direcionadas ao usuário final. Desde aquele momento, houve uma preocupação com o impacto que estas consultas causavam na performance do banco de dados dos sistemas OLTP, o que forçou a utilização de bases de dados separadas, contendo instantâneos utilizados exclusivamente para o processamento de consultas e para os sistemas de suporte à decisão (BALLARD et al. 2001).

Desde aquela época ficaram claras as diferenças entre as necessidades dos sistemas OLTP e dos SAD e EIS. Enquanto os sistemas OLTP necessitam de performance e exatidão a nível transacional, os SAD e EIS necessitam de performance na extração de dados e uma grande flexibilidade nos tipos de consulta efetuados (GUPTA, 2001).

Por outro lado, o aumento da capacidade dos sistemas OLTP de coletar, processar e armazenar dados ultrapassou em muito nossa capacidade de extrair conhecimento dos mesmos (SINGH, 2001), enquanto isso, o processo de

reestruturação das organizações e a rapidez nas mudanças ambientais causada pela globalização da economia, aumentaram a necessidade de respostas para uma série de questões que afetam diretamente a forma como as pessoas conduzem seus negócios (GUPTA, 2001).

Estes fatores, aliados ao aumento de poder de processamento dos computadores pessoais e dos sistemas operacionais para servidores, tais como o Windows NT e as várias versões de Unix, a evolução dos SGBDs e das ferramentas de consulta, geraram um ambiente propício para a evolução da soluções baseadas em data warehouse (GUPTA, 2001).

2.4.1 Conceito

Um data warehouse é o ponto central da arquitetura de processamento de informações estratégicas para tomada de decisões (KIMBALL, 1998a), fornecendo o suporte informacional para os sistemas de apoio à decisão (SAD) e para os sistemas de informação executiva (EIS). O data warehouse é construído separadamente da base de dados dos sistemas OLTP sob uma perspectiva de armazenamento de informações de longo prazo. Sua base de dados é orientada por assunto, consolidada, integrada, variável com o tempo e não volátil (INMON, 1997), fornecendo informações precisas, completas e apoiando as necessidades de informação do usuário a fim de lidar com aspectos estratégicos de longo prazo (HARRISON, 1998), constituindo uma tecnologia de gestão e análise de dados (SINGH, 2001).

O data warehouse deve ser capaz de gerar rapidamente pequenos conjuntos de resposta baseados em uma infinidade de registros. Usuários de um data warehouse alteram constantemente o tipo de consulta efetuada, dependendo de suas necessidades. Estas consultas podem utilizar-se de centenas ou milhões de registros, causando impacto variável na performance do banco de dados (KIMBALL, 1998a).

Para BOAR (2001), o data warehouse oferece uma visão estratégica do negócio, orientada à dimensão temporal, possibilitando o aprendizado com o passado, permitindo analisar o presente de forma a realizar adaptações em tempo real e

possibilitando traçar o futuro através do conhecimento antecipado das condições ambientais.

2.4.2. Características de um Data Warehouse

A definição de INMON (1997), ressalta as características principais de um data warehouse. MACHADO (2000), DOMENICO (2001) e SELL (2001), discutem estas características detalhadamente:

a) Orientação por Assunto

Orientação por assunto significa que os data warehouses agrupam as informações por área de interesse da organização, em contrapartida aos sistemas de processamento transacionais, que são orientados por processos.

b) Variação no Tempo

Os data warehouses representam os resultados operacionais em um determinado momento de tempo, isto significa que os dados de um data warehouse não podem sofrer alteração.

c) Não Volatilidade

Um data warehouse possui duas operações básicas: carga de dados e acesso aos mesmos. Estes dados se originam dos sistemas transacionais, são transformados e purificados e em seguida carregados no data warehouse. Estes dados permanecem lá até que se decida que eles não serão mais necessários.

d) Integração

Os dados de um data warehouse possuem um alto nível de integração, isto significa que inconsistências devem ser eliminadas e que as convenções de nomes de atributo, tais como sexo, estado civil, assim como os tipos de dados são formalmente unificados, definindo assim uma apresentação única para os dados do data warehouse, eliminando-se as possibilidades de respostas ambíguas.

2.4.3 Data Mart

Para INMON (1997), um data mart pode ser definido como um SGBD multidimensional que fornece uma estrutura bastante flexível de acesso à dados. Enquanto o data warehouse extrai, transforma e limpa os dados dos sistemas transacionais, mantendo-os integrados em quantidades massivas e em seu nível mais baixo, o data mart se serve destes dados, extraindo dados para um departamento ou uma área de negócio, oferecendo flexibilidade e controle ao usuário final, pois com o data mart é possível fatiar e agrupar dados de diversas maneiras.

Para KIMBALL et al. (1998b), data mart é um subconjunto do data warehouse, direcionado a um departamento ou área de negócio. Para o autor, o conjunto de todos os data marts da organização, construídos de forma incremental, compartilhando dimensões e fatos comuns, segundo um planejamento prévio, formam o data warehouse lógico da organização.

2.4.4. Arquitetura de um Data Warehouse

A arquitetura de um data warehouse pode ser definida como “a forma de representar toda a estrutura do ambiente de dados, comunicação, processamento e apresentação disponível para o usuário na empresa” (SINGH, 2001, p. 80). SINGH (2000) propõe um modelo arquitetônico multicamadas para o data warehouse. Estas camadas estão representadas pela Figura 4 a seguir.

a) Camada de Acesso à Informação.

É a camada através da qual o usuário tem acesso às informações do data warehouse através de relatórios, planilhas e gráficos para análise e apresentação através de softwares como planilhas eletrônicas e bancos de dados desktop e ferramentas específicas para análise dimensional e mineração de dados, que permitem que o usuário final manipule as informações do data warehouse de forma flexível. Esta camada é representada também pelo hardware de acesso a informações manipuladas.

b) Camada de Acesso a Dados

A camada de acesso a dados é responsável por fazer a conexão entre a camada de dados operacionais e a camada de data staging e também pela camada de data warehouse e a camada de acesso à informação, fazendo uso extensivo da linguagem SQL. A utilização de drivers apropriados permite o acesso a praticamente qualquer SGBDs e arquivos de dados, facilitando assim o processo de extração de informações dos sistemas de processamento transacionais.

Esta camada é composta por diferentes SGBDS, hardware, sistema operacional e protocolos de rede, a fim de proporcionar acesso transparente, independentemente da localização e da plataforma do usuário.

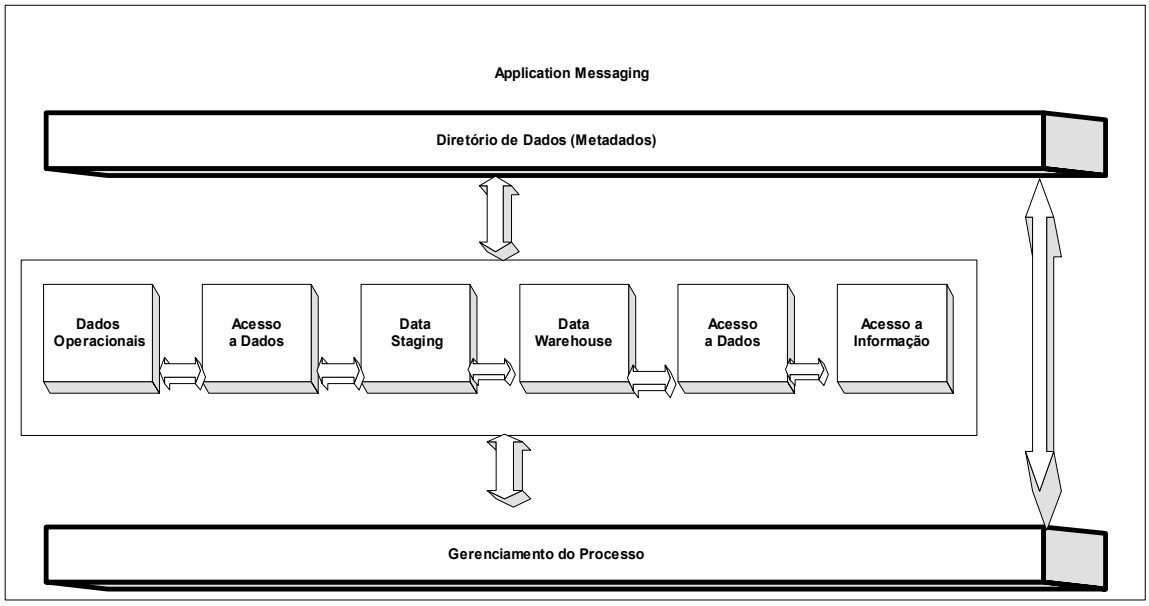
c) Camada de Diretório de Dados (Metadados)

Metadados são os dados que descrevem os dados armazenados. Esta camada permite que o usuário do data warehouse acesse seus dados de forma única, sem que seja necessário saber onde eles estão e em que formato estão armazenados.

Segundo BRACKET, apud SELL (2001), existem basicamente dois tipos de metadados: os metadados técnicos, utilizado pelo corpo técnico (DBAs e desenvolvedores) para manutenção do data warehouse, e os metadados de negócio, utilizado pelos usuários finais, que fornecem uma descrição do negócio, regras e cálculos de formação dos dados e outros itens que auxiliem na interpretação dos dados e informações sobre freqüência de atualização dos mesmos.

Um problema com os metadados é a padronização dos mesmos. Existem diversos fornecedores de soluções de data warehouse, cada um com sua arquitetura proprietária, o que obriga os profissionais do corpo técnico a manter uma documentação própria, a fim de gerenciar o data warehouse e fornecer os metadados de negócio de forma padronizada ao usuário final (KIMBALL, 1998a).

Figura 4 - Uma arquitetura multicamadas para o data warehouse



Fonte: Adaptado de SINGH, H. Data Warehouse: Conceitos, Tecnologias, Implementação e Gerenciamento. São Paulo: Makron Books, 2001.

Em um ambiente de data warehouse, o metadado é utilizado para fornecer informações sobre a localização do conteúdo do data warehouse, mapeando os dados à medida que os mesmos são transformados do ambiente operacional para o ambiente de date warehouse, informações à respeito dos algoritmos utilizados na sumarização dos dados, informações sobre as estruturas dos dados, histórico sobre a extração e transformação dos dados e estatísticas sobre a utilização dos dados.

d) Camada de Gerenciamento do Processo

Esta camada funciona como um organizador dos diversos processos que atuam em um data warehouse para mantê-lo atualizado, envolvendo todas as tarefas necessárias a construção e manutenção do data warehouse.

e) Camada Application Messaging

É o sistema de transporte de dados subjacente ao data warehouse. Esta camada é responsável pelo transporte de informações do data warehouse pela rede da empresa.

f) Camada de Dados Operacionais

Sistemas OLTP foram projetados para suportar transações bem definidas e relativamente pequenas. O objetivo de um data warehouse é liberar informações que estão presas nos sistemas OLTP e mistura-las com informações de fontes externas para que estas possam ser úteis à tomada de decisão. A camada de Dados Operacionais é o próprio banco de dados OLTP da organização.

g) Camada Data Warehouse (Física)

É a camada onde estão armazenados os dados do data warehouse propriamente dito. O data warehouse pode ser apenas uma visualização lógica ou uma cópia dos dados operacionais e dados externos livres de inconsistências e em um formato que proporcione um acesso rápido e flexível.

h) Camada Data Staging

Esta camada envolve todos os processos e ferramentas necessárias à extração, transformação e limpeza dos dados antes que estes passem para o data warehouse físico. É uma camada onde os dados permanecem até serem limpos e padronizados antes do processo de carga do data warehouse.

2.4.5 Abordagens de Implementação

As abordagens de implementação se referem à forma como o data warehouse é projetado e implementado em relação às necessidades globais da empresa. As três abordagens de implementação descritas neste trabalho são a abordagem top-down, a abordagem bottom-up e a arquitetura BUS.

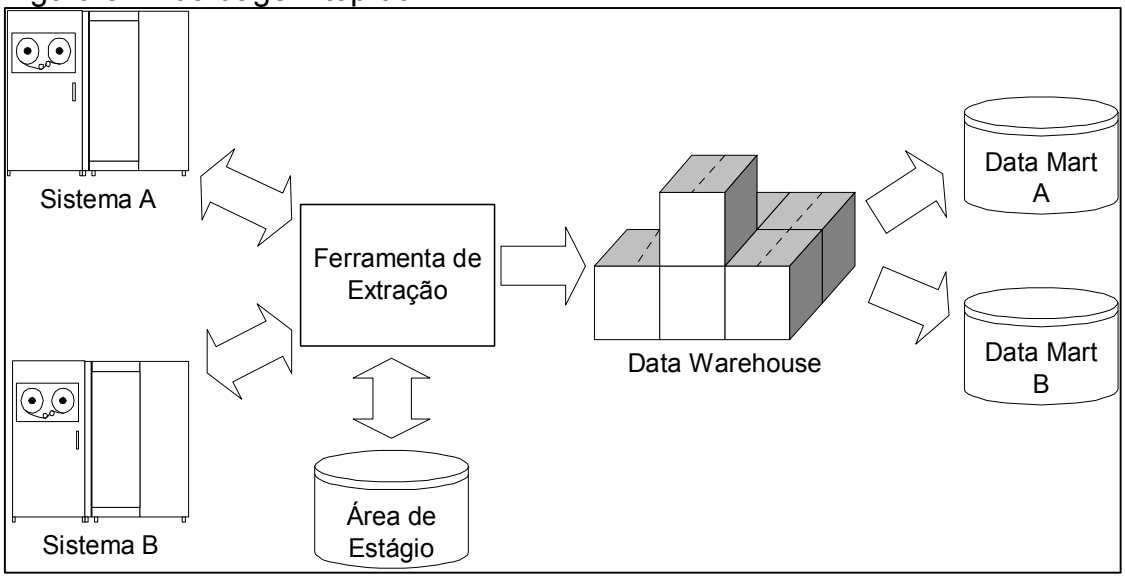
a) Abordagem Top-Down

Este tipo de abordagem foi proposta por INMON (1997) e baseia-se em um data warehouse corporativo central, baseado no modelo relacional e totalmente normalizado. O processo de extração, transformação e carga e, conseqüentemente, a área de estágio de dados são implementados de forma única e integrada. Este data warehouse corporativo contém todos os dados organizacionais e tem o

propósito de servir de base de dados para os diversos data marts departamentais implementados com base no modelo dimensional (Figura 5).

Este tipo de abordagem, apesar de fornecer uma base de dados corporativa única, homogênea e totalmente integrada, traz consigo problemas relativos a altos custos de implementação e demora na apresentação de resultados (VASCONCELLOS, 1999, BALLARD et al. 2001).

Figura 5 - Abordagem top down



Fonte: SELL, D. Uma arquitetura para distribuição de componentes tecnológicos para sistemas de informações baseados em data warehouse. Florianópolis, p.48, 2001. Dissertação (Mestrado em Engenharia de Produção) – Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina.

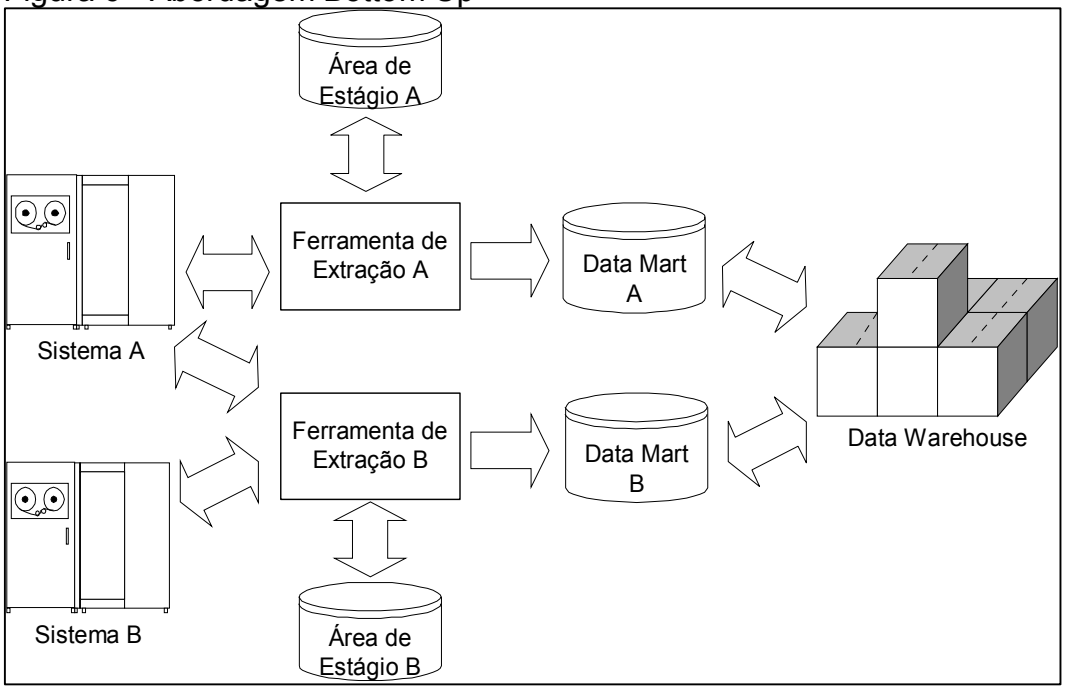
b) Abordagem Bottom Up

Esta abordagem se baseia na construção de data marts dimensionais independentes (Figura 6), cada qual com sua respectiva área de estágio e processos de extração, transformação e limpeza e metadados. O data warehouse, neste tipo de implementação é composto pelo conjunto de todos os data marts construídos, formando um data warehouse incremental lógico (VASCONCELLOS, 1999, FIRESTONE, 2001, BALLARD et al. 2001).

As vantagens deste tipo de abordagem são a rapidez e a simplicidade do projeto de desenvolvimento dos data marts, o que possibilita a disponibilização dos primeiros data marts em espaços de tempo relativamente curtos. Entretanto, a falta de um processo de planejamento prévio acarreta problemas perceptíveis somente a longo prazo. O processo de desenvolvimento independente acarreta inconsistência

entre os metadados de cada data mart, resultando no que se convencionou chamar de legamarts, ou data marts legados (FIRESTONE, 2001). Os processos de extração, transformação e limpeza independentes também acarretam nos recursos utilizados, tais como hardware e infraestrutura de redes (VASCONCELLOS, 1999, BALLARD et al, 2001).

Figura 6 - Abordagem Bottom Up



Fonte: SELL, D. Uma arquitetura para distribuição de componentes tecnológicos para sistemas de informações baseados em data warehouse. Florianópolis, p.48, 2001. Dissertação (Mestrado em Engenharia de Produção) – Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina.

c) Arquitetura BUS

Esta arquitetura foi proposta por KIMBALL et al. (1998b), como forma de unir as vantagens da abordagem top down às vantagens da abordagem bottom up. Neste tipo de arquitetura, o data warehouse lógico e incremental é formado por vários data marts planejados e integrados. Os data marts são construídos segundo o processo de planejamento. Desta forma obtém-se um data warehouse incremental e totalmente integrado aliado a um retorno rápido do investimento mediante a construção de data marts (Figura 7).

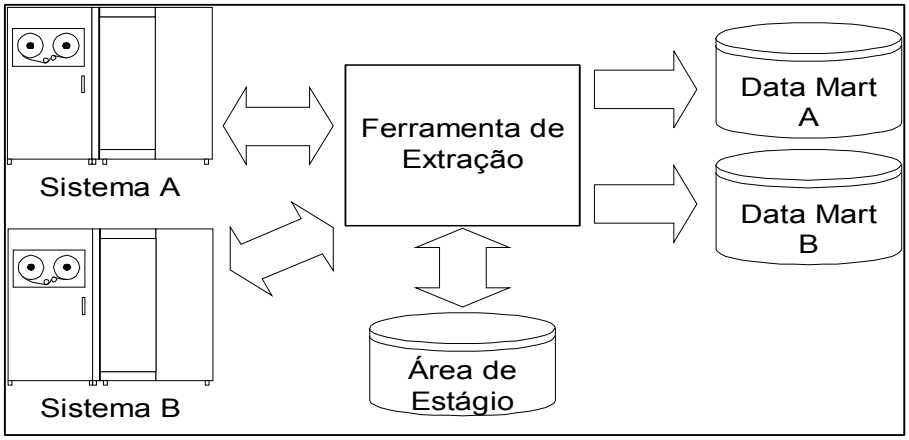
O planejamento consiste em especificar uma arquitetura global de dados com um escopo bem definido e um conjunto específico de metas e seguir este planejamento

construindo data marts incrementais totalmente aderentes aos padrões da arquitetura planejada.

Durante o processo de planejamento são definidos dimensões comuns e fatos padronizados. As dimensões comuns são definidas em nível atômico, ou seja, com menor grão possível a fim de poderem ser relacionadas a qualquer tabela de fatos.

Ao mesmo tempo em que são definidas as dimensões comuns, também são definidos os fatos padronizados. Estes fatos devem possuir a mesma nomenclatura entre os diversos data marts que formarão o data warehouse lógico e mantidos em nível atômico.

Figura 7 - A arquitetura BUS



Fonte: SELL, D. Uma arquitetura para distribuição de componentes tecnológicos para sistemas de informações baseados em data warehouse. Florianópolis, p.49, 2001. Dissertação (Mestrado em Engenharia de Produção) – Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina.

2.4.6. Estrutura de Custos de um Data Warehouse

Os custos de um data warehouse começam nos estágios iniciais do desenvolvimento e continuam durante todo o ciclo de vida do mesmo. A estrutura de custos está distribuída entre os seguintes componentes (KIMBALL, 1998b):

- **Licenças de hardware e software.** Este item inclui todo o hardware e softtware e seus upgrades. Devem ser considerados o hardware servidor e cliente, licenças de bancos de dados, ferramentas de data staging, ferramentas de conectividade e softwares para desktop.

- **Despesas contínuas de manutenção.** Considerar neste item todas as despesas de manutenção de hardware e os contratos de manutenção e suporte de software.

- **Desenvolvimento interno.** Caso o data warehouse seja desenvolvido internamente, os custos de desenvolvimento devem ser considerados. Caso seja contratado desenvolvimento externo, deve-se considerar os custos do planejamento preliminar e acompanhamento do projeto como custos de desenvolvimento interno.

- **Recursos de desenvolvimento externos, se requeridos.** Entram neste item todos os recursos externos contratados para o desenvolvimento do data warehouse que não se encaixam nos itens anteriores.

- **Despesas educacionais com o pessoal de desenvolvimento e a comunidade de usuários.**

- **Suporte pós-implantação.** Despesas com suporte ao usuário de data warehouse.

- **Despesas com crescimento da instalação.** Devem ser consideradas neste item despesas geradas pelo aumento da população de usuários, crescimento das bases de dados e alterações nos requerimentos do negócio.

4.7 Modelagem Dimensional

2.4.7.1 Conceito

A modelagem dimensional, ou esquema em estrela é uma técnica de modelagem de dados voltada especialmente para a implementação de um modelo de dados que permita a visualização de dados de forma intuitiva e com altos índices de performance na extração de dados (KIMBALL et al., 1998b, MACHADO, 2000). O modelo dimensional proporciona uma representação do banco de dados consistente com o modo como o usuário visualiza e navega pelo data warehouse, combinando tabelas com dados históricos em séries temporais, cujo contexto é descrito através

de tabelas de dimensões (BALLARD et al., 2001, HARRISON, 1998). O modelo dimensional é baseado em três elementos:

- Fatos;
- Dimensões;
- Medidas.

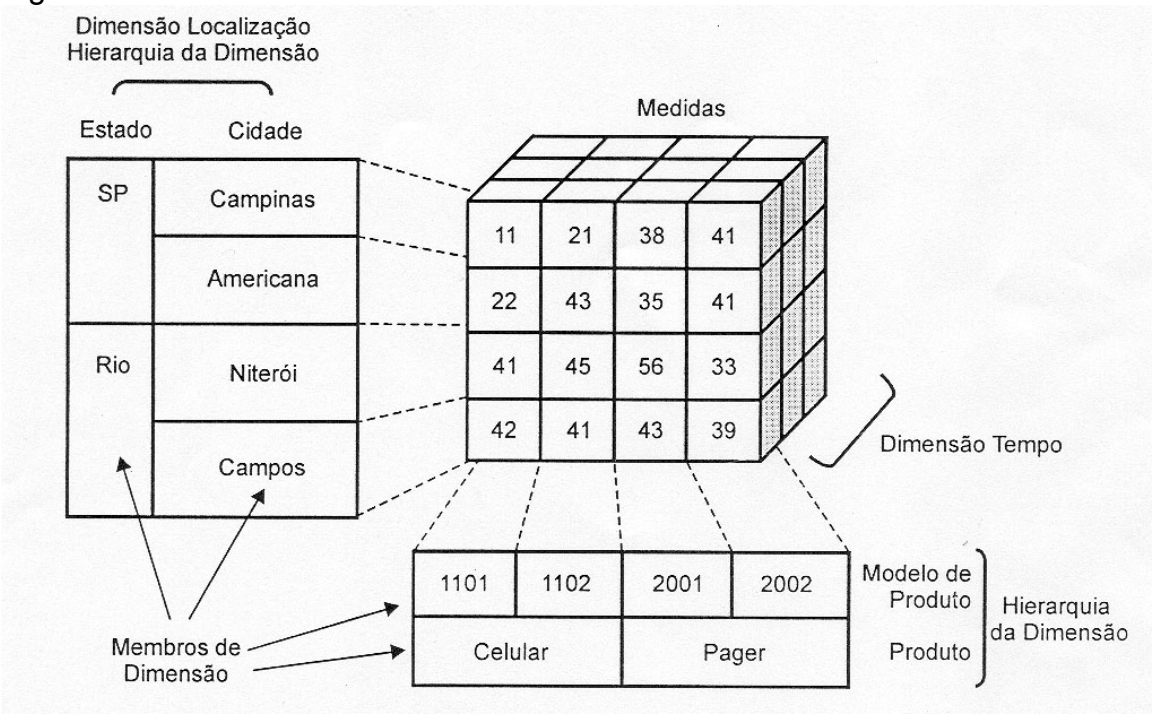
Um fato é uma coleção de itens de dados composta de dados de medida e dados de contexto. Dados de medida são as medições numéricas do negócio e os dados de contexto são chaves estrangeiras apontando para cada uma das dimensões. Cada fato pode representar uma determinada transação ou evento do negócio ocorrido em um determinado contexto obtido na intersecção das dimensões (KIMBALL et al., 1998b, MACHADO, 2000).

Uma dimensão se refere ao contexto em que um determinado fato ocorreu, tais como períodos de tempo, produtos, mercados, clientes e fornecedores elementos que possam descrever o contexto de um determinado fato (HARRISON, 1998, MACHADO, 2000), classificando as medições ativas de uma organização (KIMBALL, 1998a).

Medidas são atributos que quantificam um determinado fato, representando a performance de um indicador em relação às dimensões que participam do fato. O contexto de uma medida é determinado em função das dimensões que participam do fato (MACHADO, 2000, KIMBAL, 1998).

O modelo dimensional permite a visualização de dados na forma de um cubo, onde cada dimensão do cubo representa o contexto de um determinado fato, e a intersecção entre as dimensões representa as medidas do fato. (Figura 8). Matematicamente o cubo possui apenas três dimensões, entretanto, no modelo dimensional a metáfora do cubo pode possuir quantas dimensões forem necessárias para representar um determinado fato (MACHADO, 2000). Dimensões, fatos e medidas terão seus conceitos discutidos detalhadamente nas seções à frente.

Figura 8 - O cubo de decisão



Fonte: MACHADO, F. N. Projeto de Data Warehouse: uma visão multidimensional. Rio de Janeiro: Editora Érica, p. 66, 2000.

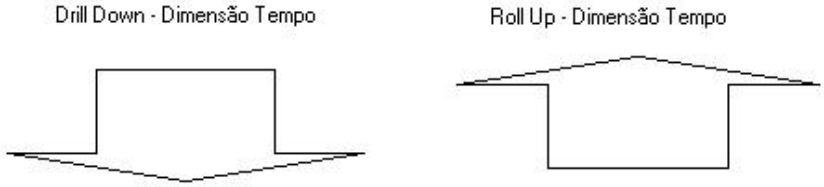
2.4.7.2 Navegação pelo modelo dimensional

As operações básicas para navegação por um modelo dimensional são drill down/roll up, drill across e slice and dice.

Drill down e Roll Up (Figura 9) são operações utilizadas para movimentar a visão verticalmente na hierarquia de uma dimensão. Através do drill down o usuário navega de um nível mais alto de detalhe até um nível mais baixo, enquanto que na operação roll up o usuário navega de um nível mais baixo de detalhe até o nível mais alto (BALLARD et al., 2001 MACHADO, 2000, SINGH, 2001). Para KIMBALL (1998a), efetuar drill down não significa apenas navegar verticalmente na hierarquia de uma dimensão. Significa obter e remover rapidamente cabeçalhos de linha de qualquer uma das dimensões da tabela de fatos.

Figura 9 - Drill down e roll up

Volume de Produção (em milhares)		1999			
		Trim. 1	Trim. 2	Trim. 3	Trim. 4
Região Sul	RS	78	67	22	56
	SC	90	67	88	99



Volume de Produção (em milhares)		Trimestre 1		
		Janeiro	Fevereiro	Março
Região Sul	RS	30	26	22
	SC	28	30	32

Fonte: MACHADO, F. N. Projeto de Data Warehouse: uma visão multidimensional. Rio de Janeiro: Editora Érica, p. 70, 2000.

Drill across combina várias tabelas de fato que compartilham as mesmas dimensões em um único relatório (KIMBALL, 1998a)

Slice and Dice (fatiar e girar) são operações que “permitem acessar um data warehouse por meio de qualquer uma de suas dimensões de forma igual” (KIMBALL, 1998a, p. 319), ou seja, através destas operações é possível navegar através dos dados do cubo de decisão ao longo de qualquer dimensão (SINGH, 2001).

Slice (Figura 10) corta o cubo mantendo a mesma perspectiva de visualização dos dados, permitindo que o usuário fixe a apresentação em um determinado detalhe (BALLARD et al., 2001, MACHADO, 2000).

Dice, ou rotação (Figura 11) significa mudar a perspectiva de visão, girando o cubo e invertendo uma ou mais dimensões (BALLARD et al., 2001, KIMBALL, 1998a, MACHADO, 2000).

Figura 10 - Slice

Volumes		Celulares e Pagers		
		Janeiro	Fevereiro	Março
Região Sul	RS	30	26	22
	SC	28	30	32

slice

Volumes		Celulares		
		Janeiro	Fevereiro	Março
Região Sul	RS	22	18	18
	SC	19	27	25

Fonte: MACHADO, F. N. Projeto de Data Warehouse: uma visão multidimensional. Rio de Janeiro, p. 71-72, 2000.

Figura 11 - Dice

Volumes	1999			
	Pré-Pagos		Pós-Pagos	
	Masculino	Feminino	Masculino	Feminino
PR	30	40	80	50
SC	20	15	60	30

Dice

Volumes		1999	
		Pré-Pagos	Pós-Pagos
PR	Masculino	30	80
	Feminino	40	50
SC	Masculino	20	60
	Feminino	15	30

Fonte: MACHADO, F. N. Projeto de Data Warehouse: uma visão multidimensional. Rio de Janeiro, p. 71-72, 2000

2.4.7.3 Fatos

O objetivo de um data warehouse não é armazenar informações a respeito de documentos, tais como pedidos ou notas fiscais, mas sim informações à respeito do assunto que envolve tais documentos, tais como as vendas da organização (MACHADO, 2000).

Fato é tudo aquilo que pode ser representado por meio de valores mensuráveis, geralmente numéricos (MACHADO, 2000, KIMBAL, 1998) e que são objeto de análise (BALLARD et al., 2001). Fatos, portanto podem ser caracterizados por algo que ocorre de forma variável ao longo do tempo e podem ser expressos em valores mensuráveis.

Fatos podem ser aditivos, semi-aditivos e não aditivos (KIMBALL, 1998a, KIMBALL, 1998b):

- **Fato aditivo** é aquele que pode ser adicionado a qualquer uma das dimensões. Os fatos mais úteis em um banco de dados são os fatos aditivos e continuamente valorados (diferentes a cada medida), pois praticamente todas as consultas podem ser feitas sobre estes fatos. Uma tabela de fatos aditivos pode ser compactada facilmente através de processos de sumarização, produzindo consultas de apenas algumas linhas a partir de tabelas extremamente extensas.

- **Fato semi-aditivo** é aquele que só pode ser adicionado ao longo de algumas dimensões, o que restringe o número de consultas apenas àquelas dimensões em que o fato pode ser adicionado.

- **Fato não aditivo** é aquele que não pode ser adicionado a qualquer das dimensões. Para este tipo de fato, só se pode resumir registros através de contagens ou então consulta-los um a um.

a) Representação de carteiras de produtos heterogêneos

Algumas organizações, principalmente as instituições financeiras, apresentam uma carteira de produtos bastante extensa, Neste tipo de organização, geralmente é necessário observar os fatos de acordo com duas perspectivas: uma global e uma por linhas de produtos (KIMBALL, 1998a, KIMBALL , 1998b).

Sob a perspectiva global, deseja-se visualizar toda a carteira de produtos em um único cubo a partir de uma perspectiva de fatos comuns a toda a linha de produtos. Este tipo de solução inviabiliza a utilização de uma tabela de fatos para cada um dos produtos, devido ao grande número de tabelas consultadas para formar o cubo.

Neste caso a solução é criar uma única tabela de fatos com as medidas comuns a cada um dos produtos (KIMBALL, 1998a, KIMBAL, 1998b).

Sob uma perspectiva de linhas de produtos deseja-se analisar uma linha em especial, com todos os seus fatos particulares, neste caso a construção de uma tabela de fatos global incorreria na existência de muitos valores nulos. A solução para este tipo de problema é a criação de múltiplas tabelas de fatos de acordo com as diferentes linhas de produtos (KIMBALL, 1998a, KIMBALL, 1998b). Para organizações que possuem uma vasta carteira de produtos é natural utilizar os dois tipos de solução simultaneamente (KIMBALL, 1998a).

b) Transações e instantâneos

Existem características operacionais do data warehouse que independem do tipo de negócio em que o mesmo está inserido. Freqüentemente, é necessário que existam duas versões de implementação de um determinado fato: uma representando cada transação individualmente e outra representando instantâneos retirados em períodos regulares de tempo (KIMBALL, 1998a).

A representação ao nível transacional é o nível mais detalhado que se pode obter na representação dos fatos, representando, geralmente, a cada fato individualmente. É a representação mais fácil de se obter na implementação do modelo dimensional e a que permite análises mais ricas, que não podem ser obtidas a partir de fatos sumarizados (KIMBALL, 1998b).

Algumas informações mais gerais, tais como a lucratividade de uma categoria de produtos em um intervalo de tempo, são difíceis, ou não podem, ser extraídas a partir da sumarização das transações. A solução para estes problemas é a criação de tabelas de fato contendo instantâneos (snapshots) retirados em determinados intervalos de tempo. Estes instantâneos contém não somente a sumarização das transações individuais, mas também informações indiretas, tais como a margem de lucro obtida no período (KIMBALL, 1998a).

c) Agregados

Agregados são resumos construídos a partir de fatos individuais, inicialmente por questões de performance, ou quando o ambiente dos fatos é inexpressivo na menor

granularidade (KIMBALL, 1998a, KIMBALL, 1998b). Agregados permitem que as aplicações antecipem os resultados a serem pesquisados pelo usuário eliminando a necessidade de se repetir cálculos comumente realizados (SINGH, 2001).

Múltiplos agregados podem ser construídos, representando os agrupamentos mais comuns dentro das dimensões do data warehouse a fim de aumentar a performance. A contrapartida ao aumento da performance é o aumento do consumo de espaço de armazenamento (KIMBALL, 1998b).

d) Tabelas de fatos sem fatos

Tabelas de fatos sem fatos são utilizadas para armazenar fatos que não podem ser associados a uma medida numérica. Tais tabelas podem ser utilizadas no rastreamento de eventos, como a freqüência dos alunos nas salas de uma escola, ou todos os elementos envolvidos em um acidente coberto por uma seguradora. Tabelas de fatos sem fatos também podem ser utilizadas ou para cobrir eventos que não ocorreram, como relacionar os produtos que não foram vendidos em uma determinada promoção (Figura 12) (KIMBALL, 1998a, KIMBALL, 1998b).

2.4.7.4 Dimensões

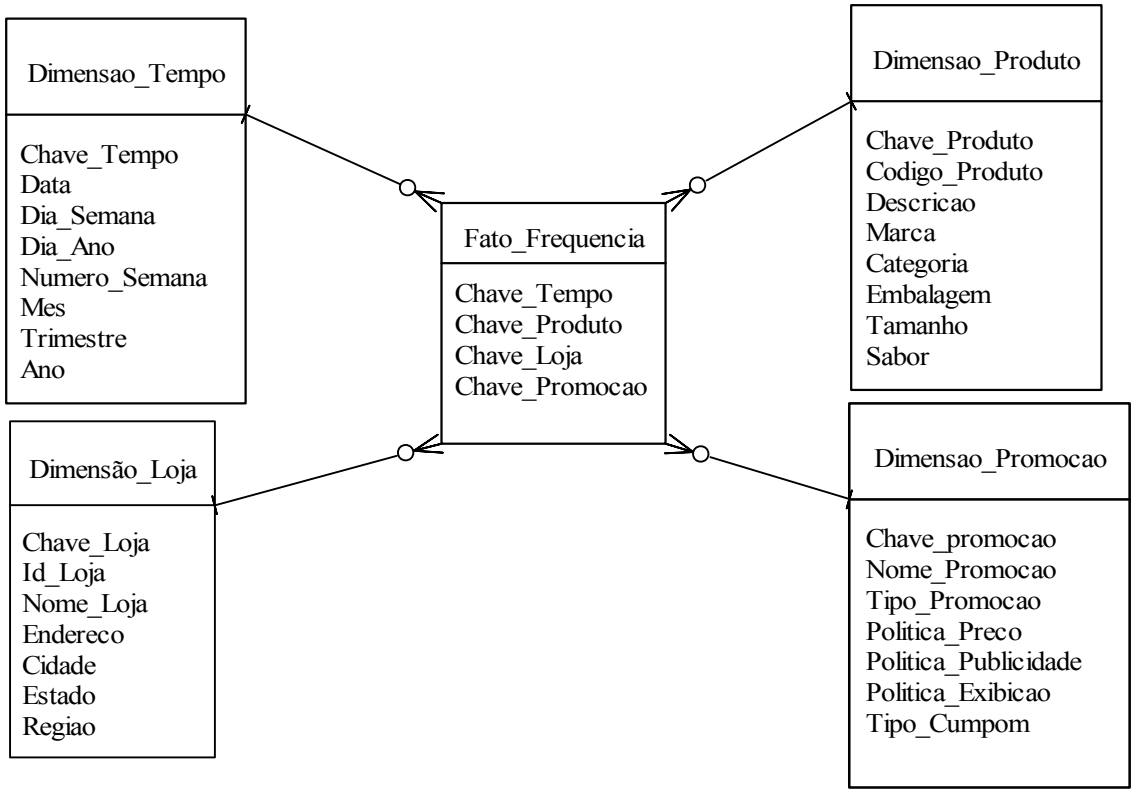
Dimensões determinam o contexto em que ocorreram os fatos. No modelo dimensional, cada dimensão está associada a um ou mais fatos, sendo estas usualmente mapeadas em entidades não numéricas e informativas (BALLARD et al, 2001).

Uma dimensão pode conter diversos membros e estes podem ser arranjados em hierarquias. Um membro de uma dimensão é um nome único utilizado para caracterizar um determinado dado. Por exemplo, os membros da dimensão tempo podem ser data, ano, mês, dia do mês, semana do mês, semana do ano etc. Os membros de uma dimensão podem ser arranjados em várias hierarquias, cada hierarquia contendo vários níveis (BALLARD et al. 2001), conforme a Figura 13.

MACHADO (2000) afirma que a maioria dos fatos envolve pelo menos quatro dimensões básicas: onde, quando, quem e o que (Figura 15). A dimensão Onde, determina o local onde o fato ocorreu (local geográfico, filial). A dimensão Quando, é a própria dimensão tempo. A dimensão Quem, determina que entidades participaram

do fato (cliente, fornecedor, funcionário). A dimensão O quê determina qual é o objeto do fato (produto, serviço).

Figura 12 - Tabela de fatos sem fatos



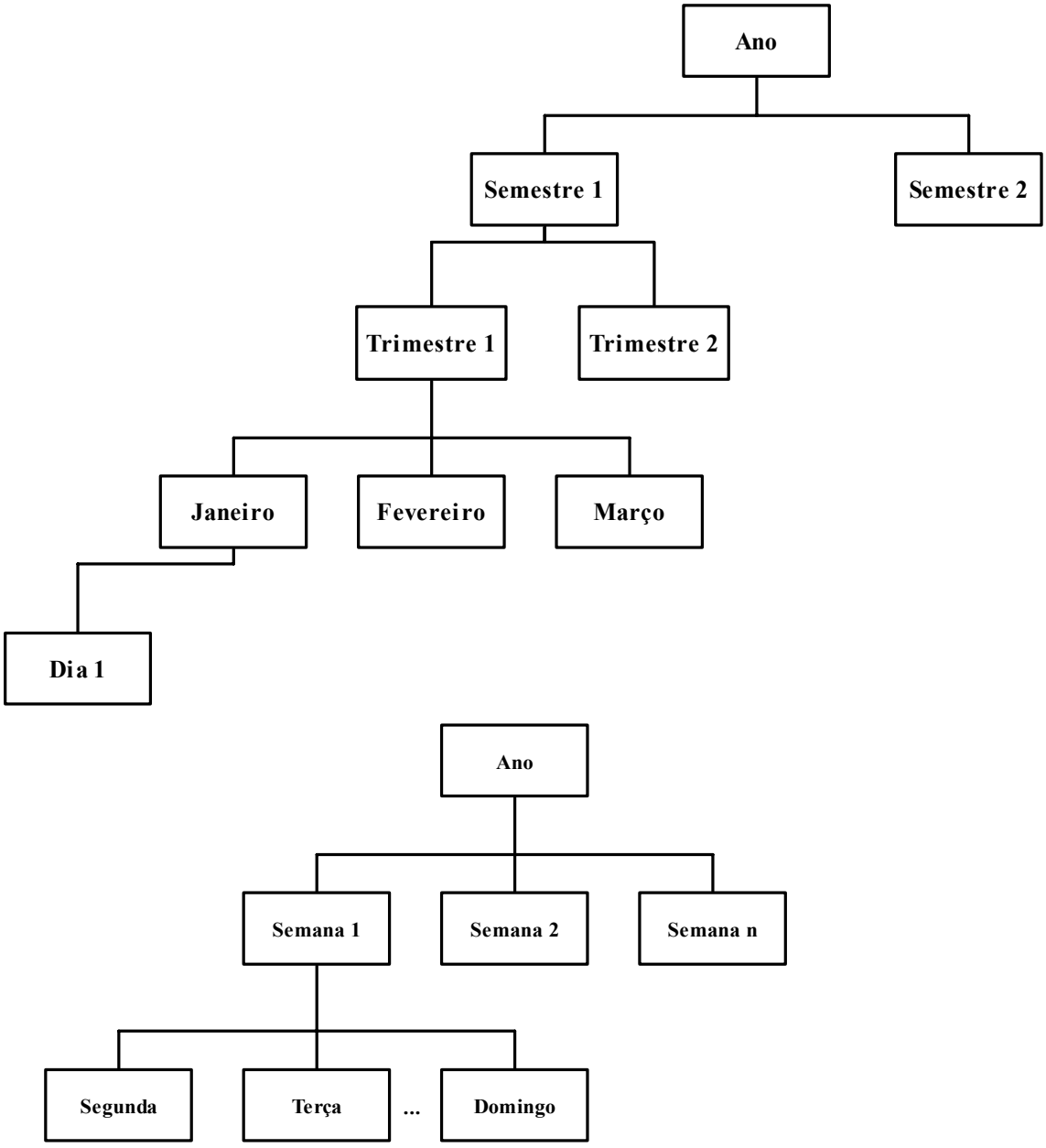
Fonte: Adaptado de KIMBALL, R. The Data Warehouse Lifecycle Toolkit: Expert methods for designing and deploying data warehouses. New York: John Wiley & Sons Inc, 1998.

Alguns tipos de dimensão merecem ser estudadas mais profundamente devido a alguns tratamentos especiais dados a elas. A seguir é apresentada uma discussão a respeito da dimensão tempo, dimensões de modificação lenta, dimensões grandes, minidimensões e dimensões sujas, dimensões muitos para muitos e chaves artificiais.

a) A dimensão tempo

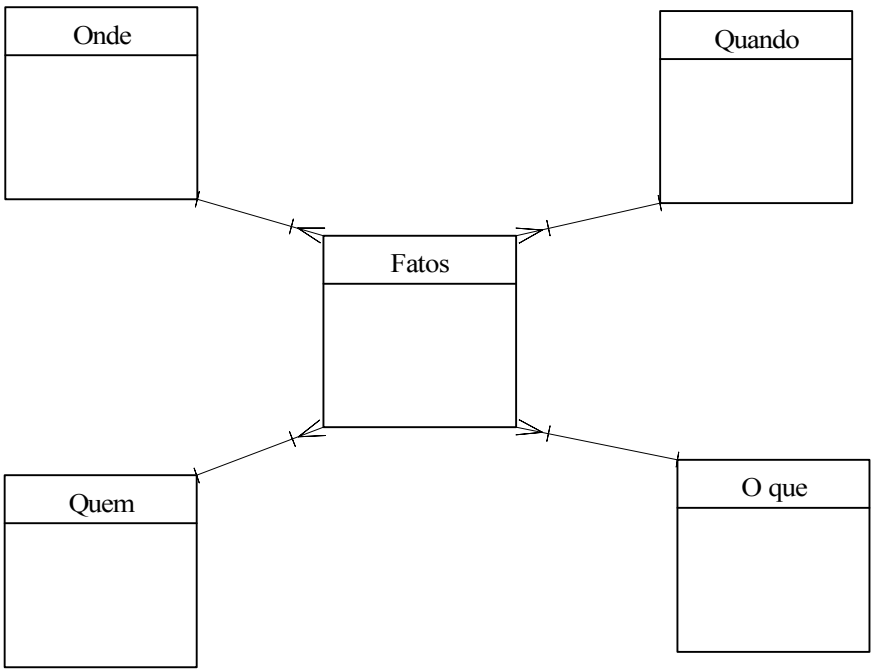
Quase todo o data warehouse envolve uma série temporal, portanto, a dimensão tempo está presente em quase todos os data warehouses. O motivo para que se crie uma dimensão tempo é representar todos os intervalos de tempo necessários ao negócio em questão, tais como períodos fiscais, temporadas, dias não úteis e outras unidades de tempo necessárias ao negócio em questão (KIMBALL, 1998a, KIMBALL, 1998b).

Figura 13 - Hierarquias na dimensão tempo



Fonte: Adaptado de BALLARD, C. et al. Data Modeling Techniques for Data Warehousing. IBM Corporation. Disponível em: <http://www.redbooks.ibm.com> Acesso em 21/06/2001.

Figura 14 - Dimensões comuns em um modelo dimensional



Fonte: MACHADO, F. N. Projeto de Data Warehouse: Uma Visão Multidimensional. São Paulo: Érica, 2000, p. 95.

b) Dimensões de modificação lenta

A maioria das dimensões se modifica lentamente ao longo do tempo. Para que se possa rastrear as modificações que uma dimensão pode sofrer, utiliza-se um dos três métodos a seguir (KIMBALL, 1998b, KIMBALL, 1998a):

- Substituir os valores antigos do registro modificado por valores novos e, assim, perder a capacidade de rastrear o histórico da dimensão.
- Adicionar um novo registro à dimensão com os valores novos do atributo, segmentando o histórico entre a antiga e a nova dimensão.
- Criar novos campos para os valores atuais e os valores antigos, mantendo o histórico da dimensão em um único registro, mas reduzindo a capacidade de manter o histórico da dimensão.

c) Dimensões grandes

Existem casos em que algumas dimensões podem conter milhões de entradas, por exemplo, a dimensão cliente em uma companhia telefônica, neste caso a navegação por esta dimensão pode tornar-se demorada. Pode-se, neste caso, utilizar índices nos atributos que sejam objetos de navegação (KIMBALL, 1998a).

d) Minidimensões

Freqüentemente, os campos mais utilizados em uma dimensão grande possuem um domínio pequeno, ou seja, assumem uma pequena quantidade de valores. Em uma dimensão cliente, estes atributos podem ser atributos demográficos, como sexo, faixa etária e classe social. Neste caso, pode-se optar pela criação de uma minidimensão separada da dimensão cliente para aumentar a eficiência da navegação (KIMBALL, 1998a).

c) Dimensões descaracterizadas

Atributos como o número da nota fiscal de venda, aparentemente, deveriam fazer parte da tabela de fatos. Em um banco de dados relacional, ele seria o atributo determinante do cabeçalho da nota fiscal. Em um banco de dados dimensional, normalmente, todos os atributos determinados pelo número da nota foram armazenados em dimensões próprias e faria parte da chave primária dos itens da nota. Ainda assim, pode-se utilizar este atributo para agrupar os fatos pelo documento original. Atributos deste tipo são representados como dimensões descaracterizadas, isto é, chaves de dimensão sem uma dimensão correspondente (KIMBALL, 1998a).

d) Dimensões sujas

Dimensões sujas ocorrem quando uma entidade aparece diversas vezes, podendo haver ligeiras diferenças em alguns de seus atributos. Em um data warehouse bancário, a dimensão conta, contendo atributos do correntista, é uma dimensão suja, pois em caso de conta conjunta, a conta aparecerá várias vezes, uma para cada correntista (KIMBALL, 1998a).

e) Chaves artificiais

Utilizar uma chave candidata como chave primária de uma dimensão pode causar problemas, caso esta chave não seja absolutamente estável ao longo do tempo. Uma modificação em uma chave de uma dimensão pode ocasionar um grande volume de mudanças nas tabelas de fato relacionadas à esta dimensão. A solução para este problema é utilizar chaves artificiais absolutamente estáveis ao longo do tempo. Uma chave artificial é um campo inteiro e autoincremental, que aumenta a cada novo registro incluído na tabela de dimensão (KIMBALL, 1998b).

f) Dimensões muitos para muitos

Na maioria dos casos, o relacionamento entre uma dimensão e a tabela de fatos é um para muitos. Entretanto, podem existir casos em que um registro de um fato pode estar ligado a muitas ocorrências de uma determinada dimensão. Um exemplo é um data warehouse para hospitais, onde um único internamento pode estar ligado a vários diagnósticos. A solução para este problema é a mesma utilizada em um modelo relacional, ou seja, dividir o relacionamento muitos para muitos em dois relacionamentos um para muitos através de uma tabela ponte (ou entidade associativa) entre a tabela de fatos e a tabela de dimensão (Figura 15) (KIMBALL, 1998b).

2.4.7.5 Granularidade

A granularidade se refere ao nível de detalhe em que serão armazenados os dados no data warehouse (BALLARD et al., 2001, KIMBALL, 1998a, INMON, 1997), quanto maior o detalhamento, mais baixo o nível de granularidade. A granularidade afeta o volume de dados do data warehouse e, portanto, a performance na extração de informações. Um outro aspecto importante é a possibilidade de extração de informações da base de dados. Quanto mais alto o nível de granularidade menores as possibilidades de extração de informações (BALLARD et al., 2001, INMON, 1997).

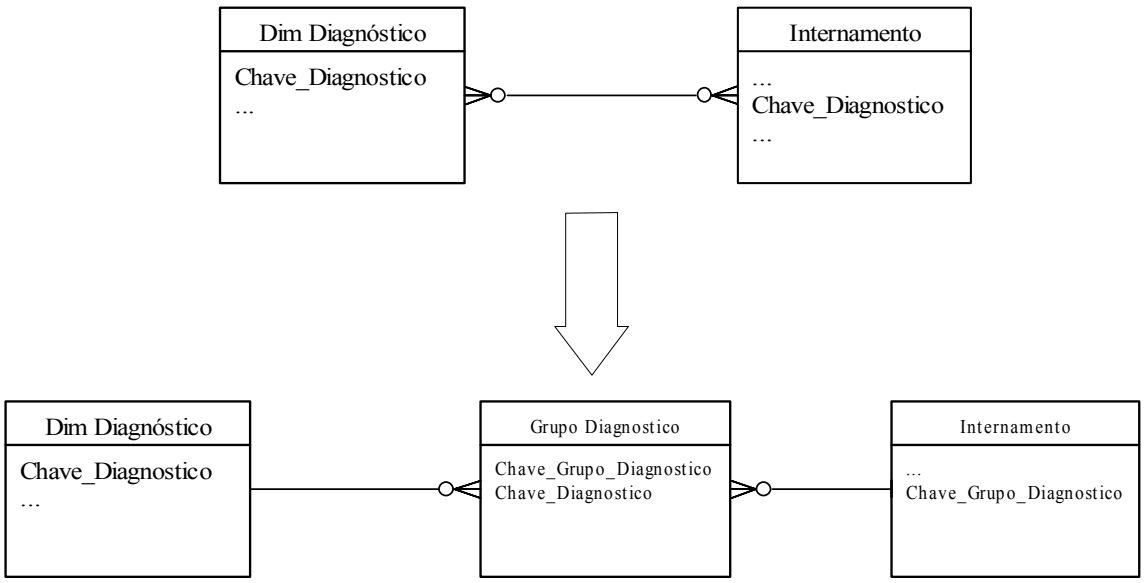
Um data warehouse pode ser implementado em níveis duais de granularidade ao longo do tempo. É possível manter as informações mais recentes em um baixo nível de granularidade, aumentando assim as possibilidades de extração de informações.

A medida que os dados vão ficando obsoletos, é possível resumi-los em um alto nível de granularidade de forma a manter a performance (INMON, 1997).

Opcionalmente, pode-se optar por múltiplos níveis de granularidade em um data warehouse, partindo-se inicialmente de um baixo nível de granularidade, passando por níveis intermediários até um alto nível de granularidade, dependendo das necessidades de informação e do custo de mantê-las no data warehouse (BALLARD et al., 2001).

KIMBALL (1998b), propõe que as tabelas de fato de um data mart na arquitetura bus, assim como as dimensões padrão, devem estar no menor nível de granularidade possível, proporcionando facilidade na alimentação da base de dados e continuidade na extração de informações ao longo do tempo.

Figura 15 - Relacionamento muitos para muitos



Fonte: Adaptado de KIMBALL, R. The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing Data Warehouses. New York: Wiley Computer Publishing, 1998.

2.4.7.6 Particionamento

O particionamento consiste em dividir os dados em unidades físicas menores, de forma a obter uma maior flexibilidade no gerenciamento do banco de dados. O particionamento, proporciona (BALLARD et al., 2001, INMON, 1997):

- facilidade de reestruturação e indexação e reorganização;
- facilidade de recuperação;
- facilidade de monitoramento;
- escalabilidade no data warehouse;
- portabilidade dos elementos do data warehouse.

Os critérios para se particionar um data warehouse dependem dos requisitos do negócio e restrições físicas da base de dados. Em geral, os critérios de particionamento podem ser os seguintes (BALLARD et al., 2001, INMON, 1997):

- tempo;
- geografia;
- área de negócio ou linha de produtos;
- unidade organizacional;
- qualquer combinação dos critérios acima.

2.4.7.7 Considerações a respeito do modelo dimensional

O modelo dimensional, ao contrário do modelo entidade/relacionamento é bastante assimétrico. Há sempre uma tabela de fatos central, de natureza altamente normalizada conectada diretamente a tabelas de dimensão não normalizadas. Esta estrutura proporciona simplicidade e rapidez ao processo de extração de informações (KIMBALL, 1998a), mas não pode ser aplicada a sistemas OLTP, devido à não normalização das tabelas de dimensão.

2.4.8 O Modelo Snowflake

O modelo snowflake incorpora tabelas dimensionais principais conectadas às tabelas de fato e tabelas dimensionais de extensão, onde são armazenadas as descrições das dimensões. Estas tabelas de extensão são obtidas normalizando-se as dimensões. (BALLARD et al., 2001, HARRISON, 1998, KIMBALL, 1998a, SINGH,

2001). As tabelas dimensionais principais contêm chaves ligando-as às tabelas de extensão. A desvantagem snowflake é a necessidade de múltiplos joins em consultas SQL, caso as consultas necessitem combinar diversas dimensões, o que aumenta a complexidade da instrução e reduz o desempenho HARRISON (1998).

SINGH (2001) afirma que, em caso de tabelas de fatos muito extensas, ligadas a tabelas dimensionais também extensas, o modelo snowflake pode melhorar a performance se houver uma redução significativa no tamanho da tabela de dimensão. Entretanto, KIMBALL (1998a) considera o esquema estrela a única técnica viável para modelagem de dados para o data warehouse. Para BALLARD et al (2001), o modelo snowflake facilita a compreensão e entendimento dos desenvolvedores, mas dificulta o entendimento por parte dos usuários finais. Há também uma redução do espaço ocupado pelas tabelas de dimensão, entretanto, este ganho não é um critério determinante na seleção da técnica de modelagem.

Segundo HARRISON (1998), existem três tipos básicos de projetos snowflake: pesquisa, em cadeia e atributo.

a) Pesquisa

O modelo pesquisa utiliza uma tabela dimensional principal conectada diretamente à tabela de fatos e tabelas de extensões conectadas à tabela dimensional principal (Figura 16).

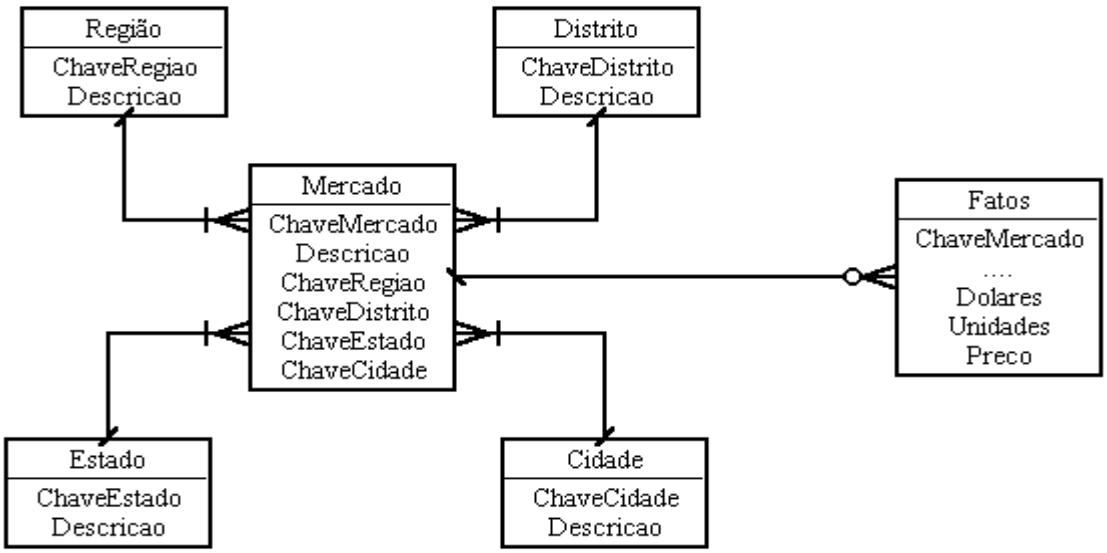
b) Em cadeia

O modelo em cadeia coloca as tabelas de extensão encadeadas, começando com uma tabela dimensional principal conectada à tabela de fatos (Figura 17). A próxima dimensão é conectada à tabela dimensional principal e assim sucessivamente até a última dimensão. Este modelo mantém um alto nível de integridade de dados por que as dimensões são completamente normalizadas.

c) Atributo

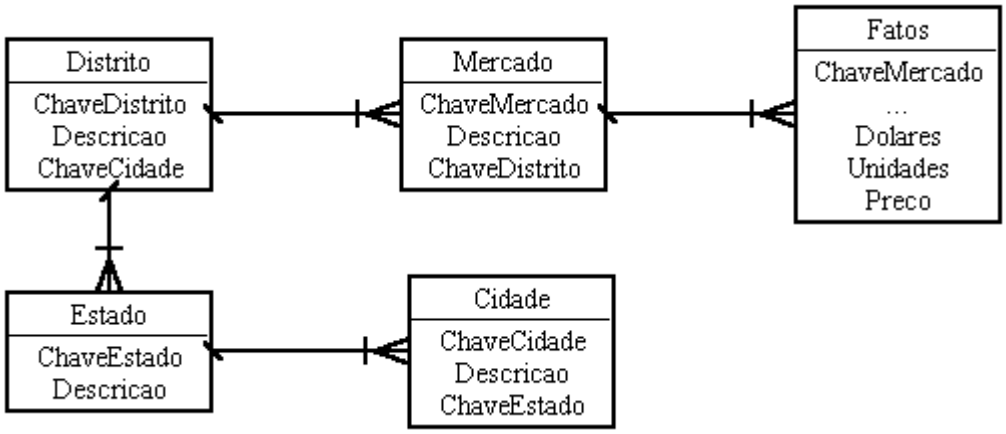
O modelo atributo permite construir uma dimensão principal artificial a partir de vários grupos de atributos dimensionais não relacionados (Figura 18).

Figura 16 - Snowflake do tipo pesquisa



Fonte: Adaptado de HARRISON, T. H. Intranet Data Warehouse. São Paulo: Berkeley, p. 72, 1998.

Figura 17 - Snowflake em cadeia



Fonte: Adaptado de HARRISON, T. H. Intranet Data Warehouse. São Paulo: Berkeley, p. 74, 1998.

2.4.9 O modelo relacional versus Modelo Dimensional

Enquanto o ambiente do modelo relacional é o dos sistemas OLTP, o modelo dimensional destina-se a suportar os sistemas de apoio à decisão. SINGH (2001) traça um quadro comparativo entre o modelo relacional e o modelo dimensional:

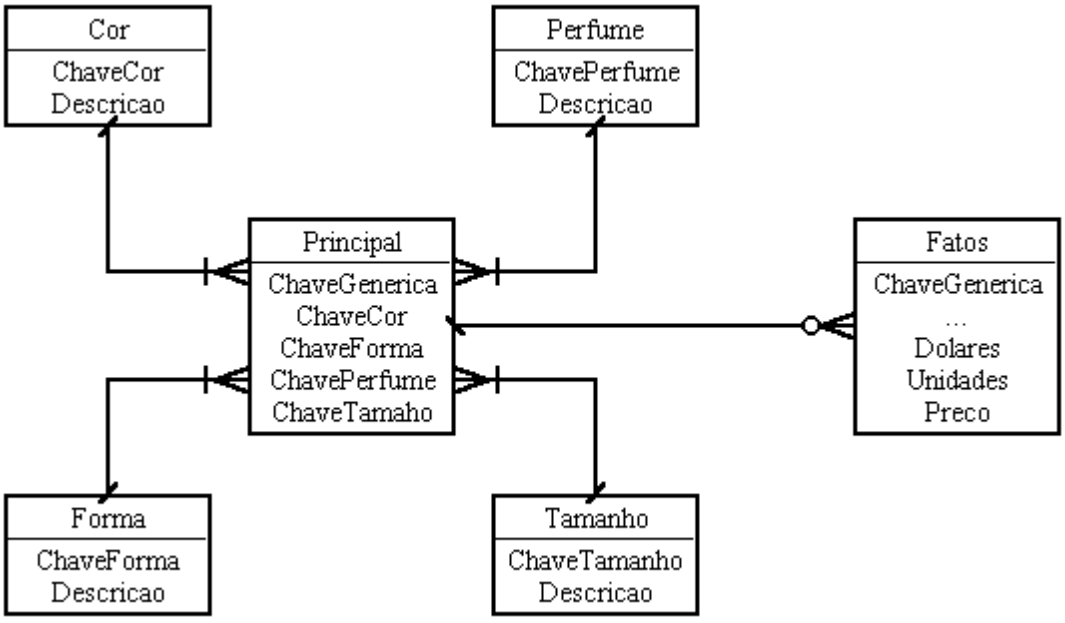
a) Transação versus Intervalo de tempo

A visualização dos dados em um modelo dimensional é feita sob a perspectiva de intervalos de tempo, consistente com os problemas globais da organização, enquanto que o modelo relacional foca transações atômicas processadas em um sistema OLTP.

b) Consistência Local versus Global

A consistência, do ponto de vista do modelo relacional, é a transação em si. Uma transação de uma venda a prazo é consistente se todas as informações desta venda a prazo foram gravadas corretamente no banco de dados. A consistência, do ponto de vista do modelo dimensional tem sua consistência baseada nos aspectos globais da organização.

Figura 18 - Snowflake do tipo atributo



Fonte: Adaptado de HARRISON, T. H. Intranet Data Warehouse. São Paulo: Berkeley, p. 75, 1998.

c) Audit Trail versus Big Picture

O modelo relacional permite o rastreamento de transações, tais como o histórico das faturas do cartão de crédito. O modelo dimensional permite responder questões

normalmente relacionadas ao trabalho do conhecimento, como a possibilidade de lucro ou não de uma categoria de produtos ou oportunidades de negócio.

d) Relacionamentos explícitos versus implícitos

No modelo relacional os relacionamentos estão explícitos entre as entidades do modelo. Há um relacionamento formal entre a entidade cliente e a entidade produto. No modelo dimensional estes relacionamentos estão implícitos pela existência de interseções entre as dimensões na tabela de fatos.

e) OLTP versus data warehouse

O modelo de dados projetado para suportar o data warehouse requer uma otimização voltada para a extração de informações. Neste tipo de aplicativo ocorrem poucas transações acessando um grande número de registros a cada transação. Os sistemas OLTP necessitam de modelos de dados altamente normalizados, visando oferecer um alto desempenho no processamento de um grande número de transações envolvendo poucos registros. A quadro 1 a seguir estabelece um comparativo entre um OLTP e um Data Warehouse.

Quadro 1 - OLTP versus data warehouse

Tópico/Função	Operacional	Data Warehouse
Conteúdo dos Dados	Valores atuais	Dados de arquivo, sumarizados e calculados
Organização dos dados	Aplicação por aplicação	Áreas de assunto ao longo da empresa
Natureza dos dados	Dinâmica	Estática até a reciclagem
Estrutura dos dados: formato	Complexa; adequada para computação operacional	Simples; adequada para análise empresarial
Probabilidade de acesso	Alta	Moderada a baixa
Atualização dos dados	Campo a campo	Acessados e manipulados sem atualização direta
Uso	Processamento repetitivo e altamente estruturado	Processamento analítico e altamente desestruturado

Tópico/Função	Operacional	Data Warehouse
Tempo de resposta	Fração de segundos	Segundos a minutos

Fonte: SINGH, H. S. **Data Warehouse.** Conceitos, Tecnologias, Implementação e Gerenciamento. São Paulo. Makron Books, 2001, p 153-154.

3 METODOLOGIA

3.1 Introdução

Este capítulo caracteriza a pesquisa, descrevendo o problema e a hipótese de pesquisa. Em seguida é descrita a metodologia utilizada para testar o modelo proposto em relação ao modelo relacional e ao modelo dimensional, a fim de verificar a viabilidade do modelo. No próximo capítulo, o modelo proposto será descrito em detalhes e no capítulo 5 serão apresentados os resultados dos testes.

3.1 Caracterização da Pesquisa

Esta pesquisa, por sua natureza, caracteriza-se como uma pesquisa aplicada, pois objetiva gerar conhecimentos para a solução de problemas relativos à extração de informações de bases de dados. Quanto à forma de abordagem, serão explorados aspectos tanto quantitativos quanto qualitativos.

3.1.1 Problema de pesquisa

É possível reduzir os custos de um ambiente de data mart, de forma que empresas com escassez de recursos financeiros possam se beneficiar deste tipo de tecnologia?

3.1.2 Hipótese de pesquisa

A concepção de um modelo híbrido de dados, que una características do modelo entidade/relacionamento e do modelo dimensional possibilita a implementação de sistemas OLTP e data mart em uma mesma base de dados, evitando-se assim os custos da duplicação das bases de dados, uma para sistemas OLTP e outra para data warehouse e data marts.

3.1.3 Etapas da pesquisa

O presente trabalho fundamenta-se em três etapas distintas:

a) Pesquisa bibliográfica a respeito de sistemas de informação, concentrando-se em sistemas OLTP, modelo entidade/relacionamento, sistemas de apoio à decisão, data warehouse e modelo dimensional.

b) Proposta de um modelo híbrido de dados, que atenda tanto aos requisitos dos sistemas OLTP quanto às necessidades de extração de dados do data warehouse. Este modelo, para ser viável, deve suprir as necessidades de informação de empresas, que não possuem recursos para investir em dois ambientes de infraestrutura tecnológica, um para sistemas OLTP e outro para o Data Warehouse.

c) Para investigar a viabilidade do modelo híbrido, são desenvolvidos e comparados dois protótipos de sistema, um baseado no modelo entidade/relacionamento e outro baseado no modelo híbrido. Estes protótipos são comparados para se verificar a viabilidade do modelo.

3.2 Apresentação e Análise dos Resultados

A fim de verificar a viabilidade da utilização do modelo híbrido, os protótipos serão comparados levando-se em conta os seguintes parâmetros:

a) **Performance no processamento transacional.** Para se medir a performance no processamento transacional as bases de dados dos dois modelos são alimentadas a partir da mesma massa de dados. O tempo de carga é o critério para comparação dos dois modelos.

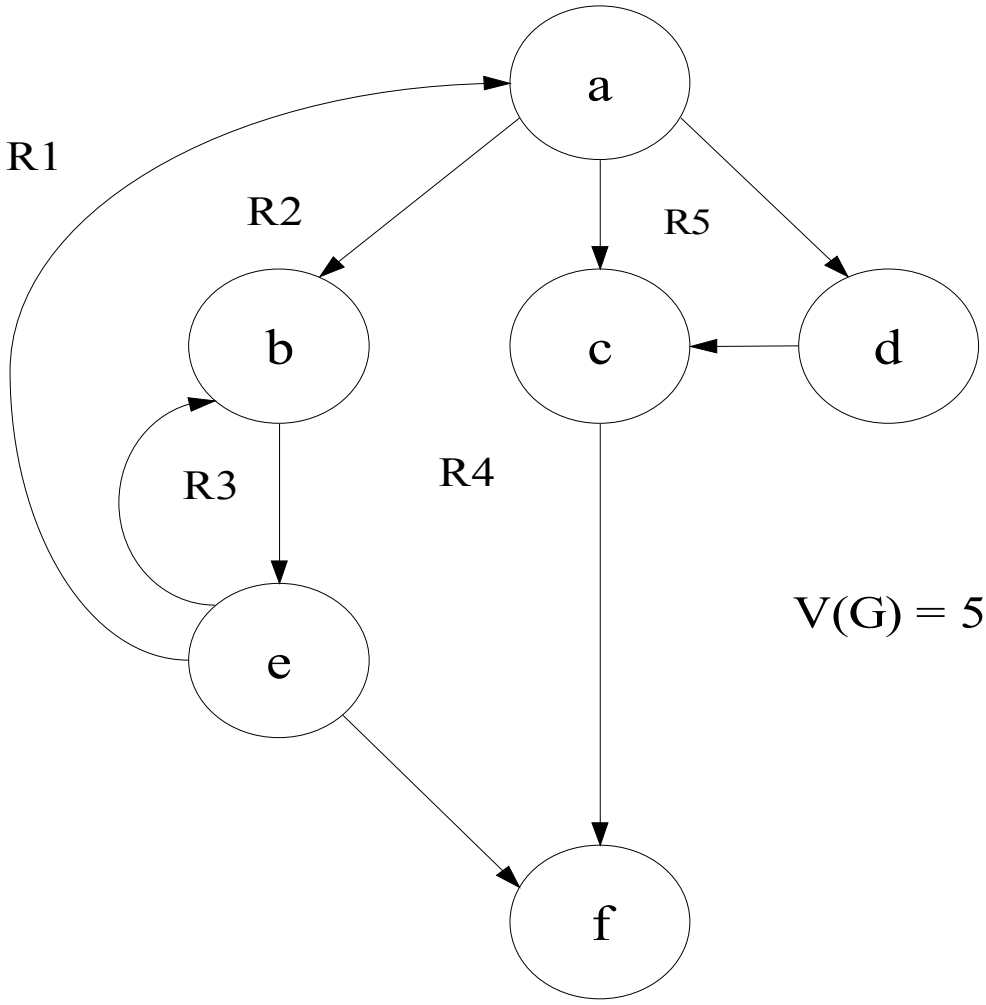
b) **Performance na extração de informações.** Para este item é utilizada a ferramenta Decision Cube do Borland Delphi. As mesmas consultas são aplicadas aos dois modelos e o tempo de resposta é utilizado como parâmetro de comparação.

c) **Redundância de dados.** O modelo de dados utilizado nos dois protótipos é comparado a fim de verificar a existência de redundância de dados.

d) **Complexidade do modelo de dados.** Este item é comparado qualitativamente. Os modelos de dados dos protótipos são analisados

a fim de se comparar a complexidade dos mesmos. Esta comparação servirá para se conhecer o nível de dificuldade oferecido na montagem de consultas SQL e a dificuldade de se entender o modelo.

Figura 19 - Complexidade Ciclomática



Fonte: PRESSMAN, R. Engenharia de Software. Rio de Janeiro: Makron Books, 1995. p. 761.

e) **Complexidade do código fonte da aplicação.** A complexidade do código fonte é comparada utilizando-se a métrica de complexidade ciclomática. Esta métrica foi proposta por McCABE (apud PRESSMAN, 1995 e KAN, 1995) e baseia-se no fluxo de controle de um programa (Figura 19). Um grafo de fluxo é desenhado para o programa representando o número de caminhos independentes que o fluxo do programa pode percorrer. Cada nó do gráfico representa uma tarefa de processamento, que pode ser composta por uma ou mais instruções fonte. Cada seta representa um caminho a ser tomado pelo fluxo do programa. A complexidade ciclomática (V(G)) é dada pelo número de

regiões do gráfico. No exemplo da Figura 19, V(G)=5, pois o gráfico tem cinco regiões.

f) **Tamanho do código fonte.** Os programas necessários ao funcionamento dos dois sistemas são comparados a fim de se verificar a complexidade dos mesmos. Para tanto é utilizada a métrica de Linhas de Código.

g) **Tamanho da base de dados.** O tamanho total da base de dados nos dois modelos.

h) **Complexidade das consultas SQL.** Serão utilizados dois critérios para medir a complexidade das consultas SQL. O primeiro é a quantidade de joins necessários ao processamento de carga do cubo de decisão. O segundo é a quantidade de joins entre dimensões necessários à carga do cubo. Os joins entre dimensões são aqueles que ligam uma dimensão a outra, comumente utilizados no modelo relacional e praticamente eliminados do esquema estrela, a fim de garantir performance na execução e simplicidade na construção das consultas.

4 O MODELO PROPOSTO

4.1 Introdução

O presente capítulo apresenta o modelo proposto. O modelo procura unir características de suporte ao processamento transacional, obtidas do modelo entidade-relacionamento às características do modelo dimensional, de modo que o modelo suporte tanto o processamento de transações, quanto aplicações de apoio informacional.

Este modelo tem a vantagem de eliminar os custos de duplicação dos ambientes de processamento transacional e informacional, através de um modelo em estrela sobreposto ao modelo entidade/relacionamento. No próximo capítulo são apresentados os resultados dos testes executados com o modelo proposto em relação ao modelo relacional e ao dimensional puro.

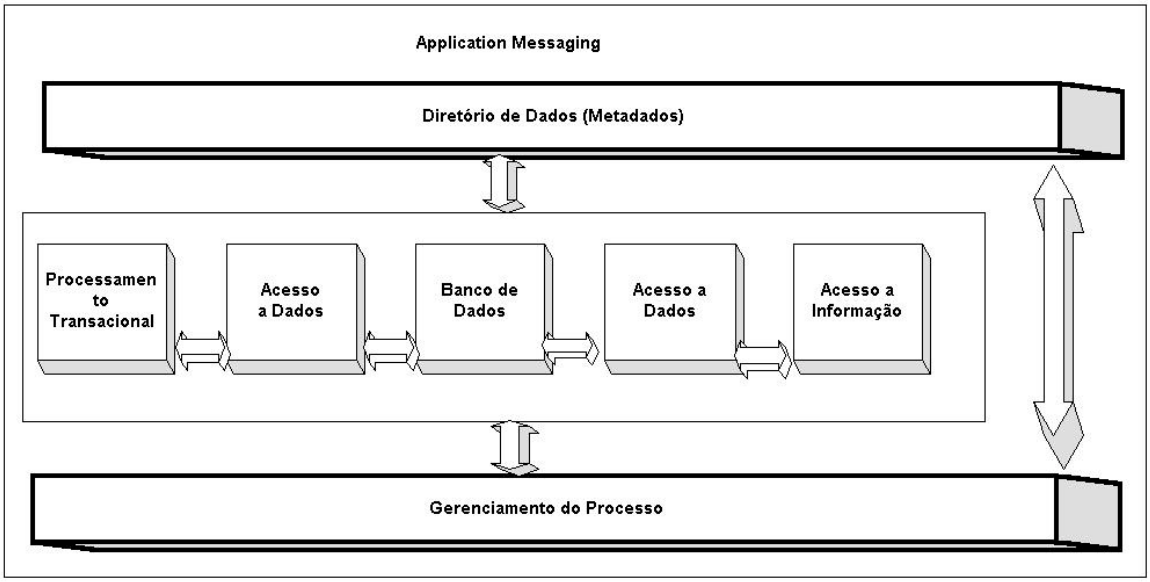
4.2. Descrição do Modelo

O modelo utiliza uma variação do modelo snowflake em cadeia (HARRISON, 1998). Este modelo normaliza as tabelas de dimensão, criando relacionamentos entre as mesmas. A normalização reduz o tamanho das dimensões e aumenta a cardinalidade dos relacionamentos entre dimensões e tabela de fatos. Entretanto, a recuperação de informações exige um aumento do número de junções entre tabelas, devido aos relacionamentos em cascata, o que acarreta uma queda na performance do processamento sistemas de processamento informacional. A queda de performance na extração de informações é resolvida conectando-se todas as tabelas de dimensão à tabela de fatos. O processo de modelagem é descrito nos próximos parágrafos.

A redução dos custos se dá através da não utilização de ambientes separados para processamento transacional e informacional , uma vez que ocorre a unificação das camadas de dados operacionais e de data warehouse e a eliminação das camadas de data staging e da parte da camada de acesso a dados que transporta os dados operacionais para a camada de data staging. Com base na arquitetura proposta por SINGH (2000) a Figura 20 a seguir propõe um modelo conceitual de para processamento transacional e informacional utilizando o modelo híbrido. O

modelo acrescenta uma camada de processamento transacional, a fim de alimentar uma base de dados única.

Figura 20 - Uma arquitetura em camadas utilizando o modelo híbrido



a) **Camada de processamento transacional.** Esta camada é responsável pelo processamento transacional da organização. É composta pelos sistemas de processamento transacional.

b) **Camada de acesso a dados.** Faz a conexão entre o banco de dados e as camadas de processamento de transações e de acesso a informação, é composta por SGBDs e todo o hardware/software necessários a serviços de acesso a bancos de dados

c) **Camada de Banco de Dados.** É o próprio banco de dados implementado com base no modelo híbrido, englobando em uma única base de dados o data warehouse e os dados dos sistemas transacionais.

d) **Camada de acesso a informação.** É a camada através do qual o usuário de sistemas informacionais tem acesso ao data warehouse.

e) **Camada de diretório de dados.** Contém os metadados técnicos e de negócios. Como não são necessárias informações a respeito do processo de data staging, esta camada é uma camada mais fina do que a camada equivalente no data warehouse tradicional.

f) **Camada de gerenciamento do processo.** Analogamente à arquitetura proposta por SINGH (2001), esta camada organiza os diversos processos que atuam tanto no processamento transacional, quanto no informacional.

g) **Camada application messaging.** Cumpre o mesmo papel que a camada equivalente na arquitetura de SINGH (2001), ou seja, é responsável pelo transporte de dados e informações pela rede da empresa.

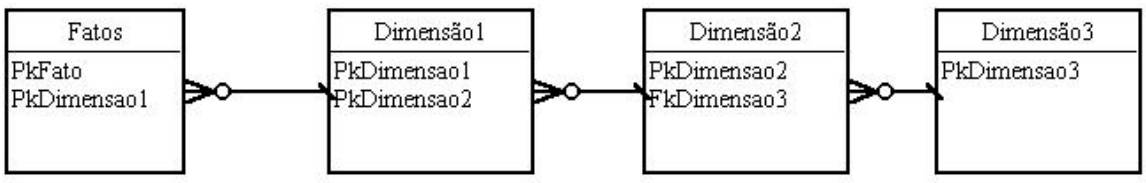
4.2. Estrutura de Custos do Modelo Híbrido

Comparando-se a estrutura de custos proposta por KIMBALL (1998b) com a arquitetura em camadas proposta para o modelo híbrido, constata-se que a eliminação da camada de data staging suprime não somente os custos de duplicação dos ambientes de informática (um para o processamento transacional e outro para o informacional), como também os custos da própria camada de data staging. Estes custos envolvem o hardware para esta camada, ferramentas de data staging, licenças de bancos de dados, custos de desenvolvimento, custos com treinamento do pessoal de desenvolvimento e custos com escalabilidade do ambieente.

4.3. Desenvolvimento do Modelo

O desenvolvimento de um modelo híbrido inicia-se a partir da elaboração de um modelo snowflake em cadeia (HARRISON, 1998), mantendo-se a granularidade das tabelas de dimensão e das tabelas de fato no nível atômico (Figura 21). As tabelas de dimensão mantêm as informações a respeito do contexto em que as transações/fatos ocorreram, enquanto que a tabela de fatos armazena todas as transações ocorridas no período. Um exemplo completo de snowflake em cadeia está ilustrado na Figura 23

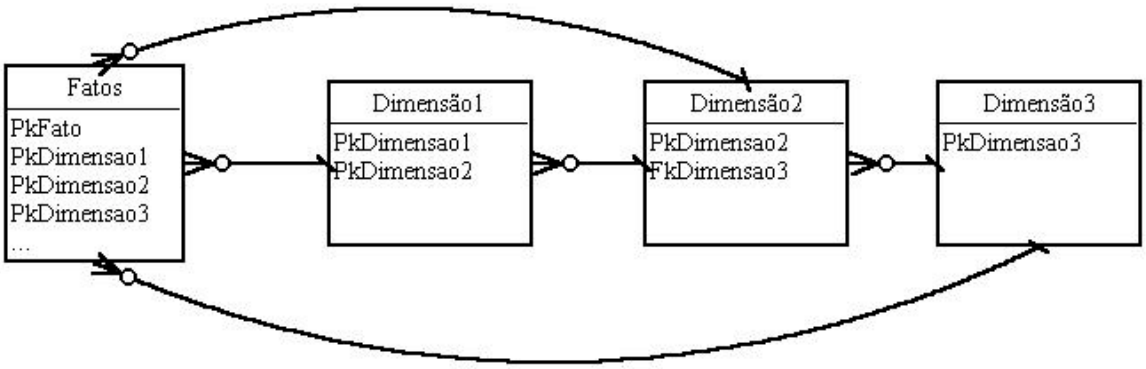
Figura 21 - Modelo snowflake em cadeia



A utilização do modelo snowflake em cadeia é importante, pois mantém as tabelas de dimensão totalmente normalizadas. A normalização das tabelas de dimensão garante a integridade referencial, fundamental para o processamento de transações. A manutenção granularidade das tabelas em nível atômico permite o armazenamento de cada transação ocorrida individualmente, permitindo a extração de informações em qualquer nível de detalhe, no caso de processamento informacional e garante um rastreamento detalhado de todas as transações no caso de processamento transacional.

O próximo passo é, exportar as chaves primárias das tabelas de dimensão que não se relacionam diretamente com a tabela de fatos para a mesma, gerando um aumento do número de relacionamentos e de chaves estrangeiras (Figura 22). Este procedimento transforma o modelo snowflake em cadeia em um modelo híbrido, capaz de suportar tanto o processamento transacional quanto informacional. A exportação de chaves estrangeiras reduz a quantidade de joins em cascata exigida pelo modelo snowflake em cadeia quando se processa a extração de informações. Esta alteração faz com que se obtenha um modelo em estrela baseado em minidimensões (KIMBALL, 1998a) sobreposto ao modelo snowflake, proporcionando duas visões diferentes do mesmo modelo de dados, uma para o processamento transacional e outra para o processamento informacional. Um exemplo de modelo híbrido de dados está exposto na Figura 24.

Figura 22 - Modelo híbrido proposto



A exemplo do modelo dimensional, para as chaves primárias, tanto das tabelas dimensionais quanto da tabela de fatos devem ser escolhidas chaves artificiais As chaves artificiais têm a vantagem de serem absolutamente estáveis ao longo do

tempo, facilitando assim quaisquer alterações em chaves candidatas, suportando o processamento transacional e mantendo intacto o histórico dos fatos ocorridos ao longo do tempo. As características contidas no modelo, capazes de suportar tanto o processamento transacional quanto informacional estão descritas nas seções seguintes.

4.5. Suporte ao processamento transacional e ao data warehouse.

O modelo proposto deve preservar as características de um data warehouse e ainda suportar o processamento transacional. A seguir estas características serão discutidas sob os aspectos do data warehouse propostas por INMON (1997) e ao mesmo tempo dos sistemas de processamento transacional.

4.5.1. Baseado em assuntos

O modelo proposto suporta perfeitamente esta característica, pois as tabelas de dimensão e de fatos devem ser projetadas para conter tanto informações necessárias ao processamento transacional quanto informações a respeito do negócio necessárias ao apoio à decisão.

4.5.2. Integrado e não volátil

O modelo adapta-se perfeitamente à arquitetura BUS (KIMBALL, 1998a, 1998b), que permite a construção de data marts integrados. Dimensões devem ser projetadas de forma que sejam compartilhadas por diversos data marts e as tabelas de fato devem ser projetadas de forma a eliminar qualquer inconsistência, de forma que o sistema forneça respostas únicas para uma determinada consulta, obtendo-se assim um data warehouse lógico e incremental planejado de forma global e construído e implantado através de liberações sucessivas.

Como o modelo deve suportar o processamento transacional e o de apoio à decisão, é necessário um tratamento específico no que diz respeito à volatilidade dos dados. Alterações ou exclusões físicas em registros das tabelas de fato e de dimensões não devem ser permitidas, de forma a preservar o histórico dos fatos ocorridos. A solução para este problema é a utilização de baixas e alterações lógicas

através da adição de registros de alteração e de baixa, daí a necessidade de se utilizar chaves artificiais como chaves primárias.

Figura 23- Um modelo snowflake em cadeia

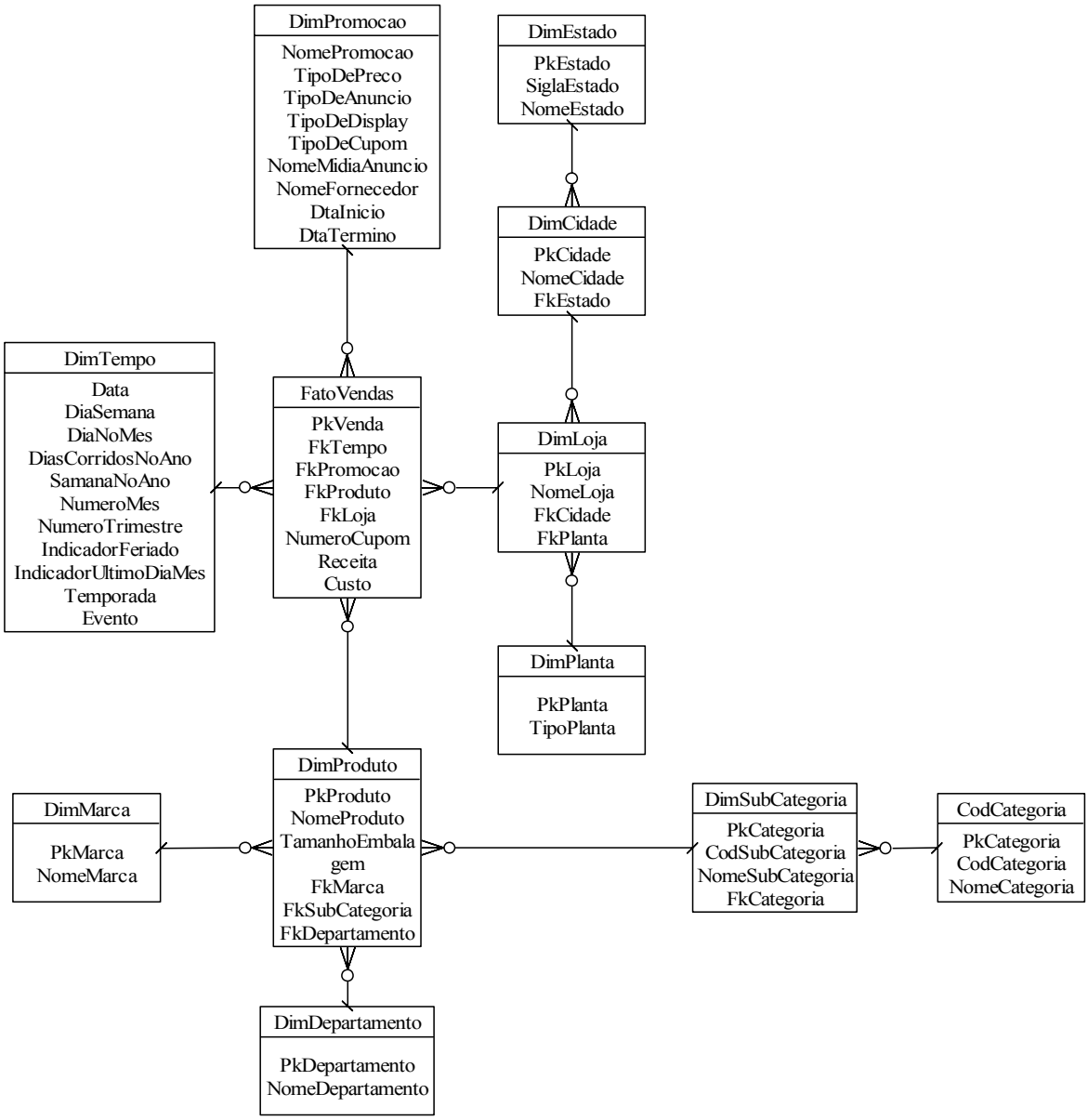
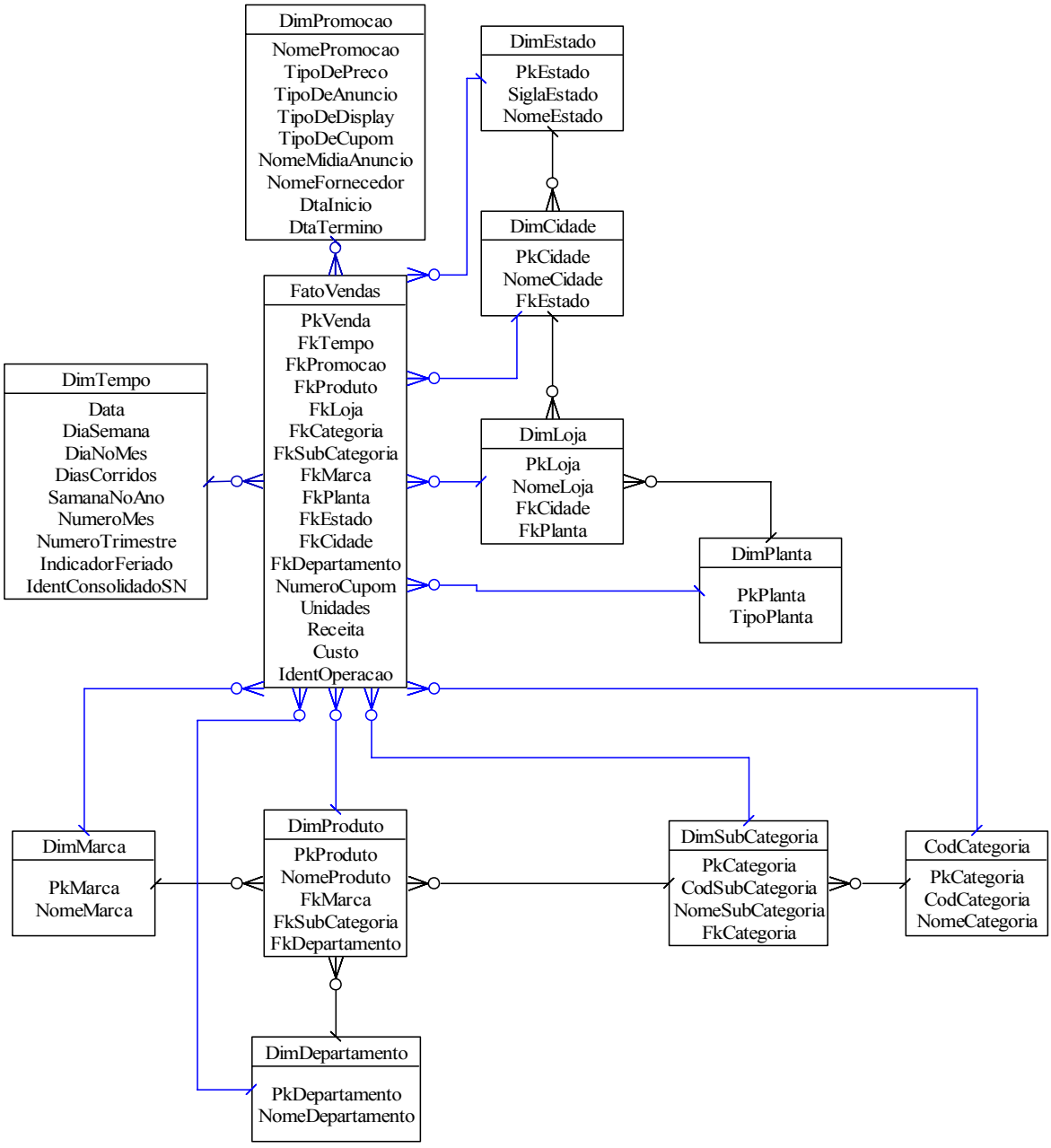


Figura 24 - O modelo proposto



.

As exclusões nas tabelas de dimensão devem ser processadas através de baixas lógicas. Para tanto se utiliza o atributo data de baixa (DtaBaixa) em todas as tabelas. Para se baixar um registro de uma tabela de dimensão, basta gravar a data de baixa no atributo apropriado. Desta forma, pode-se preparar o processamento transacional para não permitir a utilização de um registro já baixado (DtaBaixa <> NULL) e ao mesmo tempo preservar a informação histórica através da não eliminação deste registro. Da mesma forma, uma alteração de um registro de dimensão deve ser feita

mediante uma baixa lógica seguida da inclusão de um novo registro. A Figura 25 a seguir ilustra o processo de alteração da renda mensal de um cliente.

A alteração de um registro de uma tabela de fatos segue o mesmo princípio. Entretanto, como um fato contém atributos de medição do negócio, uma baixa ou alteração destes atributos de medição resulta em alteração nos resultados do negócio. Para que o histórico das alterações seja preservado, além da data de baixa, utiliza-se um atributo para identificar a operação que originou o registro (IdentOperacao) em conjunto com o atributo data de baixa (DtaBaixa). Este novo atributo deve conter os seguintes valores:

- **Novo**: para a inclusão de um novo registro.
- **Baixa**: para gravação de um registro de baixa, anulando os valores do registro original.
- **Alteração**: para gravação de um registro de alteração.

Figura 25 - Alteração da renda mensal de um cliente

PkCliente	CPFCliente	NomeCliente	Renda	DtaBaixa	
1	999999999-99	José da Silva	R$ 2.500,00		Antes da alteração

PkCliente	CPFCliente	NomeCliente	Renda	DtaBaixa	
1	999999999-99	José da Silva	R$ 2.500,00	23/01/02	Registro baixado
2	999999999-99	José da Silva	R$ 3.000,00		Registro Alterado

Desta forma, ao se baixar um registro, grava-se a data de baixa no registro original e inclui-se um registro de baixa, com seus atributos de medição contendo valores negativos, de forma a anular os valores do registro original. Em caso de alteração, procede-se como na baixa, mas gravando-se também um registro de alteração com os novos valores. A Figura 26 a seguir mostra a alteração do atributo Receita um registro de venda de R$ 70,00 para R$ 75,00. Nela observa-se que uma instrução SQL select que totalize as colunas os atributos de medição, (Receita e Custo), agrupados pelo atributo Cupom retornará como resultado o valor após a alteração (Receita = 70,00 – 70,00 + 75,00 = 75,00 e Custo = 100,00 – 100,00 + 100,00 = 100,00).

Figura 26 - Alteração de um registro de venda

PkVenda	Cupom	...	Receita	Custo	DtaBaixa	IdentOperacao	Antes da alteração
1	123		R$ 100,00	R$ 70,00		NOVO	

PkVenda	Cupom	...	Receita	Custo	DtaBaixa	IdentOperacao	Alteração do custo para R$ 75,00
1	123	...	R$ 100,00	R$ 70,00	23/01/02	NOVO	
2	123	...	R$ (100,00)	R$ (70,00)	23/01/02	BAIXA	
3	123	...	R$ 100,00	R$ 75,00		ALTERACAO	

Uma forma alternativa de controlar as baixas e alterações é utilizar um atributo indicando se o registro está baixado ou não (IdnBaixadoSN = S, para registros baixados ou IdnBaixadoSN = N para registros ativos), em substituição a data de baixa. Esta opção tem a vantagem de não utilizar atributos com valores nulos, mas traz a desvantagem de não documentar a data em que o registro foi baixado ou alterado.

Para a aplicação de data warehouse, é importante que os dados estejam consolidados para serem consultados, isto é obtido, carregando-se somente dados consolidados na base de dados do data warehouse. Como o modelo híbrido deve suportar processamento transacional, não havendo, portanto, um processo periódico de carga, utiliza-se um atributo na dimensão tempo para informar se aquela data está consolidada (IdentConsolidadoSN). Este atributo indica se a data poderá ou não receber novos fatos. Este atributo pode assumir os seguintes valores:

- **NÃO**: Indica que os fatos relacionados a este registro de tempo ainda não estão consolidados, portanto, as consultas a este período podem sofrer alterações.

- **SIM**: Indica que os fatos relacionados a este registro de tempo estão consolidados e que os sistemas de processamento transacional não podem mais atualizar informações neste período.

É importante salientar que, diferentemente do data warehouse tradicional, onde os dados precisam passar por um processo de carga antes de serem consultados. A consulta é permitida em qualquer data a qualquer momento, independentemente da data estar consolidada ou não. A diferença é que, se uma determinada data não está consolidada, novos registros ainda poderão ser adicionados a esta data, o que acarretará mudanças nos valores obtidos em consultas posteriores.

4.5.3. Variável em relação ao tempo

Como o modelo permite o acúmulo de dados ao longo de diversos períodos de tempo, não sendo alterados após sua consolidação, o modelo pode fornecer informações de apoio à decisão em diversos períodos de tempo de forma precisa.

4.5.4 Fornecer dados de apoio às decisões gerenciais

Para KIMBALL (1998a) o modelo dimensional é composto por uma tabela de fatos central, conectada a diversas tabelas de dimensão. Este modelo tem a capacidade de fornecer, informações no formato de um cubo de dados.

O modelo proposto é formado por uma tabela de fatos central, conectada a tabelas dimensionais normalizadas. O modelo proposto também conecta todas as dimensões à tabela de fato e também dimensões entre si devido à normalização das mesmas. A Figura 24 mostra o modelo híbrido com os relacionamentos entre as tabelas de dimensão e a tabela de fatos destacados, para ilustrar sua semelhança com o esquema estrela e demonstrar a capacidade do mesmo de fornecer informações na forma de um cubo de dados.

Como as tabelas de fato são mantidas no menor nível de granularidade possível, as tabelas de fato podem ficar demasiadamente grandes. Para se resolver este problema, agregados podem ser utilizados de forma a reduzir a quantidade de registros lidos em uma consulta.

4.6. Considerações finais a respeito do modelo proposto

O modelo proposto une características de suporte ao processamento transacional à características de aplicações de data warehouse. O quadro 02 a seguir resume as principais características conceituais do modelo proposto em relação ao modelo dimensional e ao modelo relacional.

A redução de custos no modelo proposto se dá através da eliminação da camada de data staging e pela unificação dos ambientes de processamento transacional e informacional.

Quadro 2 - Características conceituais do modelo proposto em relação ao dimensional e ao relacional

Característica	Proposto	Dimensional	Relacional	Snowflake em cadeia
Granularidade	Mantida em nível atômico para suportar o processamento transacional. Esta característica vem da arquitetura BUS e do modelo relacional	Depende das necessidades da aplicação. A arquitetura BUS propõe que a granularidade deve ser mantida no menor nível possível.	Mantida em nível atômico.	Depende das necessidades da aplicação
Normalização das Dimensões	As dimensões devem ser normalizadas, conforme o modelo snowflake em cadeia e o modelo relacional	Dimensões não normalizadas para melhorar a performance e simplificar modelo	Todas as entidades são normalizadas	Dimensões devem ser normalizadas
Relacionamentos	Entre as dimensões normalizadas, conforme o modelo snowflake em cadeia e das dimensões para a tabela de fatos, a fim de garantir performance no processamento transacional	Das dimensões para a tabela de fatos e, opcionalmente, de minidimensões para dimensões maiores	De acordo com as regras da normalização	Entre as dimensões normalizadase entre a tabela dimensional principal e a tabela de fatos.

5 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

5.1 Introdução

Neste capítulo são apresentados e analisados os resultados dos testes realizados com o protótipo construído segundo o modelo proposto, em relação ao modelo relacional e ao dimensional puro.

Os testes, conforme a metodologia proposta consideram os seguintes aspectos:

- performance no processamento transacional;
- performance na extração de informações;
- redundância de dados;
- complexidade do modelo de dados;
- complexidade do código-fonte da aplicação;
- tamanho do código-fonte;
- tamanho da base de dados.

Os testes foram realizados em um notebook Toshiba Satellite® 2180CDT com as seguintes especificações técnicas:

- processador AMD K6-II 475 Mhz, cache L2 de 512 KB;
- Memória 64 MB PC100 SDRAM
- Disco rígido de 4,2 GB enhanced IDE

5.2 Descrição dos protótipos

Foram construídos três protótipos de um sistema de vendas fictício, o primeiro utilizando o modelo proposto (Figura 27) e os outros dois utilizando o modelo relacional (Figura 28) e o modelo dimensional (Figura 29), respectivamente. Como em um sistema de vendas, normalmente os cupons de venda são numerados seqüencialmente, optou-se por utilizar uma chave artificial para a tabela de vendas

nos três modelos. O ambiente de programação utilizado para a construção dos protótipos foi o Delphi e o banco de dados, o Interbase 6.0. Para a conexão entre o banco de dados e a aplicação foi utilizado o Borland Database Engine (BDE).

A base de dados utilizada para carga dos protótipos contém dados a respeito das vendas de um supermercado fictício, cuja tabela de fatos contém 11063 registros, e foi extraída de KIMBALL (1998a). Uma vez que as queries que carregam os cubos de decisão nos três protótipos fazem uma varredura total da tabela de fatos e associam as tabelas de dimensão pelas chaves primárias, não foram criados índices secundários em nenhuma das tabelas dos três protótipos. Os scripts de criação da base de dados e as queries para carga do cubo de decisão estão contidos nos apêndices.

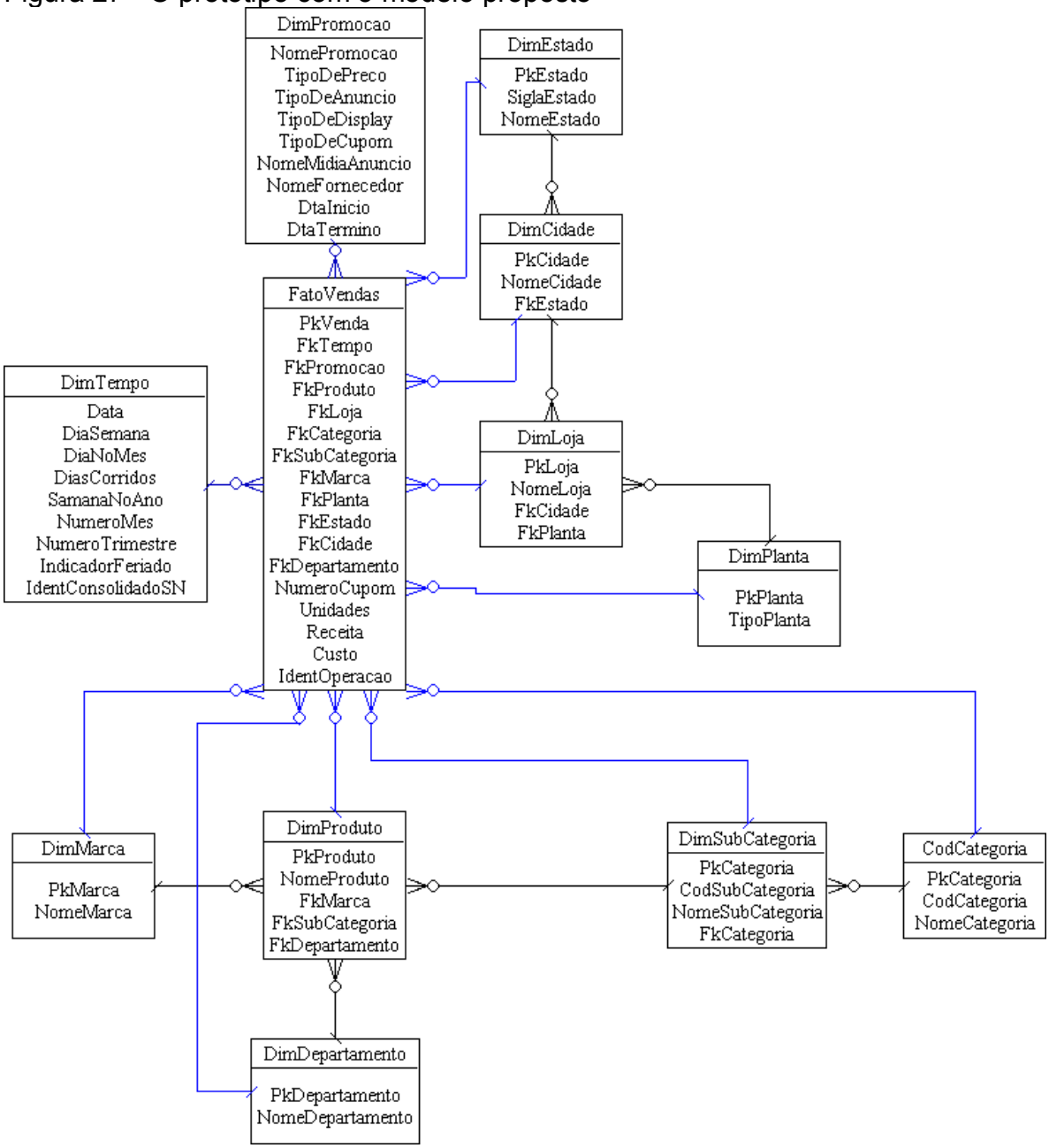
5.3 Performance no processamento transacional

Para a medição da performance no processamento transacional são carregados 11040 registros na tabela de fatos a respeito de vendas (tblFatoVendas) do modelo proposto e na tabela vendas (tblVendas) do modelo relacional.

O modelo relacional levou 76 segundos para carregar, enquanto que o modelo proposto levou 474 segundos para carregar a base de dados. O tempo de gravação por registro foi de 0,00688 segundo por registro no modelo relacional e de 0,04293 segundo por registro no modelo proposto. Esta diferença de tempo deve-se ao fato de, no modelo proposto, existir um processo de busca das chaves estrangeiras para gravação na tabela de fatos, inexistente no modelo relacional.

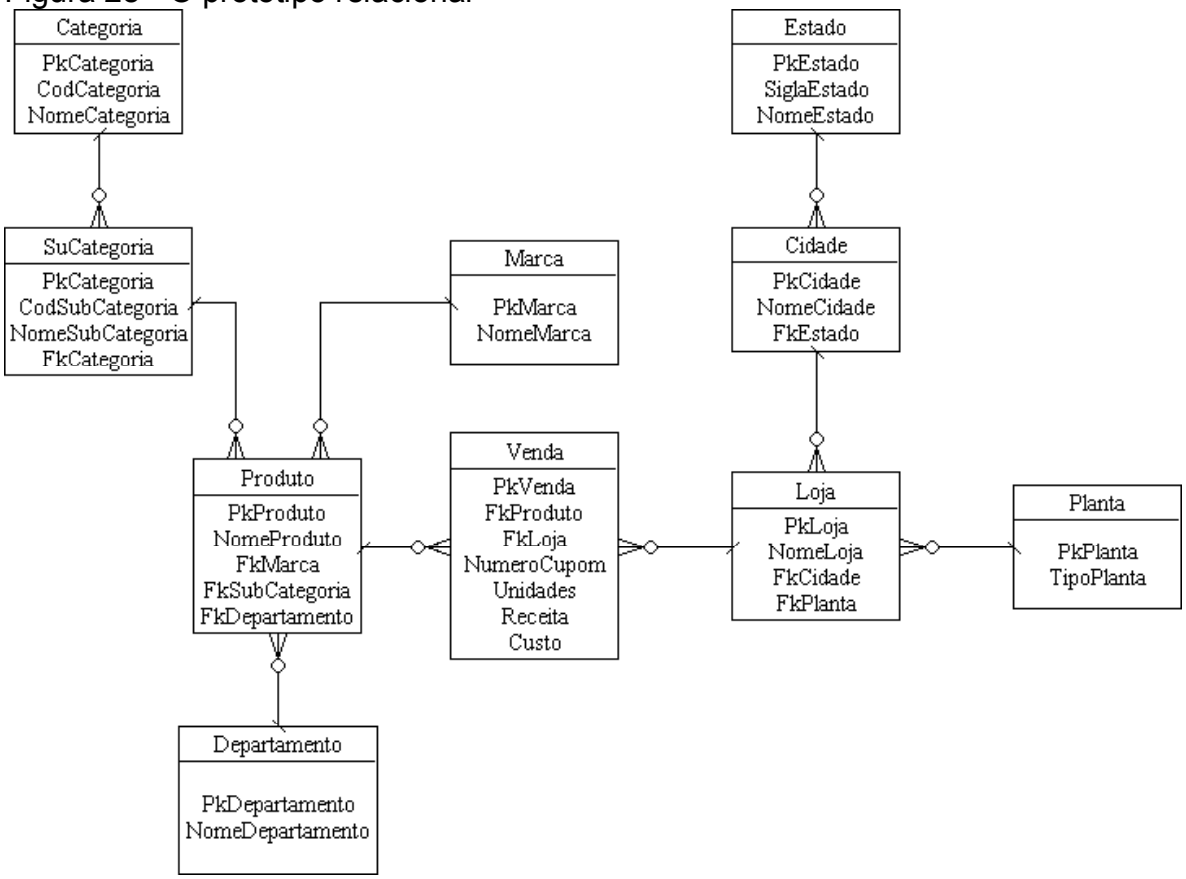
A diferença de tempo constatada (o modelo proposto foi 523 % mais lento no processo de carga que o modelo relacional), apesar de significativa se for considerado o tempo total, é insignificante, se for considerado o tempo de gravação por registro em relação ao tempo que o usuário leva para digitar os dados referentes a um registro. Deve-se levar em conta também que não foram implementadas regras de negócio comumente utilizadas em sistemas de venda, tais como cálculo de tributos, atualização de níveis de estoque e outras, que, uma vez implantadas nos dois protótipos reduziriam a diferença relativa de performance entre os mesmos.

Figura 27 - O protótipo com o modelo proposto



Fonte: Adaptado de KIMBALL, R. Data Warehouse Toolkit. São Paulo: Makron Books, 1998.

Figura 28 - O protótipo relacional

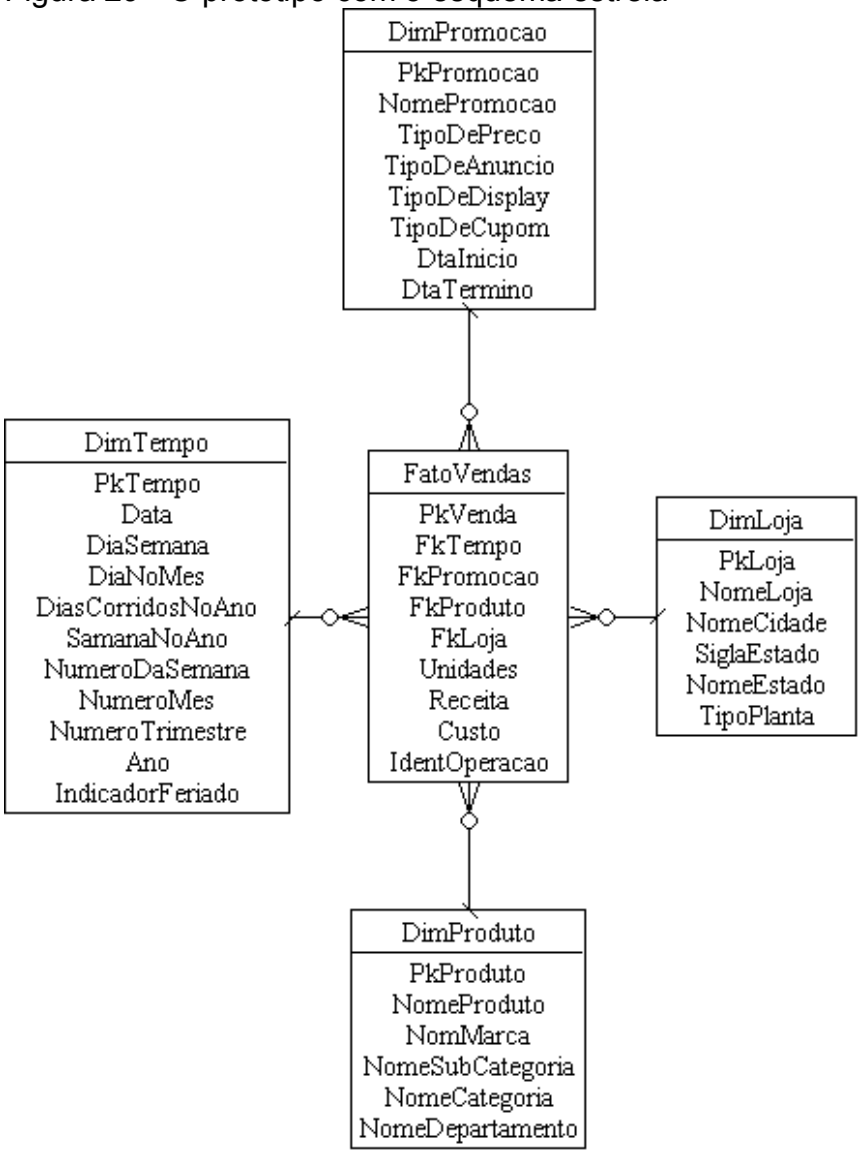


5.4 Performance na extração de informações

Neste item o modelo proposto é comparado com o modelo relacional e com o modelo dimensional puro. É criada uma consulta SQL para montagem de um cubo de decisão que retorna o mesmo conjunto de dados para os três modelos e são contados os tempos necessários à execução da consulta. Como no modelo relacional não existem as dimensões tempo e promoção, estas não também não foram utilizadas no modelo híbrido e dimensional.

O protótipo construído com o modelo proposto levou 18 segundos para retornar os resultados, o protótipo dimensional levou 10 segundos, enquanto que o relacional levou 66 segundos para retornar os resultados. O modelo foi 247% mais rápido do que o modelo relacional e 80% mais lento do que o modelo dimensional, ficando o modelo proposto mais lento que o modelo dimensional, mas muito mais rápido que o modelo relacional.

Figura 29 - O protótipo com o esquema estrela



5.5 Redundância de dados

Neste item analisa-se a redundância de dados imposta pelo modelo. Como este aspecto é importante apenas do ponto de vista do processamento transacional, o modelo proposto é analisado apenas em relação ao modelo relacional.

Analisando-se o modelo relacional, constata-se que os únicos dados que redundantes são as chaves primárias, que são exportadas para as tabelas relacionadas na forma de chaves estrangeiras a fim de impor a integridade referencial. No modelo proposto, os únicos dados redundantes também são as chaves primárias, que são exportadas para as tabelas relacionadas na forma de chaves estrangeiras. O que ocorre em relação ao modelo relacional é um aumento do número de chaves estrangeiras, devido ao aumento de relacionamentos imposta pelo modelo. Portanto, o nível de redundância nos dois modelos se equivale.

Como só se utilizam chaves artificiais no modelo proposto e estas chaves são absolutamente estáveis ao longo do tempo, pode-se afirmar que o aumento do número de chaves estrangeiras não acarreta nenhum problema de inconsistência da base de dados devido ao aumento do número de chaves estrangeiras.

5.6 Complexidade do código-fonte da aplicação

A métrica utilizada para análise da complexidade do código-fonte da aplicação é a de complexidade ciclomática, proposta por MacCabe (PRESSMAN, 1995, KAN, 1995), e baseia-se no fluxo de controle da aplicação (para maiores detalhes a respeito da métrica de complexidade ciclomática, consultar o capítulo 3, seção 3.2).

Foram comparadas as complexidades do código necessárias ao modelo entidade/relacionamento e ao modelo proposto. A única diferença de codificação entre os dois modelos está no processo de manutenção da base de dados, portanto foi analisado somente este processo. Apesar do código do modelo proposto ser mais extenso, ambos apresentaram um grau de complexidade igual a sete, pois o fluxo de controle do programa é exatamente igual nos dois modelos. Pode-se concluir, portanto, que a complexidade do código fonte não varia do modelo relacional para o dimensional.

5.7 Tamanho do Código-Fonte

Na medição do tamanho do código-fonte utilizou-se a contagem de linhas de código, sendo comparados o modelo relacional ao proposto. O modelo relacional totalizou 342 linhas de código, enquanto que o modelo proposto, totalizou 547 linhas de código. O modelo proposto necessitou de um código 62% mais extenso do que o

modelo relacional, nos protótipos testados. Esta diferença deve-se aos seguintes processos adicionais existentes no modelo proposto:

- O processo de alteração e de baixa de registros é implementado de forma lógica e não física, o que obriga a existência de processos de adição de registros de baixa, no processo de baixa de registros, neutralizando um determinado fato, e de registros de baixa e de alteração, no caso de alterações em registros, gravando assim, um registro que neutraliza o fato e outro que altera o mesmo. Apesar de mais complexo, este procedimento tem a vantagem de preservar o histórico dos fatos, uma vez que não há a eliminação ou exclusão física de registros, mas somente um processo de adição contínua. Caso este procedimento seja implantado no modelo relacional, o mesmo código será necessário.

- O processo de busca de chaves estrangeiras é implementado para buscar as chaves de todas as dimensões existentes, uma vez que elas são diretamente conectadas à tabela de fatos.

- Nos dois modelos não foram codificadas regras de negócio comuns em um sistema de vendas real, foram codificadas somente as instruções necessárias à carga das bases de dados. Uma vez codificadas estas regras de negócio, a diferença relativa entre os dois modelos se reduzirá significativamente.

5.8 Tamanho da base de dados

O protótipo que utiliza o modelo dimensional puro ocupou um total de 1870 KB, o modelo relacional ocupou um total de 1552 KB e o modelo híbrido ocupou 3172 KB. O modelo híbrido, portanto, tende a ocupar mais espaço do que os demais modelos individualmente: 48% mais que o modelo relacional, 58% mais que o modelo dimensional. Entretanto, o modelo proposto ocupou 7% menos espaço do que a soma das bases de dados do modelo relacional e do dimensional.

5.9 Complexidade das consultas SQL

Para carregar o cubo de decisão utilizando o modelo dimensional, foram necessários dois joins diretos com a tabela de fatos, o mesmo cubo de decisão necessitou de oito joins diretos com a tabela de fatos no modelo proposto. No modelo relacional foram necessários quatro joins diretos com a tabela de fatos, mais quatro joins entre dimensões.

Contando-se apenas o número de joins, os modelo proposto e o modelo relacional se equiparam, entretanto, no modelo relacional foram necessários joins em cadeia, enquanto que no modelo híbrido foram necessários apenas joins diretos com a tabela de fatos, o que simplifica a construção das consultas.

A partir das exposições acima se pode concluir que o modelo dimensional é mais simples do que os outros dois modelos, enquanto que o relacional é o mais complexo, ficando o modelo proposto em uma posição intermediária.

5.10 Considerações finais a respeito dos resultados obtidos

Este capítulo apresentou e analisou os resultados obtidos com os testes dos protótipos construídos. Levando-se em consideração os protótipos construídos, pode-se afirmar que o modelo proposto revelou-se viável na situação testada. O Quadro 3 a seguir, resume os resultados dos testes.

A performance do modelo proposto no processamento transacional ficou aquém da performance obtida com o modelo entidade/relacionamento. Esta diferença deve-se ao processo de busca de chaves estrangeiras para gravação na tabela de fatos. Entretanto, levando-se em conta o tempo necessário ao usuário digitar os dados da transação e o tempo necessário para a gravação de um registro, pode-se afirmar que esta diferença não é relevante. No que diz respeito à performance na extração de informações, o modelo foi mais lento que o modelo dimensional, mas muito mais rápido que o modelo relacional, podendo-se afirmar que o modelo proposto se adequou mais a aplicações de data warehouse do que o modelo relacional.

A complexidade do modelo aumentou em relação ao modelo dimensional puro, mas como o mesmo apresenta uma visão em estrela da base de dados, ele é mais simples que o modelo relacional na construção de cubos de decisão.

Quadro 3 - Resultado da avaliação dos protótipos

Item	Modelo Proposto	Relacional	Dimensional	Diferença relativa
Performance no processamento transacional	0,04293 segundo/registro	0,00688 segundo/registro	Não aplicado	523% mais lento que o modelo relacional
Performance na extração de informações	18 segundos	66 segundos	18 segundos	247% mais rápido que o modelo relacional 80% mais lento que o modelo dimensional
Redundância de dados	Somente chaves estrangeiras	Somente chaves estrangeiras	Não aplicado	Não aplicado
Complexidade do modelo de dados	Complexidade média para o usuário final	Complexo para o usuário final	Simples para o usuário final	Não aplicado
Complexidade ciclomática do código-fonte da aplicação	7	7	Não aplicado	0%
Tamanho do código-fonte	545 LOC	342 LOC	Não aplicado	62% maior que o modelo relacional
Tamanho da base de dados	3172 KB	1552	1870	7% menor que o espaço ocupado pelo modelo relacional e pelo dimensional juntos
Complexidade das consultas SQL	Média	Baixa	Alta	Não aplicado

O código-fonte do modelo proposto apresentou a mesma complexidade do modelo relacional, mas foram necessárias mais linhas de código para implementa-lo. Esta diferença se deve ao processo de exclusão e alteração lógica de registros e ao processo de busca das chaves estrangeiras para ligação com a tabela de fatos. Deve-se levar em conta que, se o mesmo processo de baixa e alteração lógica de registro for implementado em um modelo relacional, a diferença na quantidade de linhas de código entre os dois modelos cairá significativamente.

Finalmente, deve-se levar em conta que não foram codificadas regras de negócio, comuns em sistemas reais, em nenhum dos protótipos implementados. A

codificação destas regras acarretará queda na performance do processamento transacional e aumento do tamanho do código fonte, tanto no modelo relacional quanto no modelo proposto, o que reduzirá a diferença relativa de performance no processamento transacional e no tamanho do código fonte em ambos modelos.

6. CONCLUSÃO E ESTUDOS FUTUROS

6.1 Conclusão

A globalização gerou um ambiente onde a informação é um dos principais insumos para que as organizações possam reagir aos problemas e oportunidades que surgem no horizonte. No ambiente globalizado, utilização eficaz da tecnologia da informação pode fazer com que as organizações adquiram supremacia sobre a concorrência no mercado em que atuam.

É neste contexto, que este trabalho propõe um modelo de dados viável nas condições testadas, capaz de suportar tanto o processamento transacional quanto o processamento informacional. O modelo proposto não se destina a substituir sistemas informacionais suportados por data warehouses com terabytes de informações. Este modelo destina-se a suprir de informações aquelas empresas que não possuem recursos para investir nas tecnologias necessárias a construção de data warehouses em separado para suporte aos seus sistemas de apoio à decisão.

O modelo proposto revelou-se viável para a situação testada, produzindo resultados satisfatórios tanto para o processamento transacional quanto para o processamento informacional. A redução de custos foi conseguida unificando-se os dois ambientes de dados (transacional e informacional) e, eliminando-se a camada de data-staging e todos os custos agregados a esta camada. Deve-se considerar que a camada de diretório de dados (metadados) fica mais fina no modelo proposto, uma vez que o ambiente de banco de dados é único e não são necessários metadados a respeito do rastreamento dos dados desde sua origem nos sistemas de processamento transacional, passando pela camada de data staging até o data warehouse.

O capítulo dois faz uma revisão dos conceitos de sistemas de informação, classificando-os de acordo com sua finalidade. Em seguida é discutido o modelo entidade/relacionamento como modelo de dados para suporte ao processamento de transações. Finalmente são discutidos o modelo dimensional e o data warehouse como tecnologias fundamentais para os sistemas de apoio à decisão.

O capítulo três descreve a metodologia utilizada para a validação do modelo proposto, caracterizando a pesquisa, descrevendo a metodologia e enumerando os itens a serem analisados a fim de verificar a viabilidade do modelo proposto.

O capítulo quatro descreve as características do modelo proposto, comparando-o com o modelo dimensional e o modelo relacional. O capítulo cinco apresenta e descreve os resultados obtidos nos testes com os protótipos construídos para o estudo de viabilidade do modelo.

6.2 Estudos futuros

Face ao risco de se aplicar um modelo de dados ainda não testado em ambientes reais, optou-se por construir protótipos de laboratório para estudar a viabilidade do modelo proposto. Na seqüência deste trabalho, pretende-se aplicar o modelo em um sistema mais complexo e em ambiente real a fim de garantir de forma mais segura a viabilidade do modelo. Pretende-se também estudar formas de melhorar a performance do modelo no processamento transacional.

7 FONTES BIBLIOGRÁFICAS

BALLARD, C. et al. **Data Modeling Techniques for Data Warehousing.** IBM Corporation. Disponível em: <http://www.redbooks.ibm.com> Acesso em 21/06/2001.

BOAR, B. **Understanding Data Warehouse Strategically.** Disponível em <http://warehouse.chime-net.org/manage/stplan/bboar1.htm>. Acesso em 01/07/2001.

COUGO, P. **Modelagem Conceitual e Projetos de Bancos de Dados.** Rio de Janeiro. Campus, 1997.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados.** 7ª ed. Rio de Janeiro: Campus, 2000.

DOMENICO, J. A. **Definição de um Ambiente Data Warehouse em uma Instituição de Ensino Superior.** Florianópolis, 2001. 137 f. Dissertação (Mestrado em Engenharia de Produção) – Departamento de Engenharia de Produção, Universidade Federal de Santa Catarina.

DWBRASIL. **Data Mart.** Disponível em <http://www.dwbrasil.com.br/html/dmart.html>. Acesso em 01/07/2001.

FALSARELLA, O. M. e CHAVES, E. O. C. **Sistemas de Informação e Sistemas de Apoio à Decisão.** Disponível em: <http://chaves.com.br/TEXTSELF/COMPUT/sad.htm> Acesso em 21/06/2001.

FIRESTONE, J. M. **Architectural Evolution in Data Warehousing and Distributed Knowledge Management Architecture.** Disponível em: <http://www.dkms.com/ARCHEV.html>. Acesso em 20/08/2001.

FREITAS, H. e POZZEBOM, M. **Características Desejáveis de um EIS – Enterprise Information System – Rumo a Proatividade**. PPGA – Universidade Federal do Rio Grande do Sul. Disponível em: <http://read.adm.ufrgs.br/read05.artigo.eis.htm> Acesso em: 10/11/2000.

FURLAN, J. D.; IVO, I. M. e AMARAL, F. P. **Sistemas de Informação Executiva – EIS.** São Paulo: Makron Books, 1994.

GUPTA, V. R. **An Introduction to Data Warehousing**. Disponível em: <http://system-services.com/dwintro.asp>. Acesso em 01/07/2001.

HARRISON, Thomas. **Intranet Data Warehouse.** São Paulo: Berkeley, 1998.

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

KAN, S. H. K. **Metrics and Models in Software Quality Engineering.** Massachusetts: Addison-Wesley, 1998.

KIMBALL, R. **Data Warehouse Toolkit.** Makron Books: São Paulo, 1998a.

KIMBALL, R. **The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing Data Warehouses.** New York: Wiley Computer Publishing, 1998b.

KROENKE, D. M. **Banco de Dados: Fundamentos, Projeto e Implementação – 6ª. Edição.** Rio de Janeiro: Livros Técnicos e Científicos, 1999.

LAUDON, K. C. e LAUDON, J. P. **Sistemas de Informação.** Rio de Janeiro: Livros Técnicos e Científicos: 1998.

MACHADO, F. N. R. e ABREU, M. P. **Projeto de Banco de Dados:** Uma Visão Prática. São Paulo: Érica, 1996.

MACHADO, F. N. R. **Projeto de Data Warehouse:** Uma Visão Multidimensional. São Paulo: Érica, 2000.

PRESSMAN. R. S. **Engenharia de Software.** São Paulo: Makron Books, 1995.

SELL, D. **Uma Arquitetura para Distribuição de Componentes Tecnológicos de Sistemas de Informações Baseados em Data Warehouse.** Florianópolis, 2001. 89 f. Dissertação (Mestrado em Engenharia de Produção) – Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina.

SINGH, H. S. **Data Warehouse.** Conceitos, Tecnologias, Implementação e Gerenciamento. São Paulo. Makron Books, 2001.

SPRAGUE, R. H. JR. e WATSON, H. J. **Sistema de apoio à decisão:** colocando a teoria em prática. Rio de Janeiro: Campus, 1991.

STAIR, R. M. **Princípios de sistemas de informação, uma abordagem gerencial.** Rio de Janeiro: Livros Técnicos e Científicos, 1998.

VASCONCELLOS, J. M. Implementando um Data Warehouse Incremental. **DEVELOPERS CIO MAGAZINE**, Rio de Janeiro, n.38, p. 18-20, abr. 1999.

ANEXO A – SCRIPTS PARA CRIAÇÃO DO PROTOTIPO RELACIONAL

/* Table: TBLCATEGORIA, Owner: SYSDBA */

```
CREATE TABLE TBLCATEGORIA
(
  PKCATEGORIA    INTEGER NOT NULL,
  NOMECATEGORIA        VARCHAR(10),
 PRIMARY KEY (PKCATEGORIA)
);
```

/* Table: TBLCIDADE, Owner: SYSDBA */

```
CREATE TABLE TBLCIDADE
(
  PKCIDADE INTEGER NOT NULL,
  NOMECIDADE     VARCHAR(20),
  FKESTADO         INTEGER NOT NULL,
 PRIMARY KEY (PKCIDADE)
);
ALTER TABLE TBLCIDADE ADD FOREIGN KEY (FKESTADO) REFERENCES
TBLESTADO (PKESTADO);
```

/* Table: TBLDEPARTAMENTO, Owner: SYSDBA */

```
CREATE TABLE TBLDEPARTAMENTO
(
  PKDEPARTAMENTO     INTEGER NOT NULL,
  NOMEDEPARTAMENTO         VARCHAR(20),
 PRIMARY KEY (PKDEPARTAMENTO)
);
```

/* Table: TBLESTADO, Owner: SYSDBA */

```
CREATE TABLE TBLESTADO
(
  PKESTADO         INTEGER NOT NULL,
  SIGLAESTADO    CHAR(2),
  NOMEESTADO    VARCHAR(20),
 PRIMARY KEY (PKESTADO)
);
```

/* Table: TBLLOJA, Owner: SYSDBA */

```
CREATE TABLE TBLLOJA
(
  PKLOJA     INTEGER NOT NULL,
  NOMELOJA         VARCHAR(20),
  FKCIDADE INTEGER,
  FKESTADO         INTEGER,
  FKPLANTA INTEGER,
 PRIMARY KEY (PKLOJA)
);
ALTER TABLE TBLLOJA ADD FOREIGN KEY (FKCIDADE) REFERENCES
TBLCIDADE (PKCIDADE);
ALTER TABLE TBLLOJA ADD FOREIGN KEY (FKESTADO) REFERENCES
TBLESTADO (PKESTADO);
ALTER TABLE TBLLOJA ADD FOREIGN KEY (FKPLANTA) REFERENCES
TBLPLANTA (PKPLANTA);
```

/* Table: TBLMARCA, Owner: SYSDBA */

```
CREATE TABLE TBLMARCA
(
  PKMARCA  INTEGER NOT NULL,
  NOMEMARCA      VARCHAR(20),
 PRIMARY KEY (PKMARCA)
);
```

```
/* Table: TBLPLANTA, Owner: SYSDBA */

CREATE TABLE TBLPLANTA
(
  PKPLANTA INTEGER NOT NULL,
  NOMETIPOPLANTA       VARCHAR(10),
 PRIMARY KEY (PKPLANTA)
);

/* Table: TBLPRODUTO, Owner: SYSDBA */

CREATE TABLE TBLPRODUTO
(
  PKPRODUTO       INTEGER NOT NULL,
  NOMEPRODUTO VARCHAR(30),
  FKMARCA  INTEGER,
  FKSUBCATEGORIA       INTEGER,
  FKDEPARTAMENTO      INTEGER,
 PRIMARY KEY (PKPRODUTO)
);
ALTER TABLE TBLPRODUTO ADD FOREIGN KEY (FKMARCA) REFERENCES
TBLMARCA (PKMARCA);
ALTER TABLE TBLPRODUTO ADD FOREIGN KEY (FKSUBCATEGORIA)
REFERENCES TBLSUBCATEGORIA (PKSUBCATEGORIA);
ALTER TABLE TBLPRODUTO ADD FOREIGN KEY (FKDEPARTAMENTO)
REFERENCES TBLDEPARTAMENTO (PKDEPARTAMENTO);

/* Table: TBLSUBCATEGORIA, Owner: SYSDBA */

CREATE TABLE TBLSUBCATEGORIA
(
  PKSUBCATEGORIA       INTEGER NOT NULL,
```

```
  NOMESUBCATEBOGORIA      VARCHAR(20),
  FKCATEGORIA    INTEGER,
 PRIMARY KEY (PKSUBCATEGORIA)
);
ALTER TABLE TBLSUBCATEGORIA ADD FOREIGN KEY (FKCATEGORIA)
REFERENCES TBLCATEGORIA (PKCATEGORIA);

/* Table: TBLVENDAS, Owner: SYSDBA */

CREATE TABLE TBLVENDAS
(
  PKVENDAS         INTEGER NOT NULL,
  FKPRODUTO       INTEGER,
  FKLOJA      INTEGER,
  UNIDADES INTEGER,
  RECEITA    NUMERIC(15, 2),
  CUSTO       NUMERIC(15, 2),
 PRIMARY KEY (PKVENDAS)
);
ALTER TABLE TBLVENDAS ADD FOREIGN KEY (FKPRODUTO) REFERENCES
TBLPRODUTO (PKPRODUTO);
ALTER TABLE TBLVENDAS ADD FOREIGN KEY (FKLOJA) REFERENCES
TBLLOJA (PKLOJA);
SET TERM ^ ;

/* Triggers only will work for SQL triggers */

CREATE TRIGGER INSVENDAS FOR TBLVENDAS
ACTIVE BEFORE INSERT POSITION 0
As
Begin
   new.PkVendas = Gen_Id(GenPkVendas,1);
End
 ^
```

```
COMMIT WORK ^
SET TERM ;^
```

ANEXO B - SCRIPTS PARA CRIAÇÃO DO PROTÓTIPO EM ESTRELA

/* Table: TBLDIMLOJA, Owner: SYSDBA */

```
CREATE TABLE TBLDIMLOJA
(
  PKLOJA   INTEGER NOT NULL,
  NOMELOJA      VARCHAR(20),
  NOMECIDADE VARCHAR(20),
  SIGLAESTADO         CHAR(2),
  PLANTA  VARCHAR(20),
 PRIMARY KEY (PKLOJA)
);
```

/* Table: TBLDIMPRODUTO, Owner: SYSDBA */

```
CREATE TABLE TBLDIMPRODUTO
(
  PKPRODUTO       INTEGER NOT NULL,
  NOMEPRODUTO VARCHAR(20),
  NOMEMARCA      VARCHAR(20),
  NOMESUBCATEGORIA VARCHAR(20),
  NOMECATEGORIA        VARCHAR(20),
  NOMEDEPARTAMENTO          VARCHAR(20),
 PRIMARY KEY (PKPRODUTO)
);
```

/* Table: TBLDIMPROMOCAO, Owner: SYSDBA */

```
CREATE TABLE TBLDIMPROMOCAO
(
  PKPROMOCAO    INTEGER NOT NULL,
```

```
  NOMEPROMOCAO        VARCHAR(30),
  TIPODEPRECO    VARCHAR(20),
  TIPODEANUNCIO          VARCHAR(20),
  TIPODEDISPLAY VARCHAR(20),
  TIPODECUPOM    VARCHAR(20),
  DATAINICIO          TIMESTAMP,
  DATATERMINO    TIMESTAMP,
 PRIMARY KEY (PKPROMOCAO)
);
/* Table: TBLDIMTEMPO, Owner: SYSDBA */

CREATE TABLE TBLDIMTEMPO
(
  PKTEMPO  INTEGER NOT NULL,
  DATA          TIMESTAMP,
  DIADASEMANA    VARCHAR(9),
  DIANOMESSMALLINT,
  DIASCORRIDOSNOANO          SMALLINT,
  SEMANANOANO  SMALLINT,
  NUMERODASEMANA     SMALLINT,
  MES INTEGER,
  NUMEROTRIMESTRE    SMALLINT,
  ANO SMALLINT,
  INDICADORFERIADO    CHAR(1),
 PRIMARY KEY (PKTEMPO)
);
/* Table: TBLFATOVENDAS, Owner: SYSDBA */

CREATE TABLE TBLFATOVENDAS
(
  PKVENDAS          INTEGER NOT NULL,
  FKTEMPO  INTEGER,
  FKPROMOCAO    INTEGER,
  FKPRODUTO       INTEGER,
```

```
  FKLOJA      INTEGER,
  UNIDADES INTEGER,
  RECEITA    NUMERIC(15, 2),
  CUSTO       NUMERIC(15, 2),
 PRIMARY KEY (PKVENDAS)
);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKTEMPO)
REFERENCES TBLDIMTEMPO (PKTEMPO);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKPROMOCAO)
REFERENCES TBLDIMPROMOCAO (PKPROMOCAO);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKPRODUTO)
REFERENCES TBLDIMPRODUTO (PKPRODUTO);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKLOJA) REFERENCES
TBLDIMLOJA (PKLOJA);
SET TERM ^ ;

/* Triggers only will work for SQL triggers */

CREATE TRIGGER INSFATOVENDAS FOR TBLFATOVENDAS
ACTIVE BEFORE INSERT POSITION 0
As
Begin
   new.PkVendas = Gen_Id(GenPkVendas,1);
End
 ^

COMMIT WORK ^
SET TERM ;^
```

ANEXO C - SCRIPTS PARA CRIAÇÃO DO PROTÓTIPO HÍBRIDO

/* Table: TBLDIMCATEGORIA, Owner: SYSDBA */

```
CREATE TABLE TBLDIMCATEGORIA
(
  PKCATEGORIA    INTEGER NOT NULL,
  NOMECATEGORIA        VARCHAR(10),
 PRIMARY KEY (PKCATEGORIA)
);
```

/* Table: TBLDIMCIDADE, Owner: SYSDBA */

```
CREATE TABLE TBLDIMCIDADE
(
  PKCIDADE INTEGER NOT NULL,
  NOMECIDADE     VARCHAR(20),
  FKESTADO         INTEGER NOT NULL,
 PRIMARY KEY (PKCIDADE)
);
ALTER TABLE TBLDIMCIDADE ADD FOREIGN KEY (FKESTADO) REFERENCES
TBLDIMESTADO (PKESTADO);
```

/* Table: TBLDIMDEPARTAMENTO, Owner: SYSDBA */

```
CREATE TABLE TBLDIMDEPARTAMENTO
(
  PKDEPARTAMENTO     INTEGER NOT NULL,
  NOMEDEPARTAMENTO         VARCHAR(20),
 PRIMARY KEY (PKDEPARTAMENTO)
);
```

/* Table: TBLDIMESTADO, Owner: SYSDBA */

```
CREATE TABLE TBLDIMESTADO
```

```
(
  PKESTADO          INTEGER NOT NULL,
  SIGLAESTADO     CHAR(2),
  NOMEESTADO     VARCHAR(20),
 PRIMARY KEY (PKESTADO)
);
```

/* Table: TBLDIMLOJA, Owner: SYSDBA */

```
CREATE TABLE TBLDIMLOJA
(
  PKLOJA      INTEGER NOT NULL,
  NOMELOJA          VARCHAR(20),
  FKCIDADE INTEGER,
  FKESTADO          INTEGER,
  FKPLANTA INTEGER,
 PRIMARY KEY (PKLOJA)
);
ALTER TABLE TBLDIMLOJA ADD FOREIGN KEY (FKCIDADE) REFERENCES
TBLDIMCIDADE (PKCIDADE);
ALTER TABLE TBLDIMLOJA ADD FOREIGN KEY (FKESTADO) REFERENCES
TBLDIMESTADO (PKESTADO);
ALTER TABLE TBLDIMLOJA ADD FOREIGN KEY (FKPLANTA) REFERENCES
TBLDIMPLANTA (PKPLANTA);
```

/* Table: TBLDIMMARCA, Owner: SYSDBA */

```
CREATE TABLE TBLDIMMARCA
(
  PKMARCA  INTEGER NOT NULL,
  NOMEMARCA      VARCHAR(20),
 PRIMARY KEY (PKMARCA)
);
```

```
/* Table: TBLDIMPLANTA, Owner: SYSDBA */

CREATE TABLE TBLDIMPLANTA
(
  PKPLANTA INTEGER NOT NULL,
  NOMETIPOPLANTA       VARCHAR(10),
 PRIMARY KEY (PKPLANTA)
);

/* Table: TBLDIMPRODUTO, Owner: SYSDBA */

CREATE TABLE TBLDIMPRODUTO
(
  PKPRODUTO       INTEGER NOT NULL,
  NOMEPRODUTO VARCHAR(30),
  FKMARCA  INTEGER,
  FKSUBCATEGORIA       INTEGER,
  FKDEPARTAMENTO      INTEGER,
 PRIMARY KEY (PKPRODUTO)
);
ALTER TABLE TBLDIMPRODUTO ADD FOREIGN KEY (FKMARCA)
REFERENCES TBLDIMMARCA (PKMARCA);
ALTER TABLE TBLDIMPRODUTO ADD FOREIGN KEY (FKSUBCATEGORIA)
REFERENCES TBLDIMSUBCATEGORIA (PKSUBCATEGORIA);
ALTER TABLE TBLDIMPRODUTO ADD FOREIGN KEY (FKDEPARTAMENTO)
REFERENCES TBLDIMDEPARTAMENTO (PKDEPARTAMENTO);

/* Table: TBLDIMPROMOCAO, Owner: SYSDBA */

CREATE TABLE TBLDIMPROMOCAO
(
  PKPROMOCAO    INTEGER NOT NULL,
  NOMEPROMOCAO       VARCHAR(30),
  TIPODEPRECO    VARCHAR(20),
```

```
  TIPODEANUNCIO          VARCHAR(20),
  TIPODEDISPLAY VARCHAR(20),
  TIPODECUPOM   VARCHAR(20),
  DATAINICIO         TIMESTAMP,
  DATATERMINO    TIMESTAMP,
 PRIMARY KEY (PKPROMOCAO)
);
```

/* Table: TBLDIMSUBCATEGORIA, Owner: SYSDBA */

```
CREATE TABLE TBLDIMSUBCATEGORIA
(
  PKSUBCATEGORIA       INTEGER NOT NULL,
  NOMESUBCATEBOGORIA       VARCHAR(20),
  FKCATEGORIA    INTEGER,
 PRIMARY KEY (PKSUBCATEGORIA)
);
ALTER TABLE TBLDIMSUBCATEGORIA ADD FOREIGN KEY (FKCATEGORIA)
REFERENCES TBLDIMCATEGORIA (PKCATEGORIA);
```

/* Table: TBLDIMTEMPO, Owner: SYSDBA */

```
CREATE TABLE TBLDIMTEMPO
(
  PKTEMPO  INTEGER NOT NULL,
  DATA         TIMESTAMP,
  DIADASEMANA    VARCHAR(9),
  DIANOMESSMALLINT,
  DIASCORRIDOSNOANO          SMALLINT,
  SEMANANOANO  SMALLINT,
  NUMERODASEMANA     SMALLINT,
  MES INTEGER,
  NUMEROTRIMESTRE    SMALLINT,
  ANO SMALLINT,
```

```
  INDICADORFERIADO    CHAR(1),
 PRIMARY KEY (PKTEMPO)
);
```

/* Table: TBLFATOVENDAS, Owner: SYSDBA */

```
CREATE TABLE TBLFATOVENDAS
(
  PKVENDAS          INTEGER NOT NULL,
  FKTEMPO  INTEGER,
  FKPROMOCAO    INTEGER,
  FKPRODUTO       INTEGER,
  FKLOJA      INTEGER,
  FKCATEGORIA    INTEGER,
  FKSUBCATEGORIA       INTEGER,
  FKMARCA  INTEGER,
  FKPLANTA INTEGER,
  FKESTADO          INTEGER,
  FKCIDADE INTEGER,
  FKDEPARTAMENTO      INTEGER,
  UNIDADES INTEGER,
  RECEITA    NUMERIC(15, 2),
  CUSTO       NUMERIC(15, 2),
  IDENTOPERACAO         VARCHAR(9),
 PRIMARY KEY (PKVENDAS)
);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKTEMPO)
REFERENCES TBLDIMTEMPO (PKTEMPO);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKPROMOCAO)
REFERENCES TBLDIMPROMOCAO (PKPROMOCAO);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKPRODUTO)
REFERENCES TBLDIMPRODUTO (PKPRODUTO);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKLOJA) REFERENCES
TBLDIMLOJA (PKLOJA);
```

```
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKCATEGORIA)
REFERENCES TBLDIMCATEGORIA (PKCATEGORIA);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKSUBCATEGORIA)
REFERENCES TBLDIMSUBCATEGORIA (PKSUBCATEGORIA);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKMARCA)
REFERENCES TBLDIMMARCA (PKMARCA);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKPLANTA)
REFERENCES TBLDIMPLANTA (PKPLANTA);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKESTADO)
REFERENCES TBLDIMESTADO (PKESTADO);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKCIDADE)
REFERENCES TBLDIMCIDADE (PKCIDADE);
ALTER TABLE TBLFATOVENDAS ADD FOREIGN KEY (FKDEPARTAMENTO)
REFERENCES TBLDIMDEPARTAMENTO (PKDEPARTAMENTO);
SET TERM ^ ;

/* Triggers only will work for SQL triggers */

CREATE TRIGGER INSFATOVENDAS FOR TBLFATOVENDAS
ACTIVE BEFORE INSERT POSITION 0
As
Begin
   new.PkVendas = Gen_Id(GenPkVendas,1);
End
 ^

COMMIT WORK ^
SET TERM ;^
```

ANEXO D – SCRIPTS DE CARGA DOS CUBOS DE DECISÃO

/* Modelo relacional /*

```
SELECT Tbllloja.NOMELOJA, Tblcidade.NOMECIDADE, Tblestado.SIGLAESTADO,
Tblproduto.NOMEPRODUTO, marca.NOMEMARCA,
Tbldepartamento.NOMEDEPARTAMENTO, Tblcategoria.NOMECATEGORIA,
Tblsubcategoria.NOMESUBCATEBOGORIA, SUM( Tblvendas.UNIDADES ), SUM(
Tblvendas.RECEITA ), SUM( Tblvendas.CUSTO ), COUNT( Tblvendas.UNIDADES
), COUNT( Tblvendas.RECEITA ), COUNT( Tblvendas.CUSTO ), COUNT( * )
COUNTALL
FROM TBLVENDAS Tblvendas
  INNER JOIN TBLLOJA Tbllloja
  ON  (Tbllloja.PKLOJA = Tblvendas.FKLOJA)
  INNER JOIN TBLPRODUTO Tblproduto
  ON  (Tblproduto.PKPRODUTO = Tblvendas.FKPRODUTO)
  INNER JOIN TBLCIDADE Tblcidade
  ON  (Tblcidade.PKCIDADE = Tbllloja.FKCIDADE)
  INNER JOIN TBLMARCA Tblmarca
  ON  (Tblmarca.PKMARCA = Tblproduto.FKMARCA)
  INNER JOIN TBLDEPARTAMENTO Tbldepartamento
  ON  (Tbldepartamento.PKDEPARTAMENTO = Tblproduto.FKDEPARTAMENTO)
  INNER JOIN TBLSUBCATEGORIA Tblsubcategoria
  ON  (Tblsubcategoria.PKSUBCATEGORIA = Tblproduto.FKSUBCATEGORIA)
      INNER JOIN TBLESTADO TBLESTADO
  ON  (Tblestado.PKESTADO = Tblcidade.FKESTADO)
  INNER JOIN TBLCATEGORIA Tblcategoria
  ON  (Tblcategoria.PKCATEGORIA = Tblsubcategoria.FKCATEGORIA)
GROUP BY Tbllloja.NOMELOJA, Tblcidade.NOMECIDADE,
Tblestado.SIGLAESTADO, Tblproduto.NOMEPRODUTO, Tblmarca.NOMEMARCA,
Tbldepartamento.NOMEDEPARTAMENTO, Tblcategoria.NOMECATEGORIA,
Tblsubcategoria.NOMESUBCATEBOGORIA
```

/* Modelo Dimensional */

```
SELECT Tbldimloja.NOMELOJA, Tbldimloja.NOMECIDADE,
Tbldimloja.SIGLAESTADO, Tbldimloja.PLANTA, Tbldimproduto.NOMEPRODUTO,
Tbldimproduto.NOMEMARCA, Tbldimproduto.NOMESUBCATEGORIA,
Tbldimproduto.NOMECATEGORIA, Tbldimproduto.NOMEDEPARTAMENTO, SUM(
Tblfatovendas.UNIDADES ), SUM( Tblfatovendas.RECEITA ), SUM(
Tblfatovendas.CUSTO ), COUNT( Tblfatovendas.UNIDADES ), COUNT(
Tblfatovendas.RECEITA ), COUNT( Tblfatovendas.CUSTO )
FROM TBLFATOVENDAS Tblfatovendas
  INNER JOIN TBLDIMLOJA Tbldimloja
  ON  (Tbldimloja.PKLOJA = Tblfatovendas.FKLOJA)
  INNER JOIN TBLDIMPRODUTO Tbldimproduto
  ON  (Tbldimproduto.PKPRODUTO = Tblfatovendas.FKPRODUTO)
GROUP BY Tbldimloja.NOMELOJA, Tbldimloja.NOMECIDADE,
Tbldimloja.SIGLAESTADO, Tbldimloja.PLANTA, Tbldimproduto.NOMEPRODUTO,
Tbldimproduto.NOMEMARCA, Tbldimproduto.NOMESUBCATEGORIA,
Tbldimproduto.NOMECATEGORIA, Tbldimproduto.NOMEDEPARTAMENTO
```

/* Modelo híbrido /*

```
SELECT Tbldimproduto.NOMEPRODUTO, Tbldimloja.NOMELOJA,
Tbldimcategoria.NOMECATEGORIA,
Tbldimsubcategoria_1.NOMESUBCATEBOGORIA, Tbldimmarca.NOMEMARCA,
Tbldimplanta.NOMETIPOPLANTA, Tbldimestado.SIGLAESTADO,
Tbldimcidade.NOMECIDADE, Tbldimdepartamento.NOMEDEPARTAMENTO, SUM(
Tblfatovendas_1.CUSTO ), SUM( Tblfatovendas_1.UNIDADES ), SUM(
Tblfatovendas_1.RECEITA ), COUNT( Tblfatovendas_1.CUSTO ), COUNT(
Tblfatovendas_1.UNIDADES ), COUNT( Tblfatovendas_1.RECEITA )
FROM TBLFATOVENDAS Tblfatovendas_1
  INNER JOIN TBLDIMPRODUTO Tbldimproduto
  ON  (Tbldimproduto.PKPRODUTO = Tblfatovendas_1.FKPRODUTO)
  INNER JOIN TBLDIMLOJA Tbldimloja
  ON  (Tbldimloja.PKLOJA = Tblfatovendas_1.FKLOJA)
  INNER JOIN TBLDIMCATEGORIA Tbldimcategoria
  ON  (Tbldimcategoria.PKCATEGORIA = Tblfatovendas_1.FKCATEGORIA)
```

```
  INNER JOIN TBLDIMSUBCATEGORIA Tbldimsubcategoria_1
  ON  (Tbldimsubcategoria_1.PKSUBCATEGORIA =
Tblfatovendas_1.FKSUBCATEGORIA)
  INNER JOIN TBLDIMMARCA Tbldimmarca
  ON  (Tbldimmarca.PKMARCA = Tblfatovendas_1.FKMARCA)
  INNER JOIN TBLDIMPLANTA Tbldimplanta
  ON  (Tbldimplanta.PKPLANTA = Tblfatovendas_1.FKPLANTA)
  INNER JOIN TBLDIMESTADO Tbldimestado
  ON  (Tbldimestado.PKESTADO = Tblfatovendas_1.FKESTADO)
  INNER JOIN TBLDIMCIDADE Tbldimcidade
  ON  (Tbldimcidade.PKCIDADE = Tblfatovendas_1.FKCIDADE)
  INNER JOIN TBLDIMDEPARTAMENTO Tbldimdepartamento
  ON  (Tbldimdepartamento.PKDEPARTAMENTO =
Tblfatovendas_1.FKDEPARTAMENTO)
GROUP BY Tbldimproduto.NOMEPRODUTO, Tbldimloja.NOMELOJA,
Tbldimcategoria.NOMECATEGORIA,
Tbldimsubcategoria_1.NOMESUBCATEBOGORIA, Tbldimmarca.NOMEMARCA,
Tbldimplanta.NOMETIPOPLANTA, Tbldimestado.SIGLAESTADO,
Tbldimcidade.NOMECIDADE, Tbldimdepartamento.NOMEDEPARTAMENTO
```