Pós-Graduação em Ciência da Computação

“Proposta de Esquema Dimensional Hierárquico Genérico para Implementação em SGBD Relacional”

Por

Ney Paranaguá de Carvalho

Dissertação de Mestrado

Universidade Federal de Pernambuco

posgraduacao@cin.ufpe.br

www.cin.ufpe.br/~posgraduacao

RECIFE, FEVEREIRO / 2003

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

NEY PARANAGUÁ DE CARVALHO

"Proposta de Esquema Dimensional Hierárquico Genérico para Implementação em SGBD Relacional"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: Décio Fonseca

RECIFE, fevereiro / 2003

Dedicatória

Dedico este trabalho a minha avó materna, D. Alice Paranaguá, presença sempre vigilante e amorosa em minha vida, incentivando ao estudo e ao progresso, exemplo da certeza de que vale a pena o exercício da boa vontade.

Dedico ainda ao companheiro de mestrado Alonso Amorim, exemplo da perseverança e superação, que nos mostrou que a vitória sobre as adversidades é possível se nós acreditarmos e que, depois de muita luta, descansou.

Agradecimentos

Agradeço ao Deus maravilhoso, em quem acredito e confio.

Agradeço a minha esposa, Vanessa, pela forma especial com que me dedica o seu amor.

Agradeço a minha família, Sônia, minha adorada mãe, e aos meus irmãos Cláudia, José Filho e Ricardo, a quem dedico todo o carinho, fraternidade e cuidados, e a Waledice, minha tia.

Agradeço aos meus sobrinhos, Isabella, Vítor, Rodrigo, Vinicius e Lucas, todos especiais em minha vida.

Agradeço ao amigo Adalton Sena, sempre companheiro, atencioso e responsável, que desbrava, em conjunto comigo, os caminhos da computação no Piauí.

Agradeço a todos os amigos da Infoway, empresa em que trabalho, Peta, Chico, Conceição, Marcos Roberto, Luciani, Maxwel, Franz e Rodrigo e aos que já passaram por lá.

Agradeço ao amigo e orientador Décio Fonseca, pela confiança no desenvolvimento deste trabalho e pela tranqüilidade na condução de todo processo, decisiva para sua chegada a bom termo.

Agradeço aos professores do Centro de Informática da UFPE que foram até Teresina para nos granjear a sua experiência e conhecimento, Fernando Fonseca, Wilson Rosa, Manoel Eusébio, Carolina Salgado e Valéria Times.

Agradeço ao CEFET-Pi na pessoa de sua diretora geral, Rita Martins de Cássia, por nos ter oportunizado a participação neste mestrado.

Agradeço ao amigo Fábio Brandão de Almeida, pela sua acolhida sempre generosa durante as estadas em Recife.

Resumo

Os Sistemas de Informação têm evoluído naturalmente, impulsionados pelo aumento do volume de dados tratado e armazenado, pela disponibilidade de novas e mais baratas tecnologias para manipulação dos dados e por demandas mais complexas por informação e conhecimento, originadas do usuário. Este, por sua vez, teve o seu perfil expandido: se antes era preponderantemente de nível operacional, hoje apresenta com maior freqüência o nível de gerência e executivo, o que exige uma visão progressivamente mais ampla e corporativa do negócio.

Diante deste cenário, tecnologias para o tratamento da informação foram desenvolvidas para o atendimento às novas demandas, especialmente voltadas para a modelagem do negócio sob uma nova ótica – a dimensional – e a recuperação de informações e conhecimento diretamente pelo usuário final. No centro das novas tecnologias está o Data Warehouse, grande repositório integrador dos dados corporativos do negócio.

Os esquemas atualmente existentes para a modelagem dimensional de Data Warehouse em SGBDs relacionais – estrela e, sua derivação, flocos de neve – são simples o suficiente para proporcionar a compreensão do modelo pelo usuário final e para possibilitar um excelente desempenho no processamento de consultas. No entanto, não atacam diretamente questões que consideramos importantes: a) permitir que o usuário final realize, sem a intervenção do projetista de banco de dados, a concepção da modelagem de seu negócio; b) permitir que o usuário final diretamente implemente as modificações requeridas na modelagem dimensional decorrentes da alteração do negócio ou das alterações de sua visão do negócio e; c) poder ser genérico a fim de permitir que qualquer modelagem dimensional realizada possa ser implementada sobre os mesmos projetos lógico e físico de bancos de dados relacionais.

Para atender a estas questões propusemos neste trabalho um esquema para modelagem dimensional implementada em SGBD relacional genérico que

apresenta como principal característica uma estrutura de dimensões hierárquica construída sobre um único projeto lógico e físico de banco de dados. A utilização das ferramentas construídas a partir deste novo modelo permite que o usuário final, sem intervenção alguma, possa construir diretamente um modelo para o seu negócio unicamente influenciado pela forma como ele o compreende.

Palavras-Chave: Sistemas de Informação, Data Warehouse, Data Marts, Bancos de Dados, Modelagem Dimensional e Metadados.

Abstract

The Information Systems have evolved naturally, stimulated for the increase of the treated and stored volume of data, for the availability of new and cheaper technologies for manipulation of data and for more complex demands for information and knowledge, originated from the user. This, in turn, had its expanded profile: if before it was preponderantly of operational level, nowadays presents more frequently the level of management and executive, which demands a gradually ampler and corporative vision of the business.

Below in this scenary, technologies for the information treatment had been developed attend new demands, especially towards the modeling of the business under a new optics - the dimensional - and the recovery of information and knowledge directly for the final user. In the center of the new technologies is the Date Warehouse, great integrator repository of the corporative data of the business.

The currently existing schemmas for the dimensional modeling of Data Warehouse in RDBMS - star and, its derivation, snowflake - are simple enough to provide the understanding of the model for the final user and to make possible an excellent performance in the inquiry processing, as well. However, they do not attack directly questions that we consider important: a) to allow that the final user carries on, without the intervention of the designer of data base, the conception of the modeling of his or her business; b) to allow that the final user directly implements the modifications required in the dimensional modeling decurrent of the alteration of the business or the alterations of his or her vision of business and; c) to be generic in order to allow that any carried on dimensional modeling can be implemented on the same logical and physical projects of RDBMS.

To take care of these questions we considered in this work a schemma for implementing dimensional modeling in generic RDBMS that presents as its main characteristic a hierarchic structure of dimensions constructed on an only logical and physical project of data base. The use of the tools constructed from

this new model allows that the final user, without any intervention , can directly construct a model for his or her business influenced solely by the form as he or she understands it.

Keywords: Information Systems, Data Warehouse, Data Marts, Data Bases, Dimensional Modeling and Metadata.

Lista de Figuras

Lista de Tabelas

ÍNDICE

CAPíTULO 1
INTRODUÇÃO

1.1 - APRESENTAÇÃO

Este trabalho busca apresentar uma contribuição ao permanente esforço envidado desde o início da computação eletrônica digital de tornar a utilização de suas tecnologias e ferramentas acessíveis ao usuário final, considerando-as como instrumentos ordinários e naturais dentro dos ambientes das organizações. Desta forma, há que se considerar que os autômatos digitais têm a sua existência inerentemente associada à necessidade de representação dos sistemas de informações dos ambientes reais, sejam naturais ou desenvolvidos pelo homem, e estes, de forma geral, apresentam um comportamento pautado em um regime de mudanças, de qualquer ordem.

Assim, os sistemas de informações baseados em computação, ao tempo que devem sempre procurar estar ao alcance da manipulação direta do usuário fim, devem agregar às suas características a máxima adaptabilidade possível, garantindo a sua aderência à realidade, que é mutável, através de ferramentas utilizáveis pelo próprio usuário fim, natural conhecedor dos sistemas reais.

1.2 - HISTÓRICO

A opção pela utilização de sistemas digitais binários representa, sem dúvida, a melhor alternativa para a construção de equipamentos eletrônicos que realizem o automatismo de processos. Sobre uma célula binária, com comportamento que representa alternativas diretas e simples, se pode construir formulações decisórias de qualquer complexidade. Ao mesmo tempo, estas são as células que apresentam as menores dificuldades tecnológicas para a construção, maior velocidade de funcionamento e menor taxa de erros. No entanto, traz em sua natureza um aspecto negativo: a dificuldade, para o homem, de manipular

uma forma de tratamento de dados e processos tão distante da sua forma natural de representação de tais entes. Assim, desde o início, a comunidade de computação tem buscado abstrair a utilização dos sistemas de computação dos aspectos relacionados a sua implementação, do denominado “baixo nível”, aproximando-a da forma natural para a ótica do usuário.

Nesta busca, houve, desde o desenvolvimento dos sistemas operacionais, linguagens de programação de alto nível, sistemas de gerenciamento de bancos de dados, interfaces gráficas para os usuários, linguagem estruturada de consultas e multimídia, até as propostas relacionadas àinteligência artificial.

No âmbito dos sistemas de informações tal comportamento também tem se verificado. Assim, evoluiu-se dos sistemas cuja ênfase é notadamente transacional - visando o controle e o registro de eventos - que apresentam características bastante objetivas e que, por este motivo, tem simplificado o processo de formalização de seus modelos através da tecnologia do tratamento de informações, até os mais recentes, cuja ênfase se dá sobre a estratégia e o planejamento, e cuja alta carga informacional de seus dados e o subjetivismo envolvido – inerente à alta capacidade atômica do tratamento da informação pelo homem – de forma diversa à anterior, têm representado grandes obstáculos para a formalização de seus modelos.

Estes últimos, no entanto, são propostas para aproximar o tratamento da informação através de sistemas de computação da forma mais sofisticada como a que as pessoas podem fazê-lo.

1.3 – CONTEXTUALIZAÇÃO – OS SISTEMAS DE INFORMAÇÕES

As organizações de uma maneira geral apresentam vários níveis diferentes para o tratamento dos dados e informações. Segundo [Beuren et al 01], os mesmos constituiriam uma pirâmide em que, na sua base estariam os SA (Sistemas de Automação) e os SPT (Sistemas de Processamento de Transações) formando o nível de automação/operacional; em um nível intermediário, denominado gerencial, os SAD (Sistemas de Apoio à Decisão) e SIG (Sistemas de Informações Gerenciais); e no topo, no nível estratégico, os SIE (Sistemas de Informações Executivas). Em cada um destes níveis encontram-se características diferenciais, decorrentes diretamente dos propósitos a que estes se destinam. Ainda de acordo com [Beuren et al 01], "os sistemas de informações que atuam no segmento transacional visam concretizar e armazenar as operações realizadas na organização". O seu propósito leva então a um alto nível de detalhamento no armazenamento dos dados, cuja volatilidade é acentuada. Um outro aspecto é a freqüência de atualizações ao que o mesmo é submetido, o que leva a exigências operacionais em que preponderam os aspectos do desempenho e da garantia da integridade e consistência dos dados armazenados. Os Sistemas de Processamento de Transações são a base para a construção dos demais sistemas de informações com carga informacional maior e têm como objetivo fundamental a representação fidedigna das transações e processos reais, com o conseqüente exercício do controle e de decisões em nível operacional. [Beuren et al 01] considera que "um sistema de processamento de transações coleta e armazena dados sobre transações e algumas vezes controla decisões que são feitas como parte da transação". Para que os demais níveis de sistemas de informações sejam possíveis é indispensável que este se apresente de forma absolutamente confiável. "O SPT é o alicerce que sustenta a integridade e a precisão da informação gerada, assegurando a confiabilidade dos sistemas de informações hierarquicamente acima dele", conforme [Beuren et al 01].

Os Sistemas de Apoio à Decisão (SAD) são aqueles formados por um grupo organizado de pessoas, procedimentos, bancos de dados, e dispositivos usados para dar apoio à tomada de decisões referentes a problemas específicos. Segundo [Sprague&Watson 91], o SAD caracteriza-se como "um sistema computacional interativo que ajuda os responsáveis pela tomada de decisões a utilizar dados e modelos para resolver problemas não estruturados". Observa-se que este nível de sistema de informação não objetiva o exercício direto do controle dos procedimentos operacionais da organização, mas unicamente a, baseados naqueles, produzir acervo de informações que possa ser utilizado por gerentes intermediários a analistas de negócio em decisões comerciais que não possam ser estruturadas e formalizadas no primeiro nível. Em complemento os Sistemas de Informações Gerenciais (SIG) que, por seu turno, mesmo tendo o mesmo foco de usuário, oferece informações sumarizadas contemplando o comportamento dos negócios nos períodos passados através de totalizações e consolidando operações realizadas, formam o acervo tecnológico para o tratamento de informações em nível gerencial.

Em nível estratégico temos os Sistemas de Informações Executivas (SIE), em geral construídos sobre repositórios de dados de grandes volumes, cujos dados apresentam um comportamento estático, ou seja, são registros de eventos que ocorreram anteriormente dentro da organização e que não devem mudar – não são voláteis. Estes sistemas de informações são os guardiões da cultura, da história e da identidade das organizações e devem ser as principais fontes para o exercício do planejamento estratégico, apresentando-se como importante vantagem competitiva no cenário da alta concorrência hoje existente. Estes repositórios de dados apresentam um comportamento diametralmente oposto aos do nível basilar. Neste caso, não há preocupação com os aspectos operacionais de seu funcionamento no que tange a garantias sobre concorrência no acesso modificativo aos dados armazenados, vez que, como são registros de eventos passados não mais sofrerão modificações. As

preocupações são desviadas para a órbita da representação dos dados e dos seus relacionamentos, com intuito de que o seu tratamento seja natural para os usuários, afastando-os da necessidade do conhecimento de técnicas de projetos de bancos de dados para formalização de modelos dos seus sistemas de informações.

1.4 – MOTIVAÇÃO - O COMPORTAMENTO DOS ESQUEMAS DE BANCOS DE DADOS

Um banco de dados é construído sobre um esquema. Este representa a estrutura sobre a qual os dados devem ser armazenados e tratados e, portanto, deve ser construído através de um processo que garanta o alcance do melhor modelo de representação dos sistemas reais, observando-se o nível do tratamento de informações desejado. De acordo com [Silberschatz et al 99] o esquema é o projeto geral do banco de dados e deve ser alterado com pouca freqüência. Esta estrutura representa o modelo de dados que, ainda por [Silberschatz et al 99], são "um conjunto de ferramentas conceituais usadas para a descrição de dados, relacionamento entre dados, semântica de dados e regras de consistência". Confirmando o aspecto estático, caracterizado pela não alteração, do esquema de banco de dados, [Elmasri&Navathe 00] afirma que "a descrição de um banco de dados é chamado de esquema de banco de dados, que é especificado durante o projeto do banco de dados e não tem expectativa de mudanças".

O outro aspecto complementar para a construção do banco de dados é, uma vez projetado o seu esquema, providenciar o seu povoamento com os dados, que são denominados de instâncias do banco de dados. Estes apresentam um comportamento notadamente dinâmico, contrapondo-se ao do esquema, reflexo das mudanças de conteúdo sofridas continuamente pelos ambientes reais.

No entanto, os ambientes reais também podem se modificar. È bem verdade que esta freqüência de modificações tende a ser bastante baixa, mas, no entanto, a possibilidade existe conforme registra [Elmasri&Navathe 00] quando afirma que "mudanças de esquema são geralmente necessárias quando as exigências das aplicações de banco de dados mudam. Os recentes sistemas de banco de dados incluem operações para permitir alterações de esquema, embora o processo de mudança de esquema seja mais evoluído do que simples atualizações do banco de dados". Observa-se, desta forma, que há um esforço no sentido de se prover mecanismos automatizados que possibilitem alterações em esquemas de bancos de dados, sem a necessidade de se rescrever todo o seu modelo.

Associado ao questionamento da dinamicidade dos esquemas de bancos de dados há aquele, talvez anterior, relacionado à sua própria construção. Para a construção do esquema utiliza-se de processos de modelagem – modelo de dados – que exigem o conhecimento de um conjunto de regras formais, vinculadas ao conjunto de habilidades apresentadas tão somente por profissionais da área de projeto de banco de dados e naturalmente afastadas da compreensão direta dos usuários finais. Estes, por sua vez, são os que realmente conhecem o ambiente real. Forma-se, portanto, uma distância natural entre os conhecedores do ambiente real e os conhecedores das técnicas de modelagem utilizadas na construção dos sistemas de bancos de dados.

Em sistemas de informações de nível de automatismo/operacional este impasse ainda encontra-se distante de ser superado, vez que estes apresentam características – notadamente no que tange ao desempenho e freqüência de atualizações - que exigem, em nível de projeto do banco de dados, conhecimentos técnicos especializados da área da computação. Por outro lado, os sistemas de informações em nível estratégico, baseados em conjuntos de dados não voláteis e com alta carga informacional - resumidos

dos dados operacionais - e unicamente compromissados com a melhor representação da realidade, apresentam um cenário atraente para a proposição de técnicas ou ferramentas de modelagem de esquemas para manipulação direta do usuário final, conhecedor do ambiente real.

1.5 – OBJETIVOS - O USUÁRIO MODELA O DATA WAREHOUSE

Para a implementação dos sistemas de informações em qualquer nível há um conjunto de tecnologia de modelagem disponíveis. Assim, para [Silberschatz et al 99], pode se optar por modelos baseados em objetos ou em registros e, dentro destas alternativas, ainda opta-se por Modelo de Entidades e Relacionamentos ou Orientados a Objetos, Hierárquicos, em Rede ou Relacionais. Quando tratamos, entretanto, com os sistemas de nível estratégico, um grupo específico de tecnologia é demandado à guisa das características específicas oriundas da forma peculiar com que as informações são observadas, tratadas, armazenadas e extraídas. São tecnologias ainda recentes, sobre as quais há um grande esforço de estudo e depuração. Sobre o tema, [Beuren et al 01] registra, de forma algo extremista, que “a sustentação teórica de sistemas de informação para os altos escalões da organização é praticamente nula”. No que tange a modelagem e armazenamento, os sistemas de informações em nível estratégico utilizam-se do *Data Warehouse*.

[Inmon&Hackathorn 97] afirma que “o *data warehouse* é o ponto central da arquitetura de processamento de informações para sistemas de informática modernos suportando o processamento informacional (SAD – sistemas de apoio a decisão) através de um alicerce sólido de integração de dados corporativos e históricos para a realização de análises gerenciais”. Observa-se, entretanto, que esta tecnologia utiliza-se de técnicas de modelagem que produzem modelos especializados para cada ambiente real tratado e demanda,

necessariamente, a presença de profissionais da área de projeto de bancos de dados para a sua construção. Da mesma forma, as mudanças apresentadas no ambiente real não serão implementadas diretamente pelo usuário final – analista de negócios – na modelagem.

Diante deste cenário, estudamos e propusemos neste trabalho um esquema genérico para a modelagem de *Data Warehouse*, adaptável a qualquer ambiente real, baseado em uma hierarquia de dimensões e fatos e implementamos tal proposta em um protótipo, que é apresentado no Capítulo 5 deste trabalho.

1.6 – ORGANIZAÇÃO DO TRABALHO

O presente trabalho é organizado em 6 (seis) capítulos, que apresentam os conteúdos a seguir:

Capítulo 1 – Introdução: são apresentados o histórico e a contextualização da área de conhecimento e tecnologia a ser trabalhada, em associação com os aspectos do tratamento da informação que motivaram a proposição deste trabalho e os objetivos a serem alcançados com a sua consecução;

Capitulo 2 – *Data Warehouse* e Modelagem Multidimensional: são relacionados os aspectos históricos e as características diferenciadoras desta tecnologia, bem como as principais propostas para sua arquitetura, o papel dos metadados e os passos necessários para a sua construção; são também apresentados e discutidos os principais aspectos relacionados aos esquemas existentes para a modelagem dimensional, o esquema em estrela e o de flocos de neve. São estudados os papéis das dimensões e dos fatos, além do comportamento das operações de junção entre estas;

Capítulo 3 – Esquema Dimensional Hierárquico Genérico: é apresentada a nossa proposta para a modelagem dimensional e a sua implementação em um projeto único para bancos de dados relacionais, garantindo o objetivo inicial de haver generalidade - suportar a modelagem de qualquer negócio - e dinamicidade, permitir que as alterações do negócio sejam diretamente implementadas no modelo dimensional pelo usuário;

Capítulo 4 – Protótipo do Esquema Dimensional Hierárquico Genérico: é apresentada a implementação da proposta em ambiente *Delphi* 6 / SQL 2000, sendo relacionados as categorias de usuário que foram concebidas para a administração do *data warehouse*, os seus papéis e as ferramentas de administração utilizadas por estes, bem como alguns exemplos de interação com o repositório criado;

Capítulo 5 – Conclusão: são apresentadas a análise dos resultados obtidos e as avaliações sobre as contribuições e limitações do trabalho. São realizadas ainda considerações sobre que aspectos do esquema proposto poderiam ser melhorados;

Capítulo 6 - Bibliografia: são relacionadas as referências bibliográficas utilizadas no trabalho;

Apêndice A – *Scripts* para a Criação do Banco de Dados: são relacionados os *scripts* SQL necessários para a criação do banco de dados relacional utilizado no esquema proposto;

Apêndice B – *Stored Procedures:* são relacionadas as *stored procedures* utilizadas para o funcionamento do banco de dados relacional utilizado no esquema proposto;

Apêndice C – Visões: são relacionadas aos *scripts* SQL para a criação das visões aplicadas ao banco de dados relacional utilizado no esquema proposto.

CAPíTULO 2

DATA WAREHOUSE E MODELAGEM MULTIDIMENSIONAL

2.1 – INTRODUÇÃO

A tecnologia de data warehouse é produto de uma evolução histórica da presença dos sistemas de informações nas organizações. Esta história iniciou-se com a utilização dos sistemas de primeiro nível (automação e transacional) cuja atuação têm se expandido nas diversas áreas de operação das organizações, com o conseqüente crescimento substancial no volume de dados armazenados e no aumento da demanda dos gerentes de médio e alto nível e executivos por informações que, não somente tratem o negócio, mas possibilitem uma análise do mesmo a fim de se obter suporte para os processos decisórios que envolvam a estratégia e o planejamento das suas ações.

Desta forma, sobre o arcabouço de dados operacionais e tendo-os como fontes primitivas, desenvolveu-se toda uma tecnologia para garantir a construção e a manipulação dos sistemas de informação mais sofisticados, dotados de alta carga informacional, denominados estratégicos. A opção tecnológica que trata da extração e carga dos dados operacionais – até externos aos sistemas de processamento de transações –, sua integração, armazenamento e tratamento em um repositório de dados que permite a sua recuperação sob diversas formas para o suporte aos processos de tomada de decisão é denominado de data warehouse [Inmon&Hackthorn 97] [Inmon&Welch 99] [Kimball&Ross 02].

2.2 – HISTÓRICO

A partir da década de 70 com a consolidação das tecnologias de armazenamento em disco e o surgimento de uma ferramenta em software denominada Sistemas de Gerenciamento de Bancos de Dados, surgiu a idéia de se integrar todo o conjunto de dados das organizações em uma fonte única denominada banco de dados. A presença dos bancos de dados permitiu que a observação dos aspectos que formavam as organizações fossem baseadas em dados, em contraposição às óticas anteriores baseadas nos processos existentes. Assim, o banco de dados tornou-se um recurso corporativo básico e formou-se a convicção que os sistemas de informações baseados em computação poderiam ser utilizados para controle de gestão dos negócios.

Com a popularização da computação, acentuada a partir do final da década de 70 e consolidada na década de 80, decorrentes do desenvolvimento de computadores com cada vez maior capacidade de processamento aliada à disponibilização de ferramentas em software com maior usabilidade para o usuário final, os profissionais de maior nível das organizações passaram a ser inseridos no rol dos utilizadores daquelas opções tecnológicas, que passaram a exigir destas o atendimento a suas demandas específicas, voltadas prioritariamente àanálise do negócio e àtomada de decisões estratégicas.

Assim, como os recursos tecnológicos para o tratamento dos sistemas de informações tão somente, à época, estavam focados na controle da operação, passou-se a demandar por ferramentas que pudessem extrair os dados operacionais existentes para tratamento em ambientes externos que possibilitassem o exercício das atividades de análise, os denominados Programas Extratores [Kimball&Ross 02]. Estes programas levavam os dados para ambientes intermediários onde eram tratados, o que proporcionava constantemente falhas na integridade das informações, face às necessárias intervenções manuais para se obter a integração dos dados armazenados sob

as mais diversas formas, o que afetava diretamente a consistência dos dados, com o comprometimento da credibilidade e divulgação de valores divergentes.

Para a resolução destes problemas concebeu-se uma alternativa de integrar todos estes dados em uma única base central que, diferentemente dos sistemas de processamento de transações, deveria manter o registro histórico das informações e fazer com que elas fossem armazenada dimensionalmente, ou seja, o mesmo fato (dado) deveria ser analisado sob as mais diversas dimensões diferentes.

Com o data warehouse foi necessária a proposição de novos métodos para a estruturação de dados, tanto no que tange ao armazenamento quanto à recuperação. As suas técnicas de projeto e construção são absolutamente divergentes daquelas utilizadas em sistemas de processamento de transações, haja vista que os usuários, o conteúdo, a administração, o gerenciamento, o ritmo, as solicitações e o volume de dados são diferentes.

2.3 – CONCEITO

O data warehouse não pode ser tratado como um objeto atômico que simplesmente seja instalado em uma organização. Ao contrário, a sua perspectiva como processo é muito mais enfática, vez que para a sua obtenção estão envolvidos uma série de passos que demandam esforços que muitas vezes não encontraram uma representação objetiva para que pudessem ser formalizados e que possibilitassem a construção de autômatos. Assim, a presença do homem em sua construção é imprescindível, variando apenas o grau de extensão de suas funções.

Há várias conceituações existentes para data warehouse, que variam de uma ótica mais centrada na idéia de um produto a outras que combinam a utilização de produtos através de processos. Estas últimas parecem as mais razoáveis.

O conceito clássico pode ser encontrado em [Inmon&Hackthorn 97]: "um data warehouse é uma coleção de dados orientados a assuntos, integrados, variáveis com o tempo, não voláteis para suporte ao processo gerencial de tomada de decisão". Pela definição dada, nota-se uma ênfase no produto que apresenta determinadas características, o que pode levar a uma simplificação indevida dos aspectos relacionados à construção deste produto, que são muito mais complexos. Antes da conceituação, no entanto, [Inmon&Hackthorn 97] considera que "o data warehouse é construído e implementado de uma maneira evolucionária passo a passo, organizando e armazenando os dados necessários para a análise informacional e o processamento analítico sob uma perspectiva de longo prazo". Neste momento, então, o conceito dado é complementado com a ênfase do processo e da construção necessários. [Kimball&Ross 02] contribui para o conceito acima, ampliando-o com a exigência de que os dados devem ser precisos e completos.

Uma perspectiva mais voltada a processo é resumida em [Kimball&Ross 02] como um conjunto de ferramentas e técnicas de projeto, que quando aplicadas

às necessidades específicas dos usuários e aos bancos de dados específicos permitirá que planejem e construam um data warehouse.

Interessante se observar as diferenciações estabelecidas entre os bancos de dados transacionais e o data warehouse. Assim, enquanto os usuários dos primeiros são os que **determinam** o rumo do negócio, registrando um dado por vez e repetindo as mesmas tarefas operacionais sucessivamente, os usuários dos segundos são os que **observam** o rumo que os negócios devem tomar, realizando pesquisas em milhares de dados e compactando-os em um conjunto de respostas para a obtenção do resultado requerido. Em adição, mudam constantemente os tipos de perguntas que fazem.

Os bancos de dados transacionais armazenam as informações das operações diárias do negócio ou organização, que são repetidas sucessiva e diariamente e que são pré-definidas, o que resulta em uma mudança constante do conjunto de dados armazenados. Por trabalhar com tabelas normalizadas, evitando a existência de redundância dos dados e preocupando-se apenas em registrar os instantâneos das operações, não armazena informações históricas por muito tempo, não exigindo, por outro lado, grande capacidade de armazenamento. O data warehouse, ao contrário, armazena dados analíticos, destinados à gerência e à tomada de decisões, disponíveis para consultas complexas que manipulam grande número de registros, exigindo, por questões de desempenho, a utilização de muitos índices. Os dados são armazenados dentro de uma perspectiva histórica, durante longos períodos de tempo – podem chegar a dezenas de anos – [Inmon&Hackthorn 97] e, portanto, demandam significativa capacidade de armazenamento e processamento.

Os dados tratados em um data warehouse podem apresentar duas origens: dos sistemas transacionais existentes, que podem estar implementados em sistemas de gerenciamento de bancos de dados ou legados de sistemas de arquivos, constituindo-se em dados primitivos; de fontes externas às

organizações, oriundos dos ambientes específicos em que as mesmas estão inseridas, algumas vezes obtidos de formulações subjetivas.

Para que se possa considerar um data warehouse, segundo [Kimball&Ross 02], são necessários os atendimentos a determinados requisitos:

a) O data warehouse deve fazer com que informações de uma empresa possam ser facilmente acessadas – o conteúdo do data warehouse deve ser legível, significativo e compreensível para o usuário da área de negócios. Estes devem poder combinar os dados de infinitas maneiras, repetindo um processo denominado "separação e combinação" (s*lice and dicing*). As ferramentas utilizadas para acesso aos dados devem ser simples e intuitivas e devem retornar resultados em pequenos intervalos de tempos;

b) O data warehouse deve apresentar as informações da empresa de modo consistente – os dados armazenados devem ser confiáveis. Devem ser obtidos das mais diversas fontes, filtrados, submetidos a um controle de qualidade e liberados para uso apenas quando houver garantia que estão precisos. Informações consistentes significam informações de alta qualidade.

c) O data warehouse deve ser adaptável e flexível a mudanças – as mudanças fazem parte da existência das organizações. Não mudam apenas aquelas que desapareceram ou que irão desaparecer. Assim, o data warehouse deve ser projetado para lidar com esta mudança inevitável. As mudanças em um data warehouse são bem vindas, mas não podem mudar ou danificar os dados ou aplicações existentes. As modificações no esquema do data warehouse deve ser alvo de um processo de avaliação de impacto cuidadoso;

d) O data warehouse deve ser um baluarte seguro que protege as informações – a história e a identidade da organização está armazenada em um data warehouse. Desta forma, o acesso a essas informações por pessoas indevidas pode trazer conseqüências desastrosas. Assim, o data warehouse deve implementar um método eficaz de controle de acesso a estas informações;

e) O data warehouse deve funcionar como a base para uma melhor tomada de decisões – o data warehouse deve conter os dados apropriados para dar suporte à tomada de decisões. O valor que pode ser dado ao data warehouse

resulta do impacto obtido de decisões tomadas a partir de evidências apontadas por ele;

f) Para que um data warehouse seja considerado de sucesso, a comunidade de negócio deve aceitá-lo – não importa a criação de uma solução elegante utilizando-se do estado arte dos produtos tecnológicos. Se o data warehouse não for aceito e utilizado pela comunidade usuária ele deve desaparecer. Isto acontece porque, diferente da comunidade usuária dos sistemas de informações operacionais que não tem opção senão a sua utilização, a do data warehouse pode simplesmente ignorá-lo e o principal fator determinante de seu sucesso é a simplicidade de uso.

Data warehouse é, portanto, um conjunto de ferramentas tecnológicas e métodos de trabalho que objetivam a construção de um sistema de informações estratégicas baseado em assuntos, que guarda a história e identidade da organização e que oferece respostas em pequeno intervalo de tempo a questionamentos não programáveis, através de interfaces simples, aplicados diretamente pelo usuário final.

Deve-se registrar, por oportuno, que o nosso conceito, assim como os demais, não registra a presença do usuário final no processo de modelagem e construção do data warehouse, o que estamos propondo neste trabalho.

2.4 – CARACTERÍSTICAS

Baseado na definição clássica encontrada em [Inmon&Hackthorn 97] pode-se identificar os itens que formam as características fundamentais de um data warehouse. A estas se pode acrescentar outras de natureza menos conceitual e mais voltada aos aspectos de implementação, todas relacionadas a seguir:

2.4.1 – Orientação a Assuntos

Segundo [Inmon&Hackthorn 97] a primeira característica notável do data warehouse é a sua organização em torno dos assuntos mais importantes da corporação (i.e. as entidades de mais alto nível da empresa), fazendo com que o projeto de um data warehouse seja "orientado a dados". Os assuntos são o conjunto de informações sobre determinada área estratégica da organização e este é um principal foco no processo de modelagem do data warehouse, o que o diferencia dos sistemas transacionais cujo foco são os processos e funções específicas das organizações.

Os processos e funções operacionais não fazem parte das preocupações do projetista de data warehouse. A este apenas interessa a modelagem de dados e o projeto do banco de dados. As diferenças entre a orientação a processos e a orientação a assuntos, tônica do data warehouse, são imediatamente identificáveis no conteúdo dos dados. Nos últimos, apenas estão armazenados os dados relevantes ao processo de tomada de decisões enquanto que nos primeiros os dados são detalhados de forma a satisfazer os requisitos de imediatismo do processamento funcional. O relacionamento entre os dados constitui-se também em um diferencial importante entre as duas formas de orientação no processo de modelagem de dados.

2.4.2 – Integração

Esta é a principal característica do Data Warehouse e que marca o seu caráter corporativo. Esta integração, de acordo com [Inmon&Hackthorn 97], aparece de diferentes maneiras na consistência das convenções de nomes, na consistência das variáveis de medidas, na consistência da codificação das estruturas, na consistência dos atributos físicos e assim por diante.

Durante a construção e implementação dos sistemas transacionais, cada um dos projetistas toma decisões individuais quanto à codificação, estrutura de chaves, características físicas, convenções de nomes, etc. Estas decisões afetam decisivamente a descrição dos dados armazenados em suas aplicações, que projetadas de forma isolada, contribuem para o estabelecimento de um cenário em que o conjunto de dados primitivos, sobre os quais são derivados os dados do data warehouse, apresentam representações diferentes para a mesma informação.

As duas principais diferenças geradas pelos projetistas de aplicação são: a codificação, que é a forma com que conteúdos de domínios pré-determinados são representados. Há o caso clássico da representação do gênero, mostrado na figura 2.1; e os atributos de medidas, que derivam do tratamento de dados numéricos segundo escalas de medidas diferentes. Em ambos os casos, os dados primitivos das aplicações precisam ser integrados e consistidos antes de serem carregados no data warehouse.

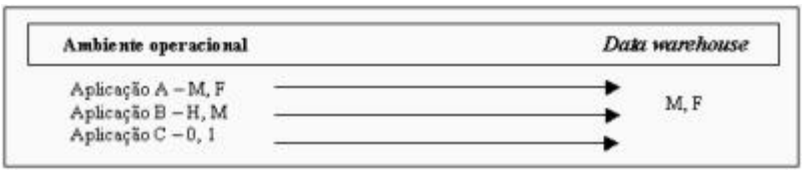

Figura 2.1 – Problemas de integração com o gênero

Fonte ***http://www.dwbrasil.com.br/html/artdw_carac.html***

2.4.3 – Variação no Tempo

De acordo com [Inmon&Hackthorn 97], "todos os dados no data warehouse são precisos em algum instante do tempo". Esta característica básica também apresenta diferenciais importantes quando comparada àquela dos sistemas operacionais. Nestes os dados são precisos no momento do seu acesso. No data warehouse os dados são armazenados sobre séries históricas que apresentam um conteúdo determinado – e preciso - em certo instante do tempo. A variância no tempo em um data warehouse se dá de três formas: a primeira é a em que os dados representam informações sobre um horizonte de tempo muito amplo – de cinco a dez anos, ao contrário dos sistemas operacionais que mantêm os dados do momento atual a até, no máximo, de sessenta a noventa dias passados, vez que não podem acumular grandes volumes de dados sob pena do comprometimento do desempenho do processamento; a segunda se dá quanto às estruturas básicas que, no data warehouse, contém obrigatoriamente, implícita ou explicitamente, o elemento de tempo como dia, semana ou mês, e este está, via de regra, concatenado à chave de acesso ao dado; e a terceira maneira é aquela em que, uma vez que os dados tenham sido armazenados corretamente no data warehouse, não podem ser atualizados.

2.4.4 – Não Volatilidade

Em um Data Warehouse os dados não sofrem alterações de natureza instantânea, registro a registro. As únicas operações possíveis são de carga inicial e acesso aos dados, do ponto de vista de consultas. Este realidade define um série de características que diferem o projeto do data warehouse daquele utilizado em sistemas transacionais. Isto significa que, nos primeiros, pode-se tomar certas liberdades no que tange a normalização e desnormalização de arquivos, visando a otimização do acesso aos dados. Em adição, há a simplicidade de operação, em que não há a exigência de se tratar

acesso concorrente, recuperação pós-falha, transações com garantias de manutenção de integridade e consistência e políticas de back-up.

Por outro lado, ao contrário do que se poderia supor em função dos sistemas transacionais serem as fontes de dados para os data warehouse, há pouca ou quase nenhuma redundância entre os dois. De fato, para que os dados sejam carregados no data warehouse passam por um processo de limpeza e transformação que os modificam completamente, seja por efeito das necessidades de integração ou em função de operações de síntese realizadas sobre os dados primitivos.

Um outro aspecto definidor da não volatilidade é o horizonte de tempo dos dados armazenados. Em um data warehouse este horizonte é bastante grande de forma que os dados já são velhos o suficiente – e consolidados – para sofrerem quaisquer modificações. Ao contrário, nos sistemas transacionais, os dados referem-se a instantâneos da realidade ocorrida naquele exato momento, que pode ainda sofrer alterações vez que ainda não está consolidada.

2.4.5 – Localização

O data warehouse pode ser construído de três formas diferentes, no que se refere à localização dos dados. Todas as alternativas têm os seus níveis de dificuldade de construção, manutenção e custo de funcionamento que serão detalhados a seguir:

a) Centralizados

Nesta alternativa, todos os dados obtidos no "*data staging*" [Kimball&Ross 02] são armazenados em um único repositório, de maneira que o poder de processamento é maximizado e o acesso aos dados é agilizado. Este é o tipo de armazenagem mais utilizada, não obstante as exigências de

armazenamento e processamento para o tratamento de um grande volume de dados integrado;

b) “Data Marts”

Neste caso os dados são armazenados por área de interesse. Assim, os dados podem ser distribuídos por servidores dedicados a cada área operacional da organização, de forma que o acesso aos dados e processamento ganha em desempenho, por não haver sobrecarga em um único servidor;

c) Nível de Detalhes

Neste modelo, os dados, assim como na anterior, também estão distribuídos. Porém seguem outras regras na determinação da distribuição: o nível de detalhe – ou granularidade – dos dados. Assim, podemos ter dados detalhados antigos e atuais, dados levemente resumidos, dados altamente resumidos e metadados. De acordo com [Inmon&Hackthorn 97], os dados detalhados atuais são aqueles que requerem maior preocupação por refletirem os acontecimentos mais recentes e de maior interesse, serem em grandes volumes por armazenarem dados de baixa granularidade e por serem armazenados em discos, meios de armazenamento que exigem uma gerência mais complexa.

Os dados detalhados antigos são acessados com baixa freqüência e, por esta razão e por serem volumosos, são armazenados em dispositivos de armazenamento de massa secundários de baixo custo, do tipo das fitas.

Os dados levemente resumidos são extraídos dos dados detalhados atuais e são sempre armazenados em disco. Este é o nível que submete o projetista do data warehouse às questões relacionadas com o espaço de tempo em que o resumo será feito e ao conteúdo destes dados resumidos, segundo [Inmon&Hackthorn 97].

Os dados altamente resumidos são compactos e de fácil acesso. Geralmente suportam a um grande número de acessos ao mesmo tempo. Podem estar implementados fora da estrutura tecnológica que forma o data warehouse, mas, no entanto, continuam fazendo parte do mesmo.

Os dados em um data warehouse podem mudar de nível por serem sintetizados, arquivados ou eliminados. O processo de sintetização corre da menor para a maior granularidade, atendendo ao agrupamento por datas ou por grupos e subgrupos. Cada nível de dados possui um horizonte de tempo definido para a sua permanência. Dentro desta permanência o dado pode modificar de nível sem que seja excluído do nível anterior. Quando a permanência é ultrapassada, no entanto, ocorre um processo denominado de envelhecimento, em que os dados são transportados para um meio de armazenamento alternativo, passando de detalhados atuais para detalhados antigos.

2.4.6 – Credibilidade dos Dados

Depreendendo o afirmado em [Kimball&Ross 02], além de se garantir um acesso simplificado aos dados, o data warehouse deve primar pela sua consistência. Desta forma, qualquer discrepância observada em seu conteúdo pode levar a um comprometimento de sua credibilidade e daí, indubitavelmente, ao processo de suporte à decisão de baixo nível, podendo acarretar em grandes prejuízos àorganização.

Para garantia da consistência, os procedimentos anteriores à carga inicial dos dados em data warehouse devem ser executados com a máxima acurácia, dentro das etapas de extração, transformação e carga denominados de "*data staging*" [Kimball&Ross 02]. A qualidade dos dados carregados pode ser medida a partir de três características: precisão, que é o grau de informações

que estão corretas; abrangência, que é o grau de dados requisitados e atendidos; e consistência, que é quanto os dados não se contradizem.

Característica da qualidade de dados	Descrição	Exemplo de medida
Precisão	Grau de informações que estão corretas.	Percentual de correção.
Abrangência	Grau de dados requisitados e atendidos.	Percentual atendimentos.
Consistência	Consistência dos dados / liberdade de contradição.	Percentual de condições satisfeitas.

Figura 2.2 –Características de qualidade dos dados

Fonte ***http://www.dwbrasil.com.br/html***

2.4.7 – Granularidade

Esta característica refere-se ao nível de detalhamento dos dados, ou o quanto estes estão resumidos. O volume de dados e o desempenho do acesso aos mesmos dependem desta característica. Assim, quanto maior o nível de detalhes – menor granularidade – maior é a necessidade de espaço para armazenamento e maior também será o custo de processamento para a sua recuperação. Por outro lado, consultas detalhadas apenas são possíveis neste nível.

Enquanto que os dados altamente detalhados e levemente resumidos servem para os Sistemas de Apoio à Decisão, aqueles altamente resumidos servem diretamente aos Sistemas de Informações Executivas por apresentarem informações estatísticas.

O balanceamento do nível de granularidade é um dos aspectos mais críticos no planejamento do data warehouse, pois sempre há uma demanda por eficiência no armazenamento e no acesso aos dados, bem como pela possibilidade de analisar dados em grande nível de detalhes.

2.5 – ARQUITETURAS FÍSICAS DO DATA WAREHOUSE

A arquitetura de um data warehouse pode ser abordada de diferentes ângulos: do armazenamento, origem e fluxo dos dados, das camadas e do funcional. Neste trabalho serão apresentadas várias abordagens, de acordo com autores e ângulos diferentes que, ao final, apresentam uma convergência conceitual clara.

Do ponto de vista estrutural, há um modelo de referência, mais geral, e modelos específicos que representam propostas mais práticas para a sua implementação

2.5.1 – Arquitetura Genérica (Modelo de Orr)

Esta proposta visa apenas sistematizar todos os papéis possíveis na construção do data warehouse, permitindo que as diferentes abordagens existentes no mercado possam se enquadrar na mesma. Foi proposta por [Orr 97] e atualizada permanentemente, e é formada pelas seguintes camadas, de acordo com a figura 2.3.

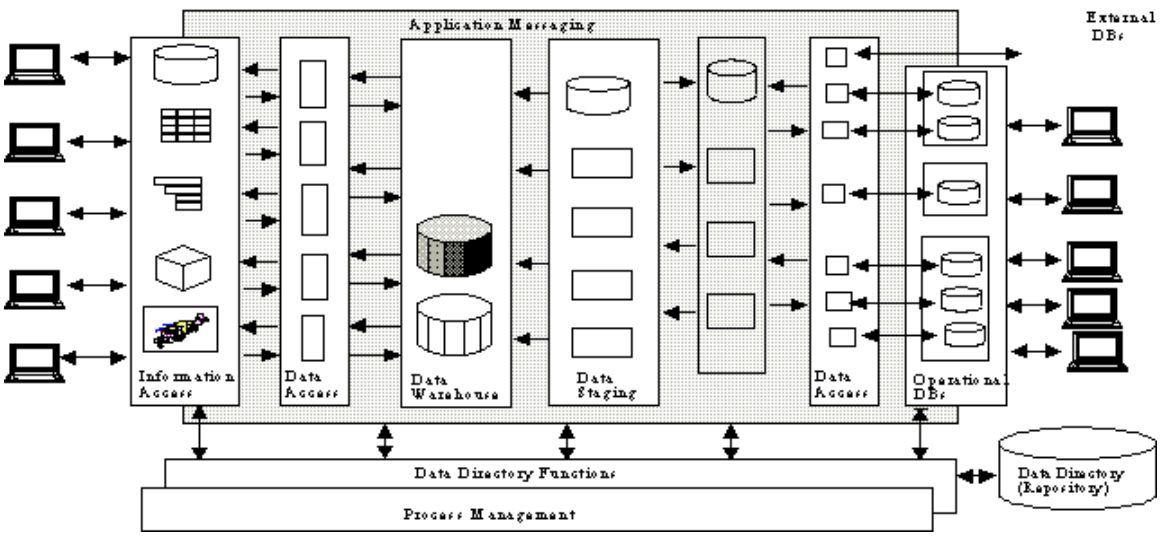

Figura 2.3 – Arquitetura genérica segundo Ken Orr. Fonte *[Orr 97]*

a) **Camada de Bancos de Dados Operacionais** : corresponde aos dados das bases de dados operacionais da organização junto com dados provenientes de outras fontes externas que serão tratados e integrados para compor o data warehouse;

b) **Camada de Acesso à Informação**: é a camada com a qual os usuários finais interagem. Representa as ferramentas que o usuário utiliza no dia a dia. Também envolve o hardware e software utilizados para obtenção de relatórios, planilhas, gráficos e outros. A cada dia surgem sistemas mais sofisticados para manipulação, análise e apresentação dos dados, incluindo-se ferramentas de "*data mining*" [Han & Kamber 00] [Hand et al 01] [Piatestky-Shaphiro et al 96] e visualização;

c) **Camada de Acesso aos Dados**: esta camada é responsável pela ligação entre as ferramentas de acesso à informação e os bancos de dados operacionais. Esta camada se comunica não só com diferentes SGBDs e sistemas de arquivos de um mesmo ambiente como também, idealmente, com outras fontes sob diferentes protocolos de comunicação, no que se chama acesso universal de dados;

d) **Camada de Metadados (Dicionário de Dados):** metadados são as informações sobre os dados mantidos pela empresa (comandos CREATE do SQL, informação em um diagrama E-R, dados em um dicionário de dados são exemplos de metadados). Para poder manter a funcionalidade de um ambiente de data warehouse é necessário ter disponível uma grande variedade de metadados, desde dados sobre as visões dos usuários até dados sobre os bancos de dados operacionais. Idealmente o usuário deve poder ter acesso aos dados de um data warehouse sem que tenha que saber onde residem estes dados ou a forma como estão armazenados;

e) **Camada de Gerenciamento de Processos**: a camada de gerenciamento de processos está envolvida com o controle das diversas tarefas a serem realizadas para construir e manter as informações do dicionário de dados e do data warehouse. Esta camada é responsável pelo gerenciamento dos processos que contribuem para manter o data warehouse atualizado e consistente;

f) **Camada de Transporte ou "*Middleware*":** esta camada gerencia o transporte de informações pelo ambiente de redes. É usada para isolar aplicações, operacionais ou informacionais, do formato real dos dados nas duas extremidades. Também inclui a coleta de mensagens e transações e se encarrega de entregá-las em locais e tempos determinados;

g) **Camada do DW:** O data warehouse propriamente dito corresponde aos dados usados para fins informacionais. Em alguns casos, data warehouse é simplesmente uma visão lógica ou virtual dos dados, podendo de fato não envolver o armazenamento destes dados. Em um data warehouse que exista fisicamente, cópias dos dados operacionais e externos são de fato armazenadas, de modo a prover fácil acesso e alta flexibilidade de manipulação.

2.5.2 – Arquitetura em Camadas (Wu e Buchmann)

A arquitetura de um data warehouse, segundo [Ming & Buchmann 97] é definida por quatro fatores: o volume de dados a ser armazenado, características dos dados, arranjo dos dados e o uso que será feito dos dados. Para cada um dos fatores há uma camada correspondente, de acordo com a Figura 2.4. e exposição a seguir:

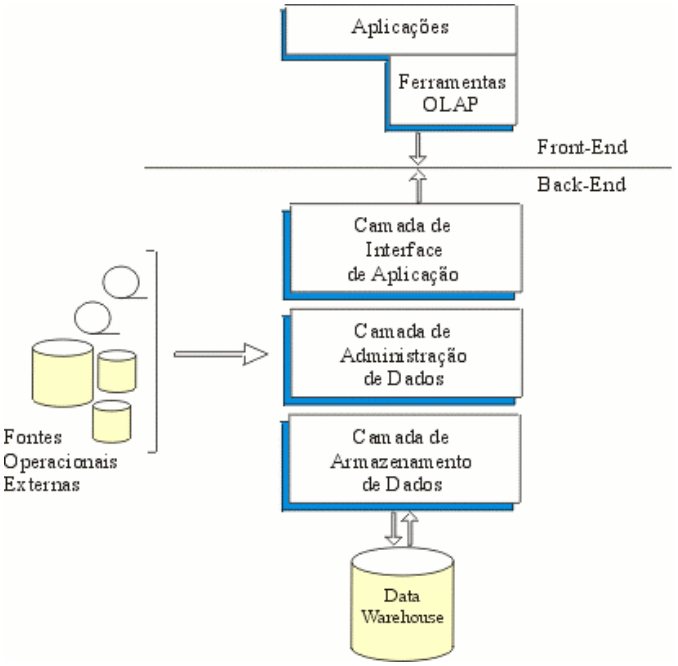

Figura 2.4 – Arquitetura em camadas segundo Wu e Buchmann. Fonte *[Ming & Buchmann 97]*

a) **Camada de Administração dos Dados**: associada ao volume de dados, define as formas eficientes para armazenamento e recuperação dos dados, construção de índices e "*data clustering*";

b) **Camada de Armazenamento dos Dados**: associada às características dos dados, define as estruturas lógicas para armazenamento dos dados, executa a transformação entre os modelos de dados externos e a estrutura lógica do data warehouse, executa os processos de "*data staging*", garante a característica de integração dos dados, responsável pelo processamento e otimização de consultas;

c) **Camada de Interface da Aplicação**: associada ao arranjo dos dados, executa o arranjo conceitual dos dados de acordo com as necessidades da aplicação, convertendo o modelo lógico do data warehouse nos modelos conceituais requeridos pelas aplicações;

d) **Camada de Apresentação**: associada ao uso dos dados, define a interface do usuário com as aplicações, comumente implementada através de ferramentas OLAP [Thomsen 97], consultas e relatórios.

2.5.3 – Arquitetura Funcional Básica

Uma arquitetura em camadas como a descrita acima tem muita utilidade na visualização das funcionalidades de cada camada do sistema. Entretanto, falta nesta arquitetura um melhor detalhamento dos componentes de um data warehouse bem como do ciclo de vida dos dados neste tipo de sistema. Através do detalhamento desses dois aspectos poderemos entender melhor o funcionamento geral de um ambiente de data warehouse. Por essas razões exibimos na Figura 2.5 uma Arquitetura Funcional Básica para data warehouse:

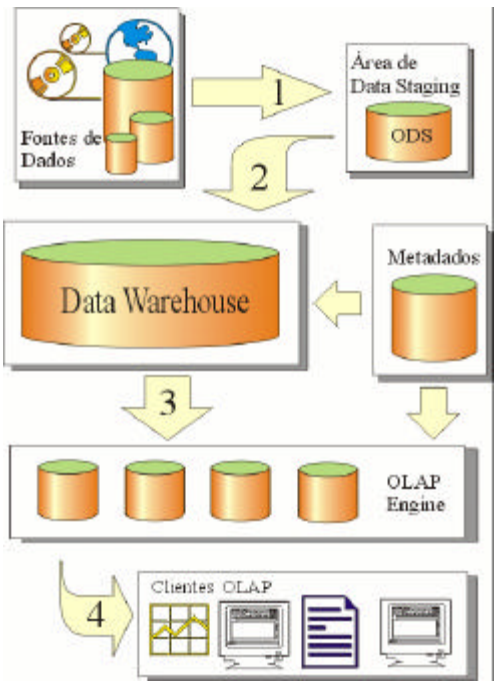

Figura 2.5 – Arquitetura funcional básica de um ambiente de data warehouse. Fonte *[Ming & Buchmann 97]*

2.5.3.1 – Componentes

a) Fontes de dados: é com os dados dessas fontes que o data warehouse é povoado. Diferentemente dos sistemas operacionais, onde as entradas de dados geralmente são feitas diretamente pelo usuário, a entrada de dados do data warehouse é feita a partir de dados dos sistemas operacionais e de fontes externas. As fontes de dados podem ser divididas em dois grupos: Fontes de Dados Internas, são os sistemas de automação do nível operacional. Os dados são provenientes de sistemas OLTP ou dos sistemas legados. Esses sistemas utilizam processamento transacional e controle de concorrência; Fontes de Dados Externas, são fontes que não fazem parte dos sistemas de informações da empresa. Os dados dessas fontes geralmente chegam através de redes públicas de comunicação. A Internet, por exemplo, é uma grande fonte de dados externa.

b) **Área de "*Data Staging*":** é uma área de armazenamento intermediária entre as Fontes de Dados e o data warehouse. Nesta área são feitos o tratamento, a integração e o controle de qualidade sobre os dados para que eles possam ser transferidos para o DW. Segundo [Kimball&Ross 02], os dados nesta área geralmente estão em terceira forma normal e são não-históricos. Isto porque esta área não tem nenhum propósito de fornecer serviços de consulta e apresentação dos dados para análise. O repositório desta área também é conhecido como ODS (*"Operational Data Store"*) [Inmon&Hackthorn 97].

Não necessariamente existe apenas um repositório para a Área de *"Data Staging"*, porém, os vários que venham a existir devem ser integrados e homogêneos.

c) **Data Warehouse**: é o repositório central, integrado, histórico e orientado a assuntos. Dependendo da arquitetura física adotada para o ambiente de data warehouse, existem variações de modelos para a sua construção. Ele pode, por exemplo, ser construído em um banco de dados multidimensional com cubos multidimensionais, ou então em um sistema gerenciador de banco de dados relacional com esquemas em estrela ou na terceira forma normal.

d) "***OLAP Engine"*:** este componente está presente numa arquitetura baseada em ROLAP. Nela, dados do data warehouse são extraídos para repositórios menores chamados Data Marts, que são departamentais, menores em tamanho e orientados aos assuntos específicos do departamento. Geralmente são baseados em bancos de dados multidimensionais (podendo também ser baseados em esquemas em estrela). São hospedados em servidores de apresentação, de onde os dados são diretamente manipulados por aplicações OLAP.

e) **Metadados**: os metadados têm um papel fundamental nos data warehouses. Os metadados basicamente são informações sobre o que são e onde estão as coisas no data warehouse. Há basicamente três classes de Metadados: **Operacionais**, que guardam informações estruturais sobre os dados no banco de dados, bem como mantêm informações estatísticas e de auditoria; **Administrativos**, mantêm informações necessárias ao correto uso do data warehouse. Alguns aspectos abrangidos são: descrições sobre as

fontes, dimensões, hierarquias, agregados, ferramentas de *"back-end"* e *"front-end"*, segurança, controle de acesso, consultas pré-definidas, histórico da extração, carga, transformação e controle de qualidade e o modelo de dados; **Corporativos**, fazem a ponte entre os conceitos lógicos e físicos do data warehouse e os conceitos do negócio, facilitando a vida do usuário executivo.

f) **Clientes OLAP**: generalizamos o termo clientes OLAP para caracterizar todas as aplicações que acessam os dados multidimensionais para análise. São as ferramentas utilizadas pelo corpo administrativo para análise gerencial. Estas aplicações incluem desde ferramentas OLAP complexas e poderosas até planilhas eletrônicas e geradores de relatórios. Entretanto, como essas últimas precisam "ter" a visão multidimensional da informação, elas precisam ter funcionalidades de ferramentas OLAP puras.

2.5.3.2 - Fluxo de Dados

Num ambiente de data warehouse os dados seguem um fluxo determinado. Inicialmente esses dados precisam ser (1) integrados de várias fontes para posteriormente serem (2) estruturados, armazenados e (3) distribuídos para (4) análise.

Cada seta direcionada na arquitetura da Figura 2.5 é um estágio do fluxo de dados no data warehouse que agrega diversas atividades:

a) Extração das várias fontes, armazenamento e integração na Área de *"Data Staging"*: uma vez na Área de *"Data Staging"*, os dados passam por diversas etapas de transformação, as quais incluem: integração e geração de novas chaves; adição do tempo; checagem da integridade referencial para tabelas de fato e tabelas de dimensão; desnormalização ou normalização; conversão de tipos de atributos; cálculos, derivações e alocações; construção de Agregados e tratamento de Valores Nulos;

b) Carga dos dados no data warehouse: uma vez feita toda a transformação na Área de *"Data Staging"* os dados são carregados no data warehouse. As principais atividades do processo de carga são: particionamento de tabelas; reconstrução de Índices; controle de qualidade e publicação dos dados;

c) A partir do data warehouse corporativo os dados são carregados nos *"data marts"*: o uso de "*data marts"* melhora o desempenho do sistema à medida que as consultas analíticas são divididas entre vários data warehouses departamentais, evitando assim a sobrecarga de consultas sobre o data warehouse corporativo. As principais atividades envolvidas nesta carga são: separação dos dados por áreas funcionais e transformação dos dados do modelo lógico corporativo para o modelo lógico departamental;

d) Neste ponto o fluxo de dados é basicamente composto de conjuntos-resposta para os Clientes OLAP. As atividades incluem processamento e otimização de consultas, geração de relatórios e controle de acesso e segurança.

A manutenção de um data warehouse também inclui o processo de *"refresh"*, que consiste em propagar para o data warehouse as atualizações feitas nos dados operacionais. Geralmente o processo de *refresh* é feito em *"batch"*, quando o sistema é desativado para consulta por algumas horas, e é periódico. Essa periodicidade define a granularidade, ou seja, o nível de detalhe em que os dados são armazenados no data warehouse. É recomendável que ele tenha granularidade mais fina possível, o nível atômico dos dados.

2.5.4 – Arquiteturas Físicas

Há várias arquiteturas físicas propostas para data warehouse, mas discutiremos aqui as duas principais vertentes: arquitetura em duas camadas (*"two-tier architecture"*) e arquitetura em três camadas (*"three-tier architecture"*).

2.5.4.1 - Arquitetura em Duas Camadas

A arquitetura em duas camadas pode ser implementada considerando-se o repositório de dados informacionais centralizados através de um data warehouse corporativo ou através de uma federação de *"data marts"* que departamentalizariam a orientação a assuntos, na forma que se segue:

a) OLAP Multidimensional

Nessa arquitetura o banco de dados multidimensional agrega as funcionalidades de todas as camadas lógicas definidas na Seção 2.5.2. As

consultas analíticas são executadas diretamente no banco de dados multidimensional. Sumários para todas as dimensões são computados para ganho de performance. Os dados no banco de dados multidimensional são carregados no menor nível de granularidade. Os principais problemas dessa arquitetura são o limite do volume de dados que o banco de dados multidimensional pode armazenar e a escalabilidade. A adição de novas dimensões ao cubo multidimensional acarreta, invariavelmente, a reconstrução total da estrutura física e dos índices.

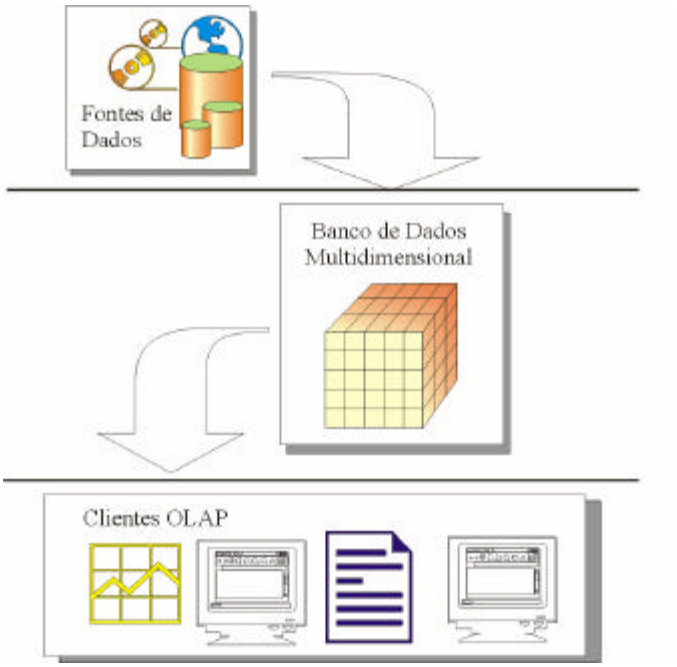

Figura 2.6 - Arquitetura OLAP multidimensional. Fonte *[Ming & Buchmann 97]*

b) Data Warehouse Federativo

"O data warehouse nada mais é do que a união de todos os *"Data Marts"* constituintes" [Kimball&Ross 02]. Esta arquitetura tem a enorme vantagem de possibilitar o desenvolvimento incremental (*"bottom-up"*) possibilitando resultados práticos em menor tempo de desenvolvimento.

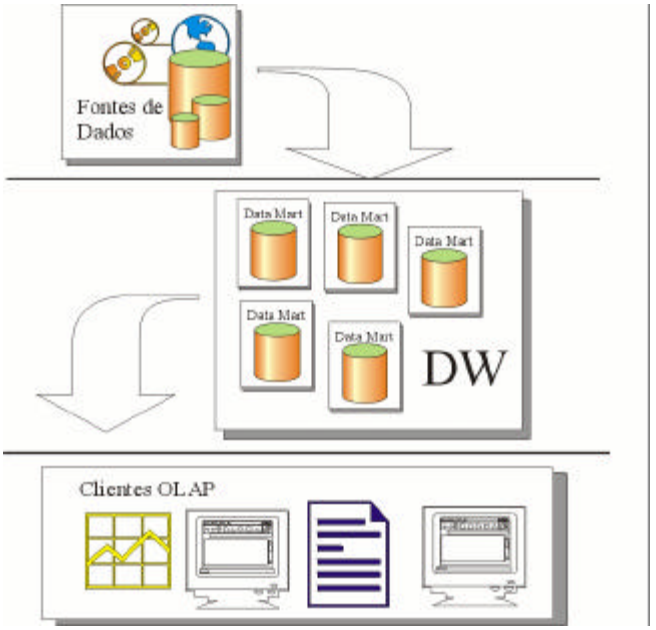

Figura 2.7 - Arquitetura de data warehouses federativos. Fonte *[Ming & Buchmann 97]*

2.5.4.2 - Arquitetura em Três Camadas

O objetivo principal dessa arquitetura é evitar a sobrecarga das atividades de análise sobre o data warehouse corporativo. Essa responsabilidade é deixada para um nível intermediário: os *"Data Marts"*.

Alguns autores defendem que o data warehouse corporativo seja projetado em terceira forma normal com a adição de atributos temporais para manutenção do histórico [Bliujute et al 98]. Os *"Data Marts"* seriam então projetados com estruturas multidimensionais como o cubo multidimensional ou o esquema em estrela.

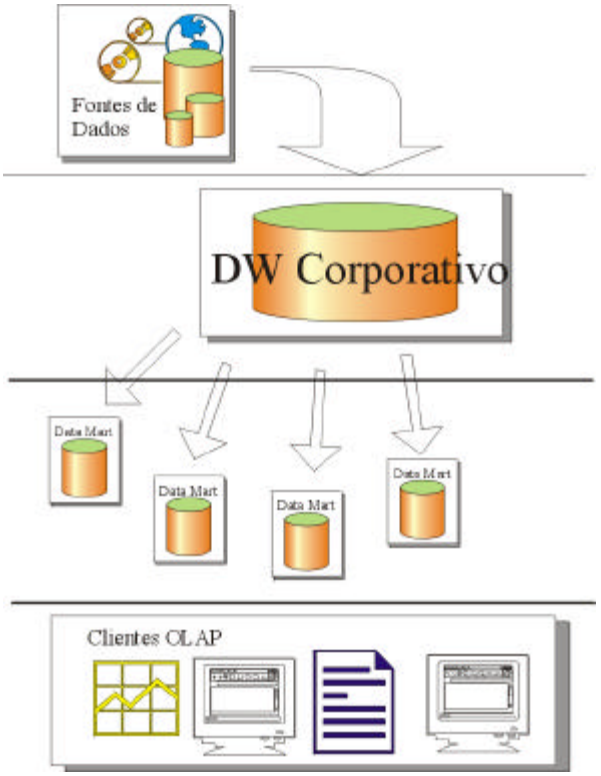

Figura 2.8 – Arquitetura em três camadas. Fonte *[Ming & Buchmann 97]*

2.6 – OS METADADOS

Os metadados são fundamentais para o funcionamento integrado dos vários módulos que compõem a solução de um data warehouse. De uma forma simplificada, [Harrison 98] define que metadados são “dados a respeito de dados”. Os metadados são utilizados no modelo proposto neste trabalho com a denominação de “atributos de sistema”. Estes fundamentalmente descrevem as principais características e comportamento dos dados armazenados no data warehouse, especialmente no que se refere a sua estruturação, como também se propõem a oferecer subsídios para as etapas anteriores à do armazenamento, que são conhecidas como ETL (extração, transformação e carga) e que não são foco do nosso trabalho.

Em [Inmon & Welch 99] encontramos que a infraestrutura de metadados que um gerente de metadados deve administrar deve contemplar:

a) o reconhecimento da necessidade de metadados comuns aos diferentes ambientes e permitir que metadados sejam compartilhados pelos diferentes componentes do ambiente; e,

b) reconhecimento da necessidade de autonomia local em cada ambiente, permitindo que os metadados sejam gerenciados localmente sem interferência ou consideração de outros usuários de fora do ambiente imediato.

Estes dois propósitos – compartilhamento e autonomia – podem parecer contraditórios, mas devem ser alcançados para que se obtenha sucesso em um projeto de data warehouse.

2.6.1 – Uma Abordagem Unificada à Infra-estrutura de Metadados

De acordo com [Inmon & Welch 99], os componentes de um ambiente projetado para data warehouse são o ambiente operacional, a camada de transformação, o ODS (*"operational data store"* – tentativa de integração de dados de sistemas legados), dados detalhados atuais do data warehouse e os dados levemente resumidos ou em *"data marts"* ou através de ferramentas OLAP, conforme se vê na Figura 2.9.

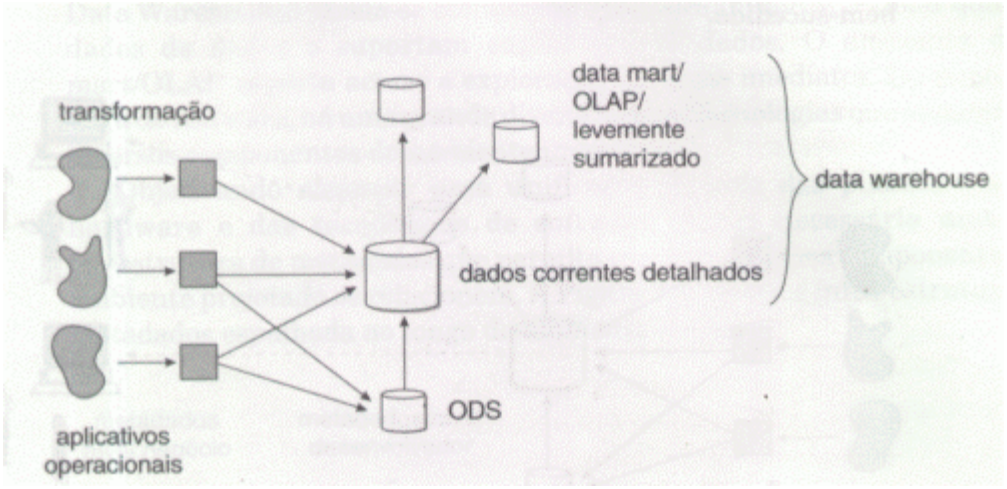

Figura 2.9 – O Ambiente Projetado. Fonte. [Inmon & Welch 99], página 77.

O ambiente projetado tem diferentes componentes que servem a diferentes propósitos e possuem diferentes usuários no ambiente de sistemas de informação. De forma semelhante, apresentam fundações tecnológicas de hardware e software diversas. Assim, objetivando alcançar uma unificação através das plataformas de hardware e software, faz-se necessária uma infra-estrutura de metadados que permita que os diferentes componentes da ambiente projetado se relacionem.

2.6.2 – A Composição dos Metadados

Na Figura 2.10 podemos observar que cada componente do ambiente projetado possui o seu próprio armazém exclusivo e privado de metadados – que devem ser compartilhados – o que está explicitado no Modelo de Orr, [Orr 97].

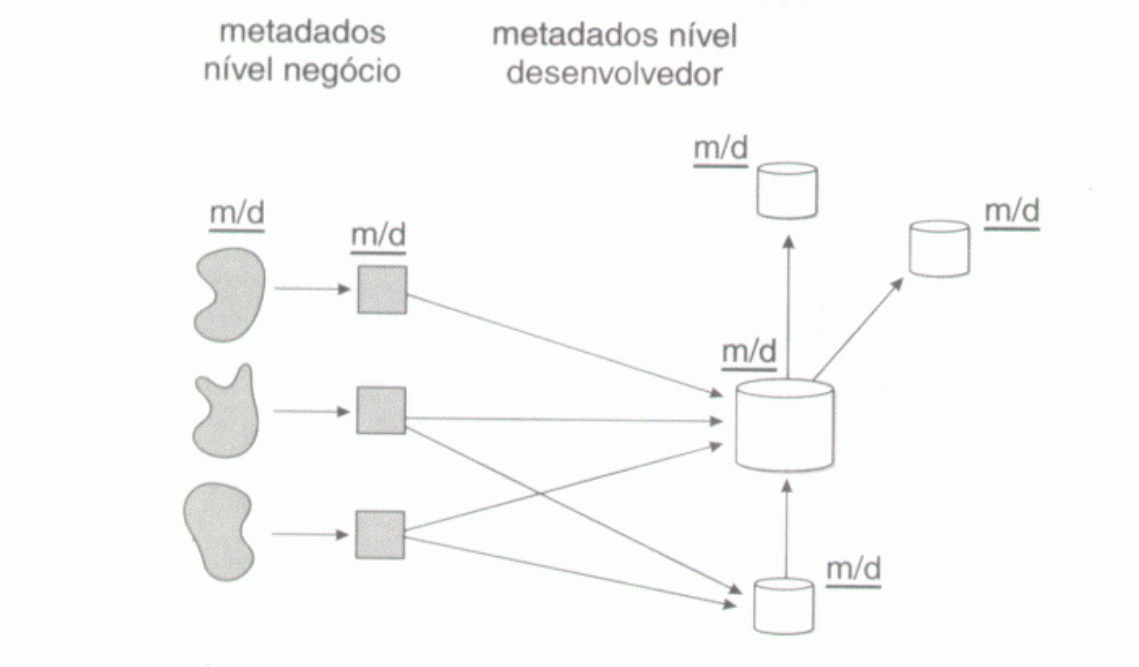

Figura 2.10 – Os Diferentes Tipos de Metadados. Fonte [Inmon & Welch 99] página 79.

Assim, os metadados que residem em um componente são bem diferentes dos metadados que residem em outro componente. Normalmente, os metadados que residem em cada nível contêm informações, tais como:

a) Ambiente Operacional
Especificações de índices
Características físicas de dados
Especificação de blocos de controle
Especificação de programação;

b) Camada de Integração e Transformação
Especificação origem-para-destino
Tópicos de conversão
Especificações de agrupamento
Arranjos de ressequenciamento;

c) Camada ODS (Operational Data Store)
Identificação origem-para-destino
Freqüência de atualização;

d) Nível corrente de detalhe no data warehouse
Identificação origem-para-destino
Informações de versão
Métricas
Programação de atualização
Referência cruzada entre negócios-tecnologia
Referência cruzada de *"aliases"*;

e) Nível de *"Data Mart"*
Identificação origem-para-destino
Estrutura física
Indexação
Referência cruzada entre negócios-tecnologia;

f) Nível de desenvolvimento
"Lay-outs" da estrutura física
Referência cruzada entre o modelo de dados-design do banco de dados
Especificação origem-destino;

g) Nível de negócio
Modelo de dados
Sistema de identificação de registro
Agrupamento de tabelas por área-assunto
Referência cruzada entre planejamento estratégico-modelo de dados.

2.7 – ETAPAS DO DESENVOLVIMENTO DO DATA WAREHOUSE

Ainda hoje é difícil apontar uma metodologia consolidada e amplamente aceita para o desenvolvimento de data warehouses. Neste trabalho relacionamos o guia funcional do ciclo de vida dimensional do negócio proposto por [Kimball&Ross 02], visto que nos parece uma abordagem advinda de experiência mais focada nos processos de projeto e modelagem.

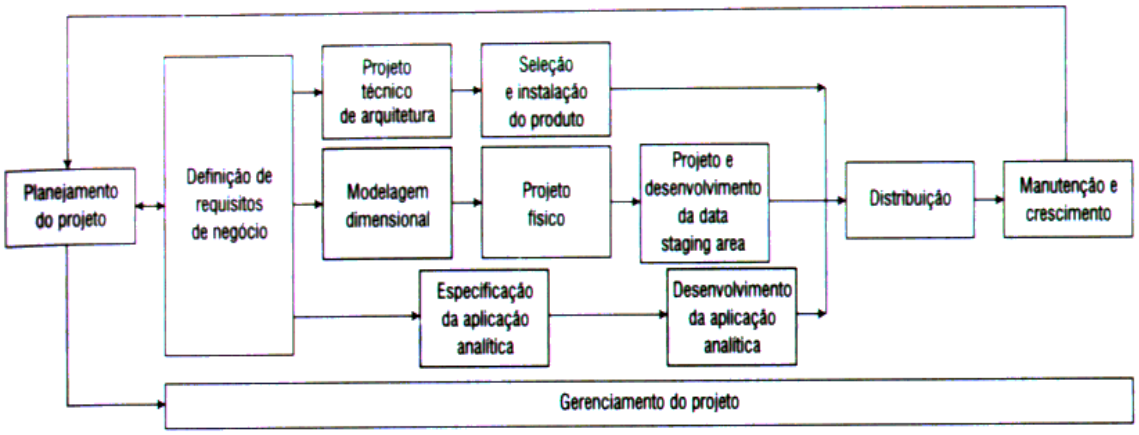

Figura 2.11.- Diagrama do Ciclo de Vida Dimensional do Negócio

Fonte [Kimball&Ross 02] página 381.

O diagrama apresentado na Figura 2.11 encapsula as atividades principais do ciclo de vida dimensional do negócio. Ele ilustra a seqüência, a dependência e a concorrência (simultaneidade) entre as tarefas do ciclo.

O ciclo de vida do data warehouse começa com o planejamento do processo. Neste módulo avalia-se a preparação da iniciativa de um data warehouse da empresa, estabelecendo o escopo e a justificativa preliminares, a obtenção de recursos e o lançamento do projeto.

A segunda tarefa se concentra na definição dos requisitos do negócio. O alinhamento do data warehouse com os requisitos do negócio é absolutamente crucial. As melhores tecnologias não salvarão uma data warehouse que não se

consiga se concentrar no negócio.Os projetistas do data warehouse devem entender as necessidades do negócio e mapeá-las em considerações de projeto. Os usuários de negócio e os seus requisitos têm um impacto sobre quase toda a decisão de projeto e implementação tomada no curso de um projeto de data warehouse.

O projeto técnico da arquitetura estabelece a estrutura geral para suportar a integração de diferentes tecnologias. Utilizando os recursos identificados no projeto de arquitetura como uma lista de compras, avalia-se e seleciona-se os produtos específicos.

Os requisitos de negócio evoluem também para a tradução dos mesmos em um modelo dimensional. O modelo dimensional é então transformado em estrutura física. Concentra-se então nas estratégias de ajuste, como agregação, indexação e particionamento, durante as atividades físicas do projeto. Em seguida os processos de extração, transformação e carga (ETL) de dados são projetados e desenvolvidos.

O conjunto final de tarefas gerado pela definição de requisitos de negócio é o projeto e desenvolvimento de aplicações analíticas.

Há a necessidade de se reunir a tecnologia, os dados e as aplicações analíticas a uma alta carga de treinamento e suporte para uma melhor distribuição dos efeitos do data warehouse. Em adição, a manutenção do data warehouse irá garantir a existência de dados sempre atualizados.O crescimento do data warehouse, com a inclusão de novos assuntos, retornam o processo novamente ao início do ciclo de vida dimensional do negócio.

2.8 – MODELAGEM DIMENSIONAL

A modelagem dimensional encerra uma perspectiva diferente para o tratamento dos dados das organizações quando comparada à modelagem utilizada para

os sistemas OLTP. Assim, neste caso o que interessa no projeto não são as entidades, relacionamentos, decomposições funcionais e análise de transição de estados, mas os fatos, dimensões e hierarquias para tratar de dados numéricos, como valores, contadores, pesos e ocorrências [Unicamp 98].

A modelagem dimensional permite a conceituação do negócio como um conjunto de valores ou medidas descritas através de várias perspectivas do negócio em questão. A modelagem dimensional apresenta os dados como uma matriz na qual cada dimensão é um tema ou assunto do negócio que será objeto da análise e o tempo é sempre uma das dimensões consideradas. É uma técnica particularmente útil para inspeção, sumarização e arranjo de dados para facilitar a sua análise.

Na modelagem de sistemas para OLTP as entidades de cada ramo do negócio são fragmentadas em tabelas de maneira a se garantir, através da normalização, a eliminação de redundâncias e falta de sincronismo entre os diversos dados armazenados o que acarretaria problemas de integridade levando a uma inevitável perda de credibilidade. Então, observa-se pela experiência e através da literatura pesquisada que a abordagem de modelagem dada aos sistemas transacionais atendem, com preponderância, os aspectos operacionais do armazenamento e atualização dos dados. O preço pago por esta atenção – normalização – é que os sistemas de produção (OLTP) são difíceis de serem consultados [Kimball et al 98]. Para que as consultas sejam efetuadas faz-se necessário que sejam programados diversas estruturas de extração e relatórios, em geral estáticos, que limitam as possibilidades do usuário tratar os dados armazenados sob diversos formas ou hipóteses. Assim, a aplicação de consultas não programadas sobre a base de dados torna-se impossibilitada o que compromete a agregação de valor aos dados armazenados. Há que se considerar, por oportuno, que a normalização leva a um “overhead” no processamento das consultas em face do número excessivo de junções que devem ser realizadas sobre as tabelas para que seja possível uma combinação de seus dados [Kimball&Ross 02] [Red Brick 95].

Ao contrário do processamento operacional, que trata dos dados atuais da empresa, a análise gerencial necessita também de dados históricos. Gerentes analisam seus negócios comparando desempenhos atuais com desempenhos em períodos anteriores [Kimball&Ross 02]. Bancos de dados OLTP geralmente não implementam nenhum tipo de suporte temporal, salvo poucas exceções. Eles refletem o estado atual do negócio. Entretanto, data warehouses são, por princípio, históricos, isto é, guardam informações atuais e do passado. Na modelagem dimensional estas informações são armazenadas nas denominadas tabelas de fatos (*"fact tables"*) que armazenam as séries históricas sobre o negócio [Kimball&Ross 02] [Kimball et al 98] [Poe et al 98] e que são acessadas através das denominadas tabelas de dimensões (*"dimension tables"*) que implementam as diferentes perspectivas do negócio sobre o dado armazenado [Kimball&Ross 02].

Fundamental observar que a compreensão do arranjo entre os fatos e as dimensões deve ser natural do ponto de vista do usuário final, a fim de que este possa identificar no modelo formalizado a representação fidedigna de seu negócio. Desta forma, compreendendo a modelagem realizada, o usuário final pode aplicar sobre este as consultas que desejar com a garantia que estas devam trazer resultados consistentes.

A modelagem dimensional, segundo [Kimball&Ross 02], apresenta as suas origens em um projeto de pesquisa em conjunto, realizado pela *General Mills* e *Dartmouth University* na década de 60, que teve seguimento com as contribuições de vários agentes, especialmente por Ralph Kimball, que por seu turno em [Kimball&Ross 02] nega que tenha sido o único criador desta tecnologia. O modelo dimensional mais aceito e utilizado é denominado de **esquema de junção em estrela** [Kimball&Ross 02] [Red Brick 95] e será abordado no tópico seguinte, com as suas variações.

2.9 – MODELAGEM EM ESTRELA (ESQUEMA DE JUNÇÃO EM ESTRELA)

O esquema de junção em estrela é formato pelas tabelas de fatos (*"fact tables"*) e tabelas de dimensões (*"dimension tables"*) que se relacionam com a cardinalidade de muitos-para-muitos, sendo que as tabelas de fatos se situam no centro do esquema, daí a sua denominação como estrela. As tabelas de fatos representam as medidas numéricas associadas a momentos de interseção no tempo entre as dimensões, enquanto as tabelas dimensões representam processos de negócios relacionados a determinado fato e os seus atributos têm natureza descritiva [Kimball&Ross 02].

As tabelas dimensão tendem a ser desnormalizadas para que a compreensão do esquema representativo do negócio por parte do usuário final seja simples e familiar e para que as operações de junções necessárias para obtenção de informações sejam minimizadas de forma a maximizar o processamento.

Um exemplo de esquema em estrela para vendas no varejo pode ser visualizado na figura 2.12 [Kimball&Ross 02]. Neste caso observa-se que se pode acessar dados relacionados às vendas através de loja, produto, promoção ou o tempo, sendo que esta última dimensão, por princípio, deve sempre estar presente na modelagem dimensional [Inmon&Hackthorn 97].

A primeira coisa que pode ser observada em um esquema dimensional em estrela é a sua simetria e simplicidade [Kimball&Ross 02]. Os usuários finais tiram proveito diretamente desta simplicidade, pois assim fica mais fácil percorrer e compreender os dados. Esta simplicidade também proporciona vantagens para o desempenho já que otimizadores de bancos de dados podem processar estes esquemas de formas mais simples e com menos junções, as atacando entre as tabelas dimensão, com as restrições estabelecidas pelo usuário, e com o resultado obtido realizar um único produto cartesiano com a tabela de fatos.

Não há dúvidas quanto às vantagens alcançadas com este modelo. No entanto, é de se questionar o que aconteceria se os números de tabelas de fatos crescessem exponencialmente face à própria complexidade do negócio e as tabelas dimensão passassem a se relacionar com múltiplas tabelas de fatos, formando uma constelação dimensional. Esta é uma dúvida que perduraria mesmo que se considerasse que o usuário final compreende muito bem o seu negócio.

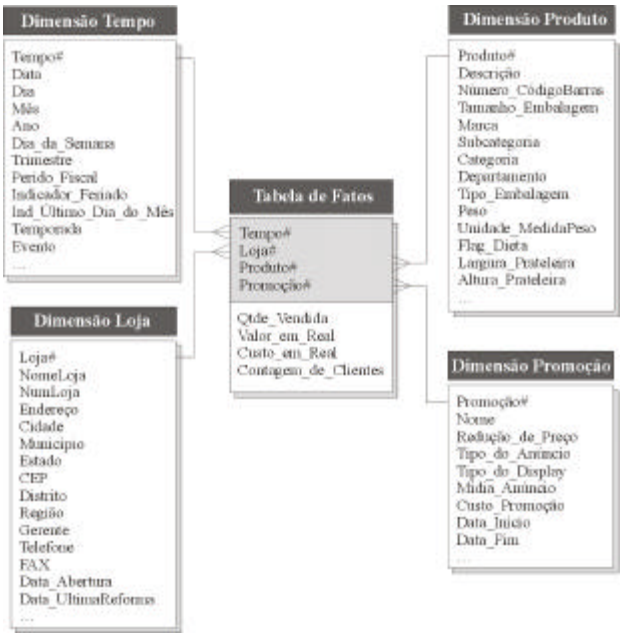

Figura 2.12 – Esquema em estrela para vendas no varejo. *Fonte[Kimball&Ross 02].*

Esta é uma das preocupações que motiva este trabalho: propor um esquema para modelagem dimensional cuja topologia fosse largamente conhecida e cuja manipulação já fosse habitual para o usuário final, inclusive por sua utilização em outras áreas do conhecimento. Ao mesmo tempo, que apresentasse um conjunto de vínculos naturais que orientariam o usuário final no caminhamento dentro das dimensões e fatos que compõem o seu negócio.

Um outro aspecto a ser considerado são as alterações possíveis na estrutura dimensional. Segundo [Kimball&Ross 02] o esquema em estrela oferece esta

adaptabilidade em razão da equivalência entre as dimensões e pelo fato das mesmas serem pontos de entrada simétricos para a tabela de fatos. Ressalva, no entanto, que "não vamos querer ajustar os nossos esquemas simplesmente porque usuários da área de negócios sugeriram novas maneiras para analisar a empresa". Esta abordagem pode causar estranheza vez que o data warehouse deve procurar como uma de suas metas a usabilidade pelo usuário final, ainda de acordo com [Kimball&Ross 02]. As ressalvas impostas ao comportamento dinâmico do modelo dimensional devem, em nossa observação, ser decorrentes das complexidades envolvidas no tratamento corporativo de todo um negócio e a sua representação, mesmo em um modelo dimensional em estrela. No entanto, se nos ativermos à abrangência dos *"data marts"*, talvez possamos proporcionar maior flexibilidade na alteração de esquemas, manipulados diretamente pelo usuário final. Esta questão será abordada no Capítulo 4 e será apresentada a sua implementação através de um protótipo.

A seguir, detalharemos os aspectos relacionados às tabelas de fatos e dimensões e como funciona a hierarquia de dimensões.

2.9.1 – Tabelas de Fatos

A tabela de fatos é a principal tabela de um modelo dimensional em que as medições numéricas de desempenho são armazenadas. Na figura 2.12 a tabela de fatos é a que apresenta as medições numéricas de Quantidade Vendida, Valor em Real, Custo em Real e Contagem de Clientes. Estes fatos representam a interseção das dimensões de Produto, Promoção e Loja em determinado momento do Tempo.

Os fatos em um modelo estrela podem ser **aditivos**, **semi-aditivos** ou **não-aditivos** [Kimball&Ross 02]. Os fatos aditivos são aqueles que podem ser somados em relação a todas as dimensões. Na tabela de fatos da Figura 2.12

os atributos numéricos Qtde_Vendida, Valor_em_Real e Custo_em_Real são **aditivos** pois podem ser somados em todas as dimensões. Assim, podemos obter a soma de Qtde_Vendida em determinado intervalo de Tempo, de um determinado conjunto de Produtos ou de Promoções ou mesmo de Lojas. O mesmo acontece para Valor_em_Real e Custo_em_Real. No entanto, o atributo Contagem_de_Clientes não pode ser somado em relação à dimensão Produto. Isto se dá porque não se pode precisar a quantidade de clientes que compraram mais de um produto, de forma que o somatório de suas medições certamente irá levar a um resultado equivocado. Este tipo de atributo da tabelas de fatos é denominado de **semi-aditivo**. Os fatos **não-aditivos** não podem ser somados em relação a nenhuma das dimensões. O que se pode fazer com estes é a realização de contagens e médias ou mesmo a sua simples listagem.

O registro de eventos em um data warehouse pode ser realizado através de diversos níveis de instantâneos ("*snapshots"*) da realidade, do mais detalhado ao mais resumido, o que é denominado de **granularidade** [Inmon&Hackthorn 97] [Kimball&Ross 02] [Kimball et al 98] [Poe et al 98]. No exemplo da Figura 2.12, o nível de granularidade do data warehouse é o dia. Neste caso, a depender dos resultados esperados das consultas aplicadas, poder-se-ia diminuir ainda mais a granularidade através do registro de cada uma das operações de venda ou, ao contrário, aumentar o tamanho do grão registrando apenas os resultados mensais das vendas. O nível de granularidade da tabela de fatos, segundo recomendação de [Inmon&Hackthorn 97], deve apresentar dados "expressos no nível de 'menor grão' de cada dimensão" porque as consultas de análise "precisam se aprofundar no banco de dados de maneira muito precisa". Em adição, [Kimball&Ross 02] afirma que os dados atômicos apresentam maior dimensionalidade e devem ser a base para o projeto da tabela de fatos para suportar demandas específicas de usuários comerciais em que eles propõem consultas inesperadas.

As tabelas de fatos possuem uma ou mais chaves estrangeiras que são as chaves primárias das tabelas de dimensão associadas aos fatos. Na Figura 2.12 as chaves estrangeiras da tabela de fatos são tempo#, loja#, produto# e promoção#. Quando todas as chaves da tabela de fato correspondem às respectivas chaves primárias corretamente nas tabelas de dimensão correspondentes, dizemos que as tabelas satisfazem à **integridade referencial** [Kimball&Ross 02].

Via de regra, a tabela de fatos apresenta uma chave primária obtida do subconjunto das chaves estrangeiras obtidas das tabelas de dimensões, que é denominada de **chave composta** ou **concatenada**. Toda a tabela de fatos em um modelo dimensional em estrela apresenta chave composta [Kimball&Ross 02].

2.9.2 – Tabelas de Dimensões

As dimensões são as perspectivas diferentes do negócio aplicadas sobre a tabela de fatos. Elas contêm os descritivos textuais da empresa e são definidas por suas chaves primárias designadas na Figura 2.12 pela notação # em seqüência àdenominação do atributo. Estas chaves funcionam com base na integridade referencial com as tabelas de fato a que estão associadas. As chaves primárias das dimensões são formadas por um único atributo. Kimball Kimball 98 a] Kimball 98 b] recomenda que essas chaves sejam artificiais (*"surrogate keys"*), uma generalização das chaves naturais dos sistemas de produção.

As tabelas de dimensão são fundamentais para o data warehouse. Elas são a fonte de todas as restrições realizadas nos procedimentos de consultas e contribuem decisivamente para a compreensão e utilidade do data warehouse [Kimball&Ross 02]. Desta forma, recomenda-se que a terminologia utilizada na

designação dos atributos dimensionais seja a mais aderente ao negócio que está sendo modelado, evitando-se as abreviações criptográficas.

2.9.2.1 – Hierarquia de Dimensões

É possível que dentro de uma dimensão haja atributos que se relacionem com a cardinalidade de 1:N. Neste caso a dimensão pode ser decomposta em outras de menor granularidade constituindo-se o que se denomina de hierarquia de dimensões [Kimball&Ross 02] [Kimball et al 98] [Poe et al 98] [Thomsen 97]. Na Figura 2.13 apresentamos dois exemplos da hierarquia de dimensões.

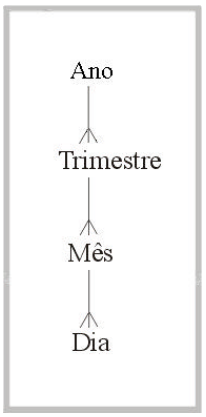

Figura 2.13 (a) Hierarquia na Dimensão Tempo

Figura 2.13 (b) – Hierarquia da Dimensão Produto

A organização de dimensões em hierarquias é particularmente interessante para permitir operações de *"drill-down"* e *"roll-up"* [4] [Inmon&Hackthorn 97], de forma que os dados podem ser analisados em diferentes níveis de detalhe. Neste trabalho, o modelo proposto apóia-se fortemente na hierarquia de dimensões como forma de se garantir o acesso seguro e inequívoco ao dado em seu menor nível de granularidade.

2.10 - MODELAGEM EM FLOCOS DE NEVE (*"SNOWFLAKE SCHEMA"*)

Em um modelo em estrela a única tabela normalizada é a tabela de fatos [Kimball&Ross 02]. Ao contrário, as tabelas de dimensões apresentam muita redundância em virtude da existência das relações de hierarquia entre seus atributos. Alguns projetistas, incomodados com a situação e preocupados com o espaço de armazenamento das dimensões propuseram um esquema alternativo para a modelagem dimensional onde as dimensões passaram a ser normalizadas e ligadas através de chaves estrangeiras. Na Figura 2.14 é apresentado um esquema em flocos de neve para a dimensão Produto.

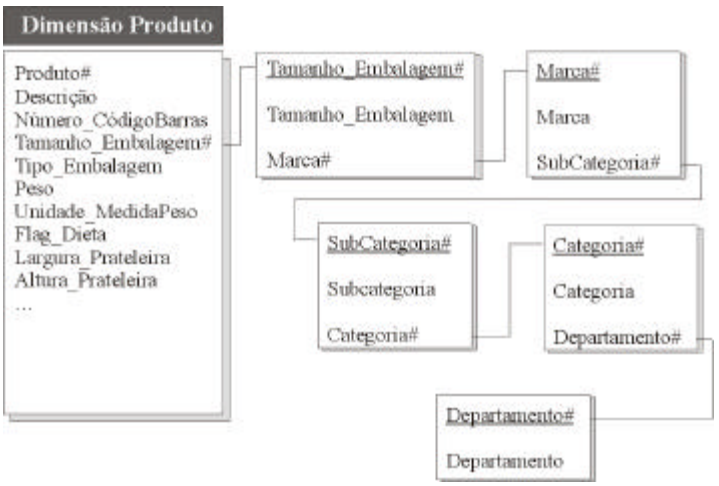

Figura 2.14 – Dimensão Produto Normalizada

A opção pelo esquema em flocos de neve é criticada por alguns consultores em dois aspectos: o primeiro é que não se justifica um aumento de complexidade na representação de dimensões visto que as mesmas, em um data warehouse, ocupam menos de 10% (dez por cento) da área de armazenamento [Kimball&Ross 02] e a economia de espaço produzida com o desmembramento não chega a 1% (um por cento) [Kimball&Ross 02]; o segundo é que com o desmembramento o número de junções necessário para a realização de consultas cresce sobremaneira, promovendo uma queda significativa no desempenho do data warehouse.

2.11 – CONSIDERAÇÕES FINAIS

Com este tópico encerramos a abordagem geral sobre a tecnologia de data warehouse e os seus processos de modelagem, nos focando inicialmente sobre uma conceituação e caracterização geral da tecnologia e derivando, mais especialmente, aos aspectos voltados àsua arquitetura.

Mais adiante apresentamos um estudo sobre as características que constituem a modelagem dimensional sobre o esquema de estrela e a sua

variação, o esquema de flocos de neve. Tal estudo é útil à medida que podemos localizar no modelo proposto no Capítulo 3, os pontos divergentes e convergentes em relação ao aceito como clássico, identificando os seus aspectos fortes e os fracos e como este modelo proposto se enquadra dentro das técnicas de modelagem dimensional existentes.

CAPíTULO 3

Esquema Dimensional Hierárquico Genérico

3.1 – INTRODUÇÃO

A busca de esquemas dinâmicos para sistemas de bancos de dados informacionais (data warehouse) traz uma questão subjacente: Que arquitetura de referência poderia ser genérica o suficiente para se adequar à modelagem de qualquer realidade, além de poder refletir posteriores alterações com a manutenção da organização? A consideração realizada em [Kimball & Ross 02] nos inclina a acreditar que esta talvez não devesse ser a principal preocupação para os projetistas de data warehouse, provavelmente em função da grande complexidade inerente à abordagem corporativa. No entanto, talvez para uma abordagem departamental, caso dos data marts, pudéssemos fixar uma estrutura de dimensões de maneira que se pudesse refletir qualquer negócio.

Este trabalho pretende responder a esta questão através da proposição de uma arquitetura de referência, hierárquica e genérica, implementável em sistemas de gerenciamento de bancos de dados relacionais, para a modelagem de Data Warehouse ou *"Data Marts"*.

Há duas opções para implementação da modelagem multidimensional através de SGBD relacional: a estrutura estrela ("*star*") e a estrutura flocos de neve ("*snow flake*"), conforme vimos no Capítulo 2 [Kimball & Ross 02] [Bliujute et al 98] [Kimball et al 98] [Thomsen 97] [Rocha 00] [Chuck et al 98] [Kenan 95] [Kimball 97]. Ambas efetivamente são **estruturas de referência,** a segunda derivada da primeira, sobre as quais são projetados os modelos dimensionais. Desta forma, os projetos lógicos de banco de dados relacionais que são construídos a partir destes modelos são compostos por tabelas específicas constituídas por tuplas [Silberschatz et al 99] [Elmasri & Navathe 00], cujos atributos apresentam uma aderência completa à realidade que está sendo modelada. Isto se dá porque a denominação das tabelas de fatos e dimensões e a denominação dos seus atributos variam e são diretamente dependentes das realidades alvos do processo de modelagem [Red Brick 95]. Uma vez concluído o projeto lógico, obtêm-se uma representação da realidade

consubstanciada em um esquema de banco de dados que é **estático** [Elmasri & Navathe 00] [Kimball & Ross 02]. Desta forma, a qualquer tempo em que uma alteração da realidade determine uma modificação no esquema de banco de dados, esta exigirá uma nova análise da situação com a inexorável obrigação de uma intervenção em nível de projeto, inevitavelmente através de um projetista de banco de dados. Este cenário conduz o usuário do data warehouse, sempre orientado a assuntos Inmon & Hackthorn 97] e realidades específicas, a uma dependência do conhecimento de técnicas de projeto em bancos de dados relacionais que, obviamente, não são de sua competência. O trabalho ora proposto procura oferecer uma alternativa de solução a esta questão.

Para que um esquema dimensional seja genérico o suficiente para representar qualquer realidade deve apresentar, segundo nossa análise, quatro características:

a) Utilizar-se de um mecanismo de sistematização de informações que seja baseado em uma forma de organização naturalmente identificável ou adaptável em qualquer ambiente de negócios. Esta forma de organização é a **hierárquica**, que depois da desorganização completa, nos parece a mais trivial;

b) Utilizar-se de unidades atômicas para armazenamento dos dados, considerando-se as formas através das quais este pode ser implementado no ambiente computacional. São formas identificadas os **alfanuméricos**, **textos** (simplificam o tratamento de grandes conjuntos de alfanuméricos), **datas, números**, **imagens** (estáticas ou em movimento) e **áudio**. Observamos que nesta proposta admitimos utilizar como atributo da tabela de fatos não apenas dados numéricos (aditivos, semi-aditivos e não aditivos) [Kimball & Ross 02], embora estes sejam os preponderantes, mas também outros tipos de dados para simples consultas para complementação da formação do conhecimento;

c) Utilizar-se de unidades atômicas para armazenamento da informação cujos domínios sejam previamente estabelecidos e os possíveis conteúdos também, para que se possa realizar junções entre os atributos das tabelas de fatos. Estas unidades são denominadas de **tabelas gerais**;

d) Utilizar-se de unidades atômicas especificamente destinadas à descrição do comportamento e das características das demais unidades atômicas implementadas como atributos das tabelas de fatos. Estas unidades são os **metadados**;

Atendendo às características primitivas acima relacionadas, chegamos a uma proposta de estrutura **genérica**, **hierárquica** e **dinâmica** (atendendo às restrições de integridade referencial de forma transparente ao usuário final), apresentada em detalhes nos tópicos seguintes.

3.2 – O MODELO DIMENSIONAL HIERÁRQUICO GENÉRICO

Apresentamos na Figura 3.1 a estrutura do modelo dimensional proposto. Nos aspectos fundamentais apresenta os mesmos conceitos propostos nas referências bibliográficas no que se refere aos aspectos de orientação a assuntos, integração, não volatilidade e temporalidade Inmon & Hackthorn 97], como também à utilização de tabelas de fatos e de dimensões [Kimball & Ross 02] [Bliujute et al 98] [Kimball et al 98] [Thomsen 97] [Rocha 00] [Chuck et al 98] [Kenan 95] [Kimball 97], não obstante apresentar algumas diferenciações quanto a estes últimos pontos.

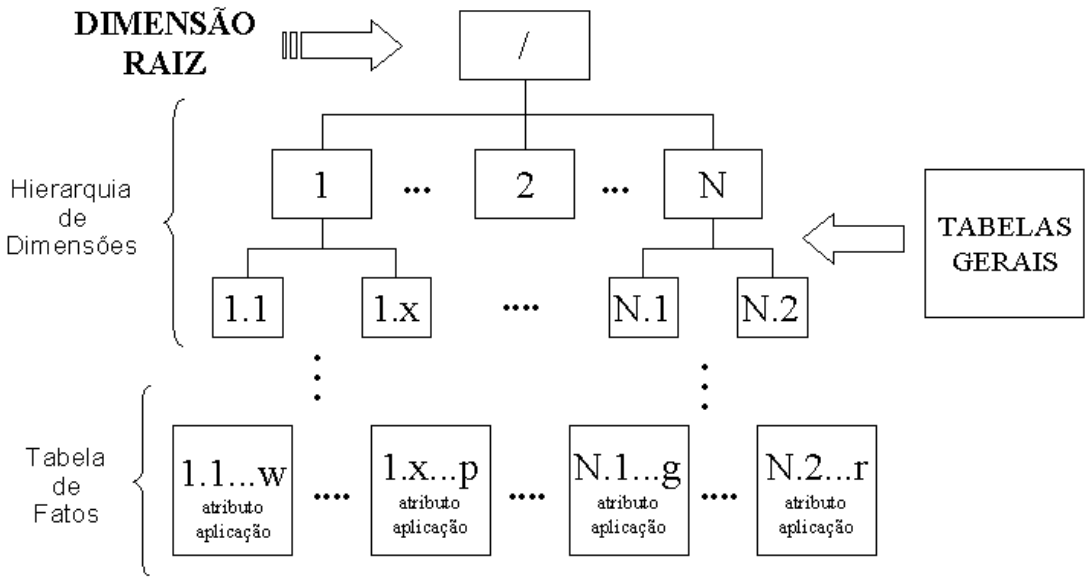

Figura 3.1 – Modelo Dimensional Hierárquico

A seguir abordaremos cada um dos pontos que compõe a estrutura da Figura 3.1.

Raiz – O Ponto de Partida

Esta porção não apresenta função alguma no que se refere ao armazenamento de atributos ou estabelecimento de ótica ou percepção sobre o negócio a ser modelado. É apenas um ponto inicial, uma referência fixa, da qual se desenvolve toda a hierarquia de dimensões.

A Hierarquia de Dimensões

Esta é uma das diferenciações deste modelo em relação aos modelos estrela e flocos de neve [Red Brick 95]. Nestes, as dimensões estão orbitando em torno da tabela de fatos, mas não guardam relacionamento algum entre si (apenas no esquema de flocos de neve se decompõe os relacionamentos 1:N dos atributos internos às dimensões, sendo que a junção à tabela de fatos ainda se dá pela dimensão original ou o produto das junções do desmembramento). Na Figura 3.1 pode-se observar que o modelo é construído a partir de uma abordagem "top-down" realizada sobre o negócio ou área do negócio até se alcançar um conjunto de fatos naturalmente identificados ao final do processo. Importante se ressaltar que ao contrário dos modelos existentes [Kimball & Ross 02] [Red Brick 95], não incluímos atributos nas dimensões. Os atributos de dimensão são implementados diretamente nas tabelas de fatos através de um dos tipos de dados atômicos, relacionados no tópico anterior. Para contribuir com a compreensão analisemos a Figura 3.2.

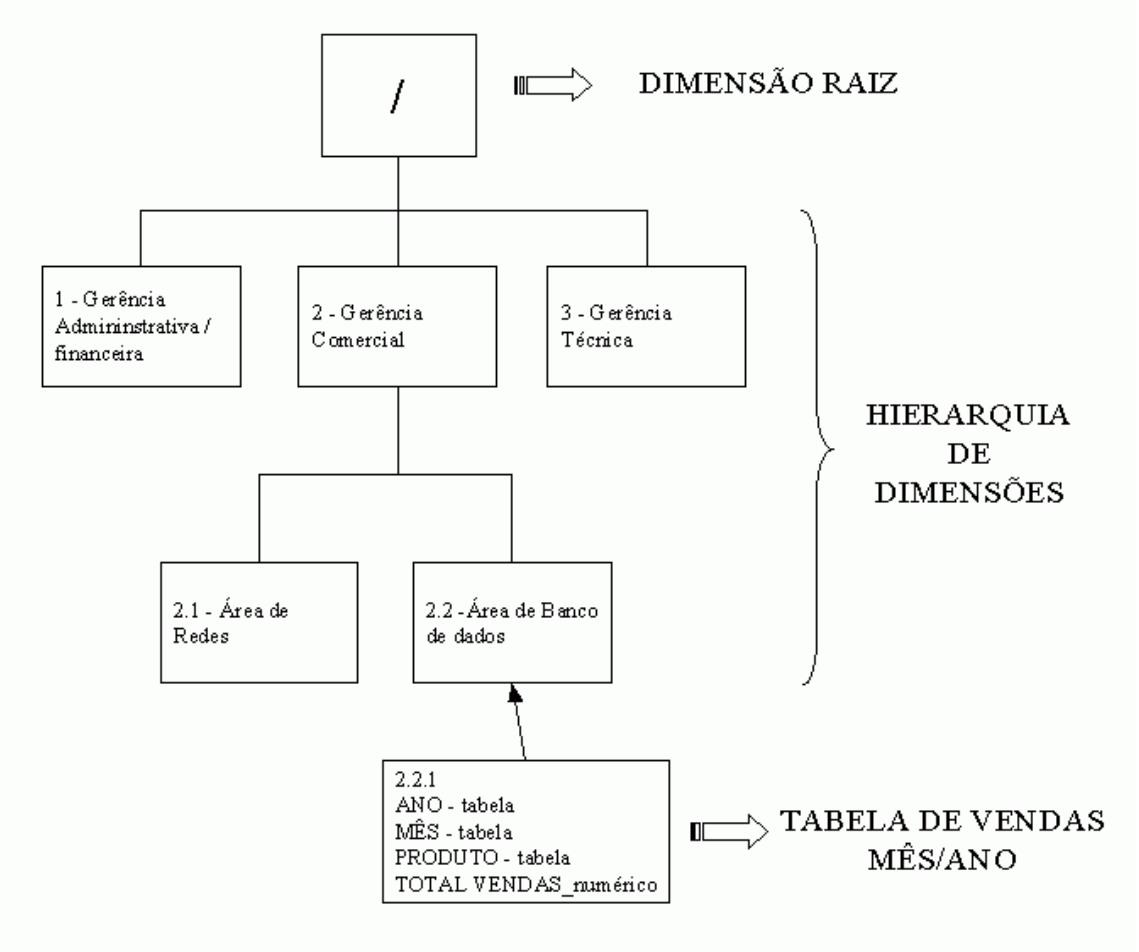

Figura 3.2 – Modelagem para Pequeno Negócio

Nesta Figura 3.2 está modelado um pequeno negócio baseado em serviços de informática. Há três gerências (administrativa/financeira, comercial e técnica), sendo que a gerência comercial possui duas áreas (redes e bancos de dados). Até este ponto estamos apenas caminhando na hierarquia de dimensões.

Através deste caminho chegamos à dimensão 2.2 – Área de Banco de Dados que, por sua vez, tem como filha uma tabela de fatos 2.2.1, denominada de Vendas por Mês/Ano que apresenta os atributos do tipo tabela - tabelas gerais - (ano, mês e produto) e do tipo numérico (total_vendas). Assim, os atributos ano, mês e produto são não-aditivos (não podem ser somados em relação a nenhuma dimensão) e o atributo total_vendas é aditivo (pode ser somado em relação a qualquer das dimensões) [Kimball & Ross 02]. Observe que para

chegarmos ao conceito de dimensões tal qual o mesmo é tratado nas referências bibliográficas [Kimball & Ross 02] [Red Brick 95] precisamos combinar a hierarquia de dimensões com as unidades atômicas do tipo **tabelas gerais**, apresentadas na Figura 3.1.

O fato de não utilizarmos atributos nas dimensões podem nos trazer algumas críticas especialmente relacionadas ao quão completa seria a caracterização das dimensões. Pode-se questionar como se armazenaria informações complementares em relação ao Produto (embalagem, apresentação, peso, tipo de prateleira, entre outros). Esta não é a questão central a ser atacada, mas poderia ser alvo de uma proposta com o objetivo de detalhar as **tabelas gerais**. Não afeta, contudo, o foco deste trabalho.

Uma comparação visual com o esquema em estrela é apresentada na Figura 3.3.

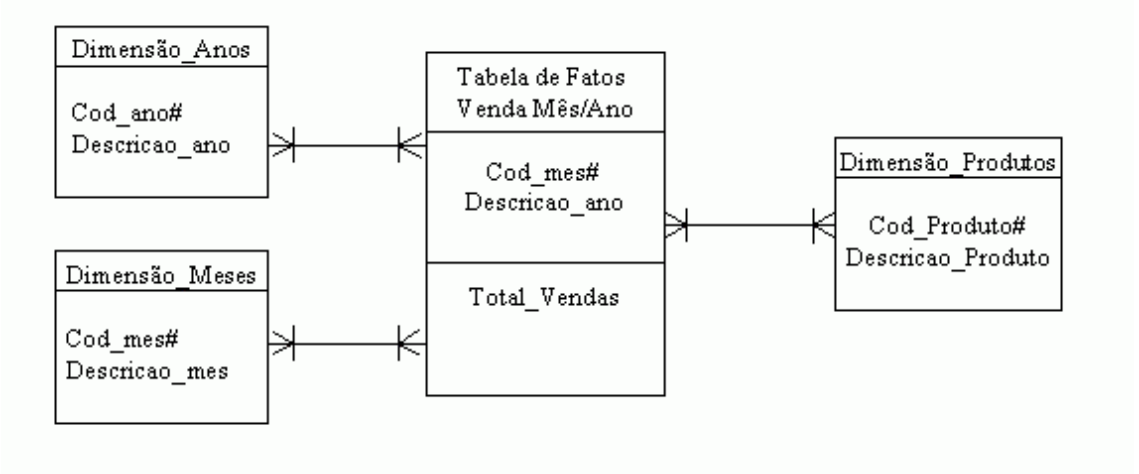

Figura 3.3 – O Modelo do Pequeno Negócio em Estrela

Sem dúvida a visualização no modelo estrela é bastante simples, assim como no modelo proposto. No entanto no modelo estrela o conceito de chaves é explícito [Kimball 98 a] [Kimball 98 b] enquanto que no modelo proposto este é um conceito que é abstraído. No modelo estrela há vários caminhos para se chegar ao núcleo, que são os fatos, enquanto no modelo proposto há um único

e rígido caminho garantido pela estrutura hierárquica. Apesar de, *a priori*, tal rigidez contribuir para formar a convicção que a variedade das consultas possíveis estaria comprometida, asseguramos que tal não ocorre. O usuário final pode, por exemplo, adicionar o Total_Vendas através de qualquer das tabelas gerais (ano, mês ou produto), consideradas em outros modelos simplesmente como dimensões.

As Tabelas de Fatos

As tabelas de fatos apresentam duas diferenças básicas em relação aos modelos clássicos [Kimball & Ross 02] [Red Brick 95]:

a) o conceito de chave é abstraído de forma que o usuário final não precisa tratá-lo diretamente [Kimball 98 a] [Kimball 98 b]. Na verdade a utilização dos atributos do tipo tabela - **tabelas gerais -**, que possuem domínios pré-estabelecidos e definidos, garantem a possibilidade da realização plena das junções necessárias entre as tabelas de fatos e de domínios. A questão remanescente é a da **unicidade** da **chave primária** [Silberschatz et al 99] [Elmasri & Navathe 00]. Tal questão pode ser facilmente contornada pela verificação da existência de um único conjunto de valores para o subconjunto de atributos das tabelas de fatos do tipo tabela. Ao contrário, no modelo estrela a questão das chaves deve ser explicitamente tratada. Consideramos esta uma vantagem do modelo proposto;

b) Os atributos aceitos nas tabelas de fatos não ficam restritos ao tipo numérico, apesar de ser esta uma questão basilar tratada na bibliografia existente [Kimball & Ross 02]. Obviamente que reconhecemos que realmente os atributos numéricos são o foco da construção dos data warehouses. No entanto, nos propusemos a enriquecer este modelo com possibilidade do armazenamento de outros tipos de dados tais como textos, vídeo e áudio, como forma de oferecer informação subjacente. Por exemplo, no momento de

analisarmos dados de vendas do último trimestre talvez fosse relevante ouvir as palavras do presidente da companhia sobre o fato de não se atingir as metas no ano fiscal.

É importante observar também que as tabelas de fatos, ao invés de ficarem ao centro como no modelo estrela, ficam nas **folhas** da hierarquia de dimensões. No entanto, o acesso às tabelas de fatos, da mesma forma, se dá unicamente através das dimensões [Kimball & Ross 02].

As Tabelas Gerais

As tabelas gerais formam um aspecto fundamental do esquema proposto. Através delas conseguimos a complementaridade em relação à hierarquia de dimensões que nos permite chegar ao conceito clássico de dimensões [Kimball & Ross 02] [Red Brick 95] e nos proporciona a abstração do conceito de chaves.

As tabelas gerais são simplesmente o conjunto de dados tratados pelo negócio em que os valores possíveis são identificados de maneira discreta e são limitados. De outra forma, seriam o conjunto de dados sobre o negócio cujas possíveis ocorrências para os domínios são pré-determinados. Assim, não se pode instanciar um fato do tipo tabela sem que o conteúdo não tenha sido instanciado na tabela geral correspondente. É imperioso, no entanto, que seja dado ao usuário o conhecimento sobre todas as tabelas gerais existentes e os seus respectivos conteúdos para que o mesmo não lance mão de outros tipos inadvertidamente.

Baseados na Figura 3.2, os atributos Ano, Mês e Produto são instâncias das tabelas gerais. Os valores possíveis para o atributo Ano poderiam ser, por exemplo, 2003, 2002, 2001 e 2000 de uma tabela geral denominada Anos. Os

valores para o atributo Mês poderiam ser Jan, Fev, Mar, Abr, Mai, Jun, Jul, Ago, Set, Out, Nov e Dez de uma tabela geral denominada Meses.

A depender da dinâmica do data warehouse ou *"data mart"* estas tabelas gerais podem ser criadas ou eliminadas. A operação de eliminação deve apresentar uma freqüência modesta, uma vez que não é comum uma grande quantidade de correções por erros de concepção na definição dos fatos. Por outro lado, a expectativa de criação de novas tabelas é razoável, vez que a própria utilização do data warehouse leva ao seu crescimento e evolução.

Há a necessidade, mesmo com a abstração do conceito de chaves, de se garantir **integridade referencial** entre as tabelas gerais e as tabelas de fatos. Assim, qualquer exclusão em uma tabela geral cujo conteúdo tenha sido instanciado em tabelas de fatos não é permitida.

Os Metadados

Os metadados nesta proposta seguem a conceituação e a funcionalidade clássica. São dados a respeito dos dados [Harrison 98]. São formas para que possamos descrever o comportamento dos atributos armazenados nas tabelas de fatos. Importante ressaltar que os metadados não são utilizados para a descrição de dimensões. Após análise criteriosa chegamos a um conjunto mínimo de 7 (sete) metadados que consegue, até o momento, atender as necessidades dos modelos de negócios tratados. São eles:

a) **Abordagem**: referimos à possibilidade da tabela de fatos refletir o seu alcance dentro do negócio (por exemplo, as filiais de uma empresa);

b) **Fonte:** referimos àfonte de dados primitivos ou fontes de dados externas;

c) **Série Temporal:** consideramos que algumas tabelas de fatos podem não apresentar um comportamento temporal (histórico), de acordo com o que já foi exposto;

d) **Método de Povoamento:** referimos às possibilidades para instanciação dos dados no data warehouse. Esta pode ocorrer por digitação ou importação (não tratamos de *"data staging"* nesta proposta [Kimball & Ross 02] [Inmon 97];

e) **Confiabilidade:** referimos à qualidade do dado. Pode ser regular, boa ou ótima, a depender da fonte, de sua atualização e do método de coleta;

f) **Periodicidade:** referimos ao período de tempo para completar o ciclo de envelhecimento do fato [Inmon & Hackthorn 97] [Gardner 98] [Inmon 97];

g) **Prazo para Antecipação da Coleta:** refere-se à antecedência necessária para que seja demandada àfonte a atualização do fato.

3.3 – IMPLEMENTAÇÃO EM BANCO DE DADOS RELACIONAL – PROJETO LÓGICO

Nesta proposta procuramos conceber um projeto lógico para implementação em bancos de dados relacionais [Silberschatz et al 99] [Elmasri & Navathe 00] que suportasse qualquer modelagem dimensional desenvolvida de acordo como a estrutura da Figura 3.1. Assim, utilizamos técnicas da modelagem de sistemas UML (*"Unified Modeling Language"*) [Jacobson et al 00] [Fowler & Kendall 00] [Melo 02] para a análise de requisitos e construção de interfaces, cujos diagramas apresentamos a seguir, além do projeto lógico do banco de dados implementado em uma ferramenta CASE [Lima 02] acompanhado das descrições das correspondências entre o modelo dimensional da Figura 3.1 e as tabelas relacionais que o implementam.

3.3.1 – Diagrama de Classes

O diagrama de classes UML [Jacobson et al 00] [Fowler & Kendall 00] [Melo 02] para este trabalho pode ser visualizado na Figura 3.4 abaixo.

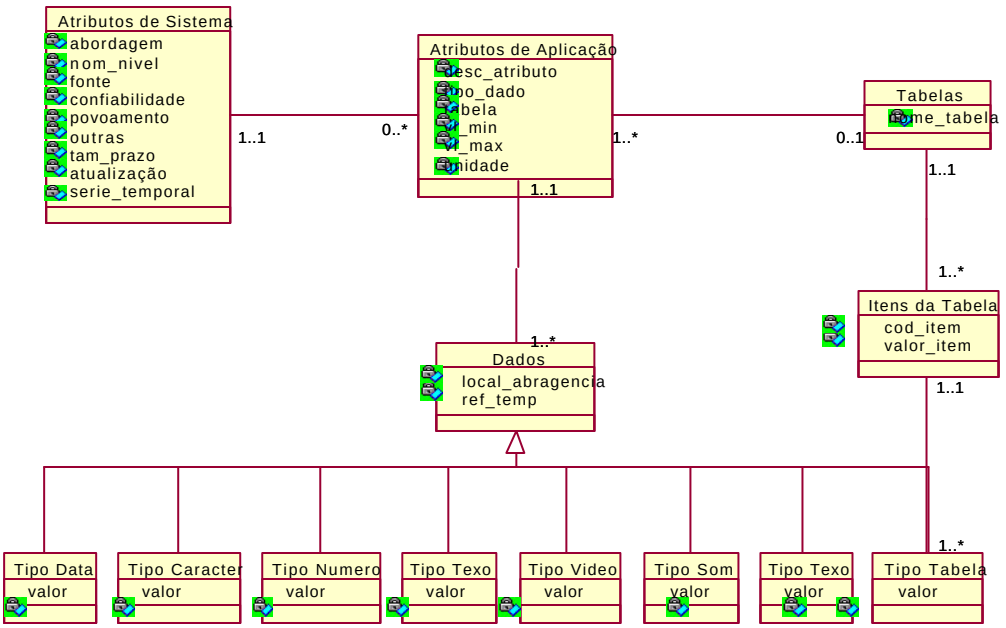

Figura 3.4 – Diagrama de Classes para Implementação do Esquema Proposto

O esquema dimensional proposto neste trabalho é implementado através de 13 (treze) classes em UML [Jacobson et al 00] [Fowler & Kendall 00] [Melo 02]. Para estas classes não há métodos propostos, mas tão somente atributos. Importante ressaltar que um auto-relacionamento da classe **atributos de sistema** não está representado por não haver como fazê-lo na versão disponível da ferramenta utilizada, o Rational Rose [Quatrani 01] [Boggs 02]. Deve-se observar também que para cada unidade atômica de representação de dados, que irá se consubstanciar em fatos há uma classe específica, inclusive para as tabelas gerais.

3.3.2 – Diagramas de Casos de Uso

A representação dos cenários, atores e seus papéis na solução proposta está apresentada na Figura 3.5. Deve-se registrar, por oportuno, que neste trabalho cuidamos apenas dos cenários relacionados à administração do data warehouse, o seu povoamento e a manutenção das tabelas gerais. O cenário relacionado ao papel do usuário na extração das informações não é contemplado neste trabalho apesar do mesmo ter sido implementado.

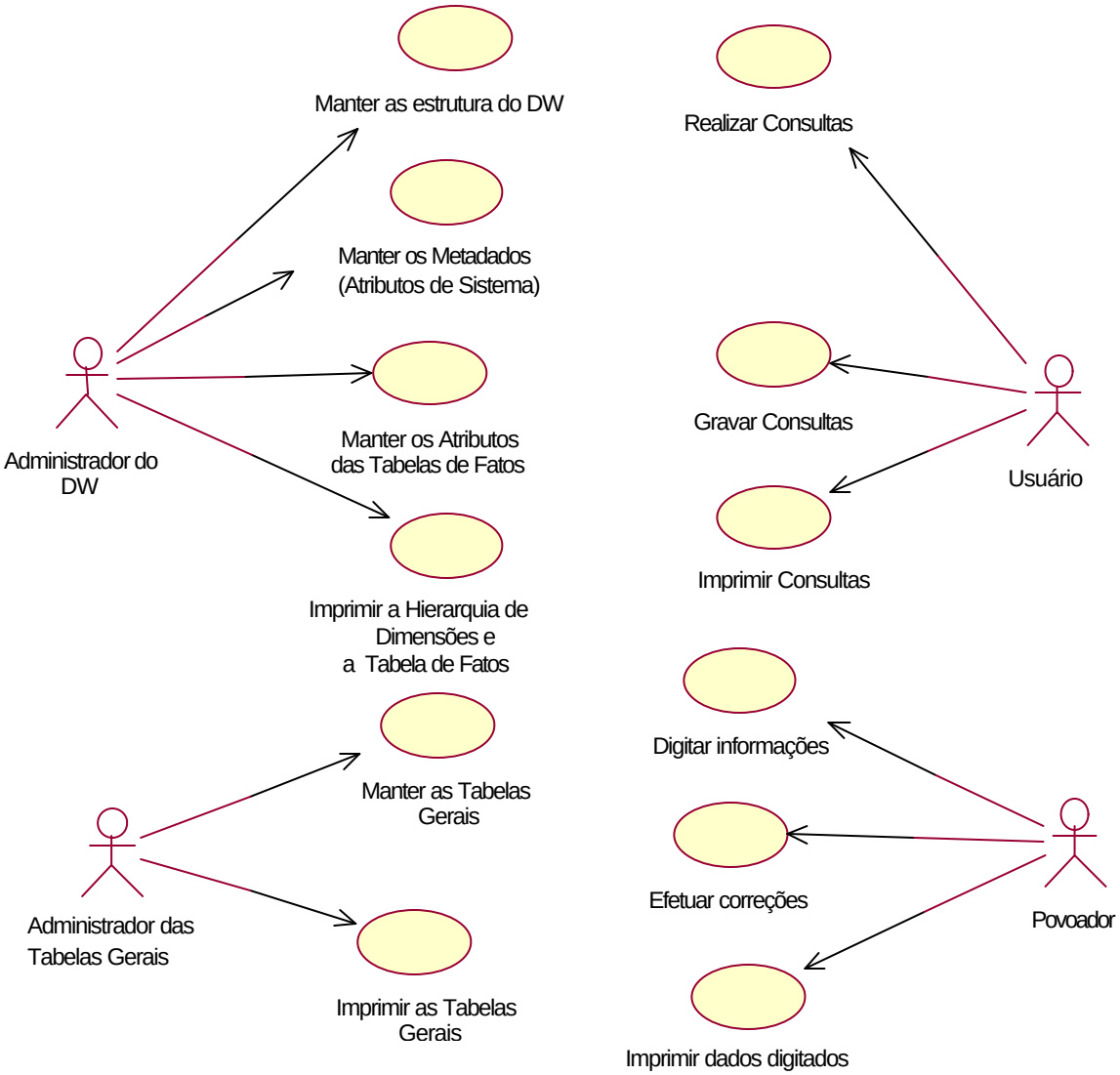

Figura 3.5 – Diagramas de Casos de Uso dos Possíveis Cenários

Dentro dos cenários considerados, discorreremos sobre os atores e seus papéis:

a) **Administrador do DW**: é o responsável por qualquer alteração na estrutura do data warehouse. As alterações possíveis envolvem a criação, eliminação ou alteração de quaisquer dimensões, além de modificação na estrutura da hierarquia. Toda e qualquer alteração envolvendo as tabelas de fato, tanto no que tange a seus metadados ou atributos é da exclusiva competência deste ator. Na solução implementada, as restrições de integridade referencial [Silberschatz et al 99] [Elmasri & Navathe 00] [Kimball & Ross 02] [Red Brick 95] e garantias de que fato algum pode ser eliminado do data warehouse se estiver povoado são observadas criteriosamente.

b) **Administrador das Tabelas Gerais**: é responsável pela criação, alteração e exclusão de quaisquer das tabelas gerais. É o detentor do conhecimento sobre que tipos de tabelas, com os seus respectivos conteúdos, estão disponíveis para utilização nas tabelas de fatos. A sua função é crucial porque, em conjunto com o Administrador do DW, garante a implementação completa do conceito de dimensões [Kimball & Ross 02] [Bliujute et al 98] [Kimball et al 98] [Thomsen 97] [Rocha 00] [Chuck et al 98] [Kenan 95] [Kimball 97];

c) **Povoador**: é o responsável pela carga de dados no data warehouse. Já afirmamos que este trabalho não contempla os aspectos relacionados a *"data staging"* [Kimball & Ross 02] [Inmon 97] ou aos seus processos ETL (Extração, Transformação e Carga). A carga de dados é realizada mediante digitação pura e simples, sendo que algum trabalho de limpeza pode ser realizado em etapas anteriores, mas foge ao escopo deste trabalho.

3.3.3 – Projeto Lógico (ERWin)

Apresentamos neste tópico o projeto lógico produzido através da ferramenta CASE (Computer-Aided System Engineering) denominada ERWin 3.0 [Lima 02]. É um modelo lógico bastante aproximado do relacional [Silberschatz et al 99] [Elmasri & Navathe 00] e, dentro do aspecto prático, é utilizado para a sua implementação.

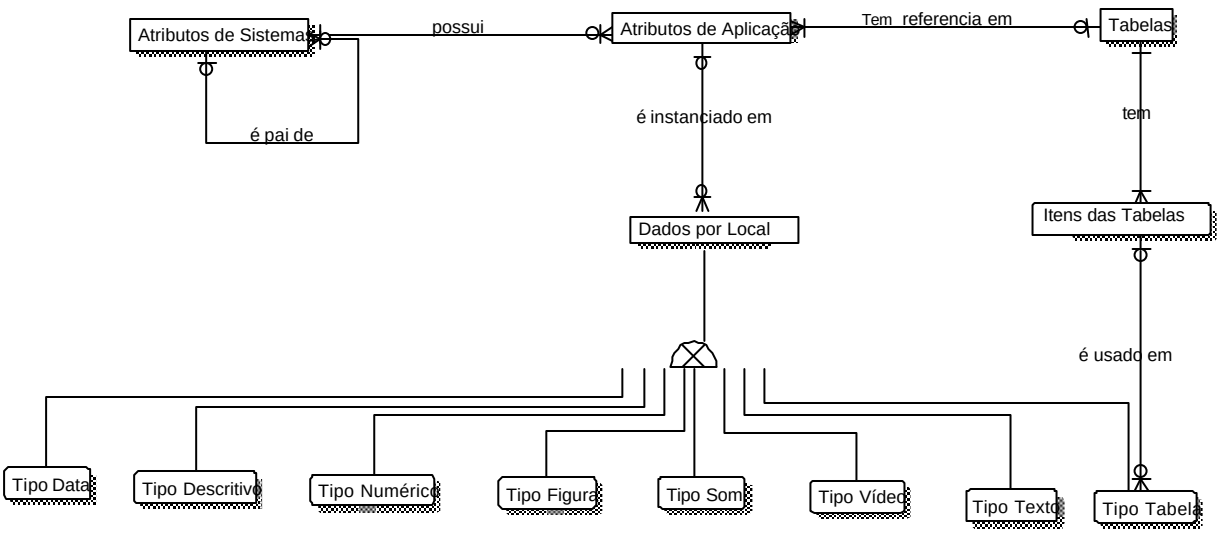

Figura 3.6 – Projeto Lógico do Modelo Dimensional

Vamos a seguir estabelecer as correspondências entre os blocos que compõem o modelo dimensional proposto e as suas devidas implementações no modelo lógico.

3.3.4 – Implementação da Hierarquia de Dimensões

Estabelecendo como referências as Figuras 3.1 e 3.6 podemos afirmar que o bloco denominado hierarquia de dimensões é representado por uma única relação no projeto lógico. Esta relação é a de **Atributos de Sistema**. A sua principal característica do ponto de vista lógico é um auto relacionamento 1:0,1,N [Silberschatz et al 99] [Elmasri & Navathe 00] que implementa a formação da árvore hierárquica, cujas chaves serão conhecidas quando tratarmos do projeto físico.

3.3.5 – Implementação das Tabelas de Fatos

Este é o maior subconjunto de relações deste projeto lógico. Como já é conhecido das referências bibliográficas este tipo de relação irá ocupar o maior espaço – cerca de 90% (noventa por cento) do data warehouse [Kimball & Ross 02]. È formado pelas relações **Atributos de Aplicação, Dados por Local, Tipo_Data, Tipo_Descritivo, Tipo_Numérico, Tipo_Figura, Tipo_Som, Tipo_Vídeo, Tipo_Texto e Tipo_Tabela.**

Na verdade, a estrutura da tabela de fatos se inicia ainda na relação **Atributos de Sistema**. Lá estão instanciados os seus metadados. Apenas quando uma instância desta relação é uma tabela de fatos então é exercitado o relacionamento desta com a relação **Atributos de Aplicação** para a construção da estrutura dos atributos das tabelas de fato. Em **Atributos de Aplicação** são instanciadas as descrições de todos os atributos – observe que o usuário poderá utilizar a descrição que melhor lhe aprouver, não estando condicionado aos identificadores mnemônicos utilizados largamente em projetos de banco de dados -, o seu tipo de dados (sendo do tipo tabela a que tabela geral está vinculado), os seus valores máximo e mínimo se forem do tipo numérico (já implementando uma restrição de domínio [Silberschatz et al 99]

[Elmasri & Navathe 00]) e a unidade de medida, se for o caso. Esta etapa tendo sido cumprida estará pronta toda a estrutura das tabelas de fatos, faltando apenas instanciá-la.

Para instanciar os fatos das tabelas de fatos entram em cena a relação **dados por local** associada às suas especializações, de acordo com o tipo básico do dado que estará assinalado em **atributos de aplicação**. A relação de **dados por local** é utilizada também para reunir todos os dados instanciados relativos a um mesmo evento. Pode parecer um tanto confuso agora, mas quando abordarmos os aspectos físicos do projeto todas as dúvidas deverão ser dirimidas através de referências diretas aos atributos das relações e utilizando os exemplos.

3.3.6 – Implementação das Tabelas Gerais

As tabelas gerais da Figura 3.1 são implementadas no projeto lógico através das relações **Tabelas** e **Itens das Tabelas**. Na primeira cada uma das tabelas gerais é descrita e na segunda os seus valores possíveis são instanciados. Observe que há uma cardinalidade [Silberschatz et al 99] [Elmasri & Navathe 00] de 1:0,N entre ambas significando que quando uma determinada tabela geral é criada, as suas instâncias poderão ser povoadas posteriormente. Importante ressaltar que estas se relacionam com relações de outros blocos do modelo dimensional, a saber: a relação de **Atributos de Aplicação** que trata da estruturação dos fatos das tabelas de fatos e a relação **Tipo_Tabela** que trata da instanciação de qualquer fato, na tabela de fatos, do tipo tabela. Tais relacionamentos garantem que nenhum atributo (fato) das tabelas de fatos será aceito no tipo de dado tabela se a tabela geral correspondente não tiver sido criada. De forma análoga, nenhum fato do tipo tabela poderá ser instanciado se o seu conteúdo não estiver previamente instanciado em **Itens das Tabelas**. Garantimos assim, de forma transparente para o usuário, a integridade referencial [Silberschatz et al 99] [Elmasri & Navathe 00]. A utilização das

tabelas gerais ficará mais evidenciada no tópico seguinte quando falaremos das junções necessárias para a extração de informações.

3.3.7 – As Junções

Utilizando o esquema ilustrado na Figura 3.6, podemos observar que todos os valores tratados ficam instanciados apenas nas especializações da relação **Dados por Local**. Assim, qualquer junção necessária para extração de informações será aplicada apenas às relações **Dados por Local , Tipo_Data, Tipo_Descritivo, Tipo_Numérico, Tipo_Figura, Tipo_Som, Tipo_Vídeo, Tipo_Texto e Tipo_Tabela.** Desta forma, não precisamos percorrer nenhuma outra relação do projeto lógico nem tampouco realizar junções sucessivas sobre as relações que compõem a hierarquia de dimensões. Estas, juntamente com as tabelas gerais, são utilizadas unicamente para implementar o conceito de dimensões. Com esta estruturação, o data warehouse pode crescer indefinidamente em dimensões ou fatos, mas as junções permanecerão sendo realizadas apenas sobre as relações acima citadas. Concluímos que neste modelo dimensional, portanto, as junções necessárias ficam restritas às tabelas de fatos o que nos orienta a acreditar em um razoável desempenho para o processamento de consultas.

3.4 - IMPLEMENTAÇÃO EM BANCO DE DADOS RELACIONAL – PROJETO FÍSICO

Apresentamos neste tópico o projeto físico de banco de dados relacional baseado no projeto lógico já abordado. Iremos detalhar o funcionamento de cada uma das relações e de seus atributos a fim de explicar, de maneira completa, o funcionamento da implementação do modelo dimensional

proposto. Iremos procurar trabalhar com exemplos para auxiliar no processo de explanação e facilitar a compreensão.

3.3.1 – Descrição do Projeto Físico

O diagrama do projeto físico do modelo proposto é apresentado na Figura 3.7. Tal projeto foi desenhado com o auxílio da ferramenta CASE denominada ErWin [Lima 02].

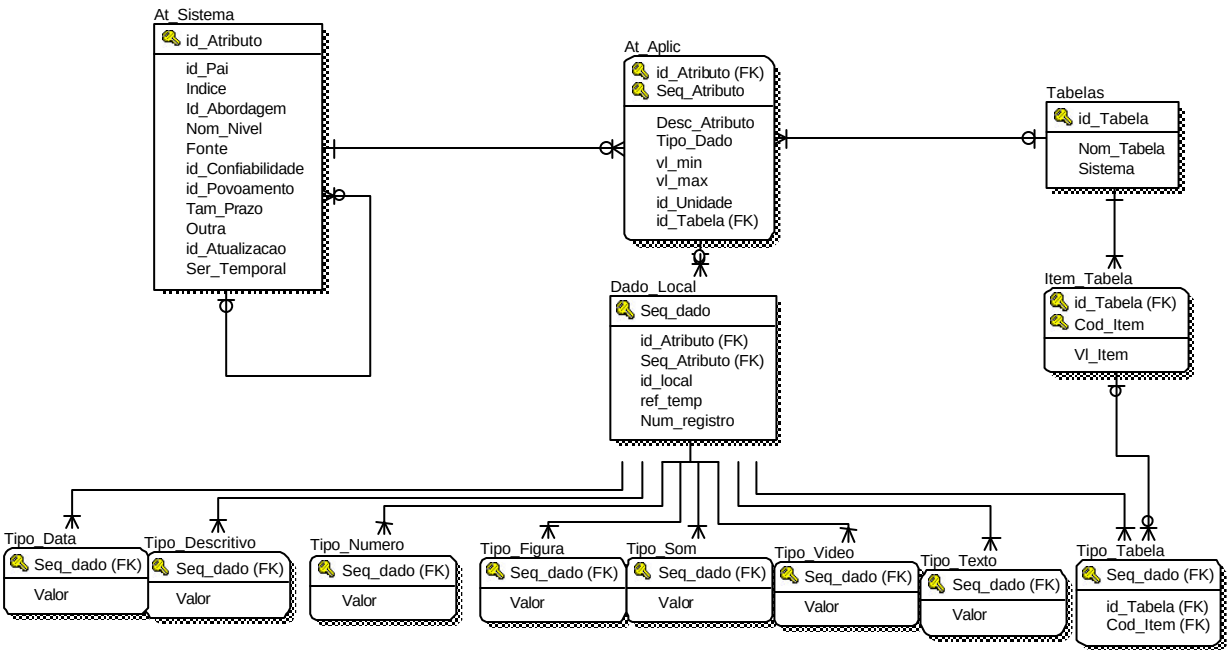

Figura 3.7 – O Projeto Físico do Modelo Dimensional

A Relação AT_Sistema

Esta relação é responsável pela implementação da hierarquia de domínios e dos metadados das tabelas de fatos. Observe que utilizamos a mesma estrutura para ambas as representações já que a tabela de fatos é um nodo folha da árvore utilizada no modelo. Importante ressaltar que para as

dimensões não há instanciação dos atributos que implementam os metadados. Na Tabela 3.1 apresentamos o dicionário de dados correspondente.

Tabela 3.1 – Dicionário de Dados de AT_Sistema

Atributos	**Tipo**	Comentário
Id_Atributo	Int (PK)	Identificador do nó da arvore. Trata-se de número seqüencial auto-implementado que identifica cada nó da árvore.
Id_Pai	Int	É a chave estrangeira oriunda do autorelacionamento que existe nesta relação.
Índice	Varchar(20)	Este campo possui uma identificação mais amigável para a dimensão ou fato dentro da árvore. Ele possui um formato de acordo com o exemplo: Ex.: 2.1 – Área de Redes (Figura 3.2)
Id_abordagem	Int	Possui o código da abordagem do fato (METADADO).
Nom_Nível	Varchar(100)	Identificação da dimensão ou fato (METADADO).
Fonte	Varchar(50)	Fonte de dados primitivos dos fatos(METADADO).
Id_Confiabilidade	Int	Possui o código do nível de confiabilidade do fato (METADADO).
Id_Povoamento	Int	Possui o código do método de povoamento do fato (METADADO).
Outra	Varchar(60)	Descrição de outro método de povoamento possível para o fato (METADADO).
Tam_prazo	Int	Prazo de antecipação (em dias) para coleta do fato na fonte primitiva (METADADO).
Id_Atualização	Int	Prazo para envelhecimento do fato (METADADO).
Ser_Temporal	Char(1)	Determina se o fato é histórico ou não (METADADO). (Como foi comentado aceitamos fatos não temporais apenas como informação complementar)

Os atributos Id_Atributo e Id_Pai são seqüenciais utilizados para garantir a existência e o caminhamento na hierarquia de dimensões. Tomando como referência a Figura 3.2 podemos admitir que, para a dimensão 2.2 – Área de Banco de Dados, o seu atributo Id_Atributo poderia ter o conteúdo 5 e Id_Pai poderia ser 2, pois estaria apontando para a dimensão 2 – Gerência Comercial, o seu pai. Usando estes ponteiros podemos nos deslocar na hierarquia com facilidade. Os demais atributos são metadados que, à exceção de Nom_Nível que descreve a dimensão ou o fato, são utilizados apenas nas tabelas de fatos. Os seus domínios são determinados por tabelas gerais correspondentes que denominamos **de sistema** por não apresentarem dados, mas sim metadados.

A Relação At_Aplic

Esta relação é instanciada apenas quando desejamos representar uma tabela de fatos em nossa modelagem dimensional. Observe que ela está afeta à criação da estrutura das tabelas de fatos, mas não à instanciação de dados. A Tabela 3.2 apresenta o seu dicionário de dados.

Tabela 3.2 – Dicionário de Dados de At_Aplic

Atributos	Tipo	Comentário
Id_Atributo	Int (PK)	Chave estrangeira procedente da relação AT_Sistema.
Seq_Atributo	Int (PK)	Ordem (seqüência) do fato na tabela de fatos.
Desc_Atributo	Varchar(100)	Descrição do fato.
Tipo_Dado	Int	Código do tipo de dado de acordo com a natureza do fato.
Id_Tabela	Int	Chave estrangeira vindo da relação TABELAS. Se o tipo de dado for diferente de tabela este campo deve ser preenchido com valor nulo.
Vl_min	Float	Valor mínimo se for do tipo numérico.
Vl_max	Float	Valor máximo se for do tipo numérico.
Id_Unidade	Int	Código da unidade de medida se for do tipo numérico.

Quando uma tabela de fatos precisa ser representada no modelo utilizamos as relações At_Sistema e At_Aplic. Enquanto a primeira se encarrega de posicionar a tabela de fatos na hierarquia e registrar os seus metadados, a segundo cria a sua estrutura de dados para a instanciação de fatos. O atributo Seq_Atributo é um seqüencial que ordena os fatos dentro da tabela e será utilizado para o referenciamento individual a cada um deles, inclusive nos processos de extração de informações. O atributo Desc_Atributo permite que o usuário final denomine o fato da forma que melhor lhe convier, sem a utilização de mnemônicos incompreensíveis [Kimball & Ross 02]. Se for do desejo do usuário, este pode utilizar como identificador até mesmo uma frase. O atributo Tipo_Dado enquadra o fato dentro de um dos tipos de dados possíveis no modelo proposto. Os seus possíveis valores estão instanciados previamente em uma tabela geral especificamente designada para este fim. O atributo Id_Tabela é utilizado apenas se o fato for do tipo tabela. Neste caso já é estabelecido o relacionamento com a relação Tabelas e Item_Tabela. Os atributos Vl_Min e Vl_Max são utilizados para estabelecer restrições de integridade de domínio [Silberschatz et al 99] [Elmasri & Navathe 00] para o caso do fato ser numérico e Id_Unidade para estabelecer a medida de grandeza para também o caso do fato ser numérico.

De acordo com a Figura 3.2 a tabela de fatos 2.2.1 – VENDAS MÊS/ANO, seria instanciada em At_Sistema e At_Aplic da seguinte forma:

At_Sistema(**6**,5,2.2.1,1,”VENDAS ÊS/ANO”,”VENDEDOR”,1,1,””,60,360,”S”);
At_Aplic(**6**,1,”ANO DAS VENDAS”,1,**1**,,,,);
At_Aplic(**6**,2,”MÊS DAS VENDAS”,1,**2**,,,,);
At_Aplic(**6**,3,”PRODUTO VENDIDO“,1,**3**,,,,);
At_Aplic(**6**,4,”QUANTIDADE VENDIDA”,2,,0,99999,1,).

Onde 1,2 e 3 são respectivamente os códigos das Tabelas de Anos, Meses e Produtos. Os valores 0 e 99999 são o mínimo e o máximo admissíveis

para os produtos vendidos e 1 é o código para a Unidade “UN" na tabela geral interna que guarda os valores possíveis para as unidades. Observe que esta tabela de fatos – VENDAS MÊS/ANO – sempre será identificada pelo código 6 e o seu fato “ANO DAS VENDAS”, por exemplo, sempre será identificado pela chave composta 6,1.

A Relação Dado_Local

Esta relação é utilizada para instanciação dos fatos. Todas as vezes que o modelo dimensional implementado for povoado esta relação e as suas especializações [Silberschatz et al 99] [Elmasri & Navathe 00] são instanciadas. A seguir, na Tabela 3.3 temos uma descrição de seus atributos.

Tabela 3.3 – Dicionário de Dados de Dado_Local

Atributos	Tipo	Comentário
Seq_dado	Int (FK)	Identifica cada instancia do fato nas relações que são das especializações de Dado_Local.
Id_Atributo	Int (PK)	Chave estrangeira proveniente da relação At_Aplic.
Seq_Atributo	Int (PK)	
Id_local	Int	Código do local de abrangência do fato. Está relacionado às tabelas gerais e é um dimensão inerente a qualquer fato tratado no data warehouse.
Ref_temp	DateTime	Caso o fato seja histórico este atributo garante a implementação da dimensão tempo.
Num_Registro	Int (PK)	Identifica cada instancia do Tabela de Fatos.

Os atributos Id_Atributo e Seq_Atributo são utilizados para identificar o fato dentro da Tabela de Fatos para qual um determinado valor será instanciado. Os atributos Id_local e Ref_tempo são utilizados para exigir sempre a presença das dimensões abrangência e tempo em qualquer fato tratado no data warehouse. A obrigatoriedade da dimensão tempo é fartamente

documentada nas referências bibliográficas Inmon & Hackthorn 97] [Kimball & Ross 02] [Harrison 98] [Bliujute et al 98] [24]. Já a dimensão abrangência não recebe a mesma atenção, mas, em nosso trabalho, identificamos a necessidade de demarcar o espaço que, junto com a dimensão tempo, pode e deve ser associado a todos os eventos de qualquer negócio. Ora, todo e qualquer evento de um negócio se passa em um determinado momento e em determinado lugar. Desta forma podemos utilizar a dimensão abrangência para situar o evento, por exemplo, em uma filial, ou cidade, ou estado, ou região ou país.

O atributo Num_Registro é utilizado para agrupar as instâncias dos fatos relacionados a um mesmo evento. Voltando à referência da Figura 3.2, se a Tabela de Fatos registrasse as vendas de 50 unidades do produto 1 (“Controle de Estoque”) no mês de janeiro do ano de 2002, de 100 unidades do produto 1 (“Controle de Estoque”) no mês de julho do ano de 2002 e 80 unidades do produto 2 (“Sistema de Marcação de Consultas”) no ano de janeiro de 2002, assim seriam instanciados os fatos na relação Dado_Local:

a) **Ano**: 2002
Mês: Janeiro
Produto: 1 (“Controle de Estoque”)
Total_Vendas: 50

Dado_Local(**1**,6,1,1,200202,**1**) (Parte do fato ANO do evento 1)
Dado_Local(**2**,6,2,1,200202,**1**) (Parte do fato MÊS do evento 1)
Dado_Local(**3**,6,3,1,200202,**1**) (Parte do fato PRODUTO do evento 1)
Dado_Local(**4**,6,4,1,200202,**1**) (Parte do fato TOTAL_VENDAS do evento 1)

b) **Ano**: 2002
Mês: Julho
Produto: 1 (“Controle de Estoque”)
Total_Vendas: 100

Dado_Local(**5**,6,1,1,200202,**2**) (Parte do fato ANO do evento 2)
Dado_Local(**6**,6,2,1,200202,**2**) (Parte do fato MÊS do evento 2)
Dado_Local(**7**,6,3,1,200202,**2**) (Parte do fato PRODUTO do evento 2)
Dado_Local(**8**,6,4,1,200202,**2**) (Parte do fato TOTAL_VENDAS do evento 2)

c) **Ano**: 2002
Mês: Janeiro
Produto: 2 (“Sistema de Marcação de Consultas”)
Total_Vendas: 80

Dado_Local(**9**,6,1,1,200202,**3**) (Parte do fato ANO do evento 3)
Dado_Local(**10**,6,2,1,200202,**3**) (Parte do fato MÊS do evento 3)
Dado_Local(**11**,6,3,1,200202,**3**) (Parte do fato PRODUTO do evento 3)
Dado_Local(**12**,6,4,1,200202,**3**) (Parte do fato TOTAL_VENDAS do evento 3)

Pode causar espanto afirmarmos sempre que instanciamos apenas parte do fato. No entanto, a afirmativa é verdadeira. Podemos observar que nenhum dos valores citados no enunciado do exemplo foi instanciado. Eles serão instanciados ao final nas relações que são as especializações da relação Dado_Local a partir dos tipos a eles atribuídos. Desta forma, ano, mês e produto serão instanciados na relação Tipo_Tabela e Total_Vendas na relação Tipo_Numero. Estas instanciações serão apresentadas mais adiante.

Com o exemplo ficaram claras também as funcionalidades dos atributos Seq_Dado e Num_Registro. Observe que o primeiro é incrementado continuamente. Isto se dá porque através de Seq_Dado é que um fato de um evento específico é localizado nas relações de Tipo correspondentes. Seq_Dados é a chave primária para as relações de Tipos.

O atributo Num_Registro, por seu turno, é o que reúne todos os fatos de um mesmo evento. Observe que na primeira venda descrita, em todas as tuplas de Dado_Local há um mesmo conteúdo para Num_Registro, que é 1. Isto significa que todas as tuplas se referem a um mesmo evento denominado 1. Na segunda venda o seu conteúdo já é 2 para todas as tuplas e na terceira o seu conteúdo é 3.

A Relação Tabelas

Esta relação é a porção do projeto físico que implementa o conceito de Tabelas Gerais, já tratado, do modelo dimensional. A ela estão relacionadas At_Aplic e Item_Tabela. A sua função é oferecer a chave primária e as respectivas denominações para as dimensões do modelo. Na Tabela 3.4 está relacionado o seu dicionário de dados.

Tabela 3.4 – Dicionário de Dados de Tabelas

Atributos	Tipo	Comentário
Id_Tabela	Int	Identificador das Tabelas Gerais.
Nom_Tabela	Varchar(100)	Denominação das Tabelas Gerais.
Sistema	Char(1)	Pode assumir o valor “S” se a tabela geral for de uso interno (METADADO) ou pode assumir o valor “N” se ela for usada para instanciar algum fato do tipo Tabela.

Na solução implementada sugerimos a designação de uma equipe específica para a manutenção destas Tabelas Gerais.

A Relação Item_Tabela

Esta relação, combinada com a anterior, implementa o conceito de Tabelas Gerais. Nela são instanciados os valores possíveis para todas as

Tabelas Gerais do modelo. Relaciona-se com a anterior e com a relação Tipo_Tabela para garantir que qualquer valor (fato) instanciado nesta tenha sido previamente definido como possível. Implementa integridade referencial com ambas. Na Tabela 3.5 apresentamos a descrição de sua estrutura.

Tabela 3.5 – Dicionário de Dados de Item_Tabela

Atributos	Tipo	Comentário
Id_Tabela	Int (PK)	Chave estrangeira da relação TABELAS.
Cód_Item	Int (PK)	Código do item dentro da relação.
Vl_Item	Varchar(100)	Conteúdo do item da tabela.

Para exemplificar iremos instanciar as Tabelas Gerais de Anos, Meses e Produtos. Para tanto utilizaremos os valores citados nas relações At_Sistema e Dados_Local.

a) Tabela de Anos

Tabela(1,”ANOS)
Item_Tabela(1,1,”2000”)
Item_Tabela(1,2,”2001”)
Item_Tabela(1,3,”2002”)
Item_Tabela(1,4,”2003”)

b) Tabela de Meses

Tabela(2,”MESES”)
Item_Tabela(2,1,”janeiro”)
Item_Tabela(2,2,”fevereiro”)
Item_Tabela(2,3,”março”)
Item_Tabela(2,4,”abril”)
Item_Tabela(2,5,”maio”)
Item_Tabela(2,6,”junho”)

Item_Tabela(2,7,”julho”)
Item_Tabela(2,8,”agosto”)
Item_Tabela(2,9,”setembro”)
Item_Tabela(2,10,”outubro”)
Item_Tabela(2,11,”novembro”)
Item_Tabela(2,12,”dezembro”)

C) Tabela de Produtos

Tabela(3,”PRODUTO”)
Item_Tabela(3,1,”Sistema de Controle de Estoque”)
Item_Tabela(3,2,”Sistema de Marcação de Consultas”)

A seguir descreveremos a estrutura de todas as relações de Tipo do projeto físico do modelo proposto. Todas apresentam estruturas semelhantes diferenciando-se apenas no tipo de dado armazenado. Instanciaremos, para efeito de exemplo, apenas os fatos relacionados na relação Dados_Local.

A Relação Tipo_Tabela

Esta é a relação que armazena os valores das dimensões implementadas através das Tabelas Gerais. A sua estrutura é descrita na Tabela 3.6.

Tabela 3.6 – Dicionário de Dados de Tipo_Tabela

Atributos	Tipo	Comentário
Seq_dado	Int (PK)	Chave estrangeira oriunda da relação Dado_Local.
Id_Tabela	Int (PK)	Chave estrangeira oriunda da relação Item_Tabela.
Cód_Item	Int (PK)	

Observe que não são armazenados valores na mesma, mas apenas as referências para a relação Item_Tabela. Esta decisão de projeto, ao tempo que diminui a quantidade de espaço requerido para o data warehouse promove um número maior de junções quando da extração das informações, comprometendo o desempenho. Concluiremos a parte final da instanciação de fatos iniciada em Dados_Local.

a) **Ano**: 2002
Mês: Janeiro
Produto: 1 (“Controle de Estoque”)
Total_Vendas: 50

Tipo_Tabela(1,1,3) (Instanciando o ano de 2002)
Tipo_Tabela(2,2,1) (Instanciando o mês de janeiro)
Tipo_Tabela(3,3,1) (Instanciando o produto 1)

b) **Ano**: 2002
Mês: Julho
Produto: 1 (“Controle de Estoque”)
Total_Vendas: 100

Tipo_Tabela(5,1,3) (Instanciando o ano de 2002)
Tipo_Tabela(6,2,7) (Instanciando o mês de julho)
Tipo_Tabela(7,3,1) (Instanciando o produto 1)

c) **Ano**: 2002
Mês: Janeiro
Produto: 2 (“Sistema de Marcação de Consultas”)
Total_Vendas: 80

Tipo_Tabela(9,1,3) (Instanciando o ano de 2002)
Tipo_Tabela(10,2,1) (Instanciando o mês de janeiro)

Tipo_Tabela(11,3,2) (Instanciando o produto 2)

A Relação Tipo_Numero

O objetivo desta relação é instanciar os dados numéricos. A sua estrutura está descrita na Tabela 3.7.

Tabela 3.7 – Dicionário de Dados de Tipo_Numero

Atributos	Tipo	Comentário
Seq_dado	Int (PK)	Chave estrangeira oriunda da relação Dado_Local.
Valor	Float	Armazena um valor do tipo numérico.

Instanciaremos os últimos fatos, relacionados ao Total_Vendas, do exemplo que está sendo trabalhado.

a) **Ano**: 2002
Mês: Janeiro
Produto: 1 (“Controle de Estoque”)
Total_Vendas: 50

Tipo_Numero(4,50) (Instanciando o valor 50)

b) **Ano**: 2002
Mês: Julho
Produto: 1 (“Controle de Estoque”)
Total_Vendas: 100

Tipo_Numero(8,100) (Instanciando o valor 100)

c) **Ano**: 2002
Mês: Janeiro

Produto: 2 (“Sistema de Marcação de Consultas”)

Total_Vendas: 80

Tipo_Numero(12,80) (Instanciando o valor 80)

A Relação Tipo_Figura

Esta relação é utilizada para armazenar as imagens. A sua estrutura é descrita na Tabela 3.8.

Tabela 3.8 – Dicionário de Dados de Tipo_Figura

Atributos	Tipo	Comentário
Seq_dado	Int	Chave estrangeira oriunda da relação Dado_Local.
Valor	Image	Armazena um arquivo de figura.

A Relação Tipo_Data

Esta relação é utilizada para armazenar as datas. A sua estrutura é descrita na Tabela 3.9.

Tabela 3.9 – Dicionário de Dados de Tipo_Data

Atributos	Tipo	Comentário
Seq_dado	Int	Chave estrangeira oriunda da relação Dado_Local
Valor	DateTime	Armazena um valor do tipo data.

A Relação Tipo_Descritivo

Esta relação é utilizada para armazenar cadeias de caracteres. A sua estrutura é descrita na Tabela 3.10.

Tabela 3.10– Dicionário de Dados de Tipo_Descritivo

Atributos	Tipo	Comentário
Seq_dado	Int	Chave estrangeira oriunda da relação Dado_Local
Valor	Varchar(100)	Armazena um valor do tipo descritivo.

A Relação Tipo_Som

Esta relação é utilizada para armazenar os arquivos de áudio. A sua estrutura é descrita na Tabela 3.11.

Tabela 3.11– Dicionário de Dados de Tipo_Som

Atributos	Tipo	Comentário
Seq_dado	Int	Chave estrangeira oriunda da relação Dado_Local
Valor	Image	Armazena um arquivo de áudio.

A Relação Tipo_Vídeo

Esta relação é utilizada para armazenar os arquivos de vídeo. A sua estrutura é descrita na Tabela 3.12.

Tabela 3.12 - Dicionário de Dados de Tipo_Vídeo

Atributos	Tipo	Comentário
Seq_dado	Int	Chave estrangeira oriunda da relação Dado_Local
Valor	Image	Armazena um arquivo de vídeo.

A Relação Tipo_Texto

Esta relação é utilizada para armazenar grandes cadeias de caracteres. A sua estrutura é descrita na Tabela 3.13.

Tabela 3.13 – Dicionário de Dados de Tipo_Texto

Atributos	Tipo	Comentário
Seq_dado	Int	Chave estrangeira oriunda da relação Dado_Local
Valor	Text	Armazena um texto extenso.

3.5 - CONSIDERAÇÕES FINAIS

Neste Capítulo 3 apresentamos o esquema proposto como alternativa para a modelagem dimensional para data warehouse e sua implementação em um banco de dados relacional geral.

Discorremos sobre os conceitos e funcionalidades dos seus vários componentes e como estes procuram se aproximar dos modelos clássicos, apesar de apresentarem as suas especificidades. A sua concepção surgiu de uma inquietação técnica com a questão da usabilidade de ferramentas de

formalização de negócios por parte do usuário final, cujos esforços são convergentes com aqueles materializados na tecnologia de data warehouse.

Consideramos que este é um modelo simples, baseado em estruturas e relações simples, mas que possibilita a modelagem de um amplo espectro de negócios. Certamente muitas questões relacionadas à própria implementação mereceriam novas reflexões. Pode-se questionar a utilização das tabelas gerais como elementos para representação de dimensões ou mesmo questionar a utilização das relações de Tipos como únicos repositórios dos fatos. A experimentação com alguns modelos nos ofereceram resultados no que tange a desempenho, aspecto crucial quando tratamos do armazenamento de dados da ordem de giga ou mesmo terá bytes.

De qualquer forma, acreditamos que a questão central que procuramos abordar, a possibilidade do próprio usuário final modelar um sistema de informações estratégico refletindo a realidade de seu negócio, foi atendida. A busca de generalidade, que é associada à primeira e passa pela estrutura hierárquica e pelo projeto único para o banco de dados relacional, foi preponderante neste trabalho, de forma que, o cuidado com outras questões como o estudo dos requisitos para armazenamento e o desempenho das junções não foram prioritários mas, no entanto, não comprometem os resultados obtidos.

No Capítulo 4 apresentaremos a implementação do esquema proposto utilizando-se como referência o modelo de negócio relacionado na Figura 3.2. Serão relacionadas as ferramentas desenvolvidas para a administração do data warehouse bem como uma visão do mesmo do ponto de vista do usuário, quando procuraremos denotar a simplicidade e naturalidade do seu uso.

CAPíTULO 4

Protótipo para Esquema Dimensional Hierárquico Genérico

4.1 – INTRODUÇÃO

Neste Capítulo serão apresentadas as principais operações que podem ser realizadas através do protótipo desenvolvido para implementação do esquema dimensional ora proposto. Usaremos como exemplo a modelagem proposta na Figura 3.2 para um pequeno negócio. Todos os passos para a construção do modelo dimensional do negócio e o seu povoamento, bem como as instanciações das tabelas gerais requeridas serão apresentadas.

Para a administração do data warehouse, povoamento e administração das tabelas gerais, que são funções que, dentro de nossa concepção, devem ser executadas por grupos distintos, foram concebidos 3 (três) módulos independentes, apesar de estarem aplicados a uma mesma base de dados. São eles:

a) **Administrador do data warehouse**: responsável pela criação e manutenção da hierarquia de dimensões e das tabelas de fatos correspondentes. Permite a definição do conteúdo dos metadados relacionados às tabelas de fato, bem como a estruturação dos seus atributos, com o estabelecimento de sua denominação e tipos de dados básicos;

b) **Povoador**: responsável pela instanciação dos fatos nas tabelas de fatos, situadas no nível folha da hierarquia de dimensões. O único método de povoamento possível é a digitação, sendo que não são tratadas as questões relacionadas ao *"data staging"* [Kimball & Ross 02];

c) **Administrador de Tabelas**: responsável pela criação e manutenção das tabelas gerais, permite a instanciação e a eliminação de seu conteúdo.

O protótipo foi implementado sobre a plataforma de desenvolvimento Delphi 6 e o sistema gerenciador de banco de dados MS-SQL 2000 [Battisti 01] [Ribeiro 01] [Shapiro 01] [Pacheco & Teixeira 02].

A partir do próximo tópico iniciaremos a apresentação do funcionamento dos módulos que compõem o protótipo através da modelagem de um pequeno negócio em informática, de acordo com a Figura 4.1.

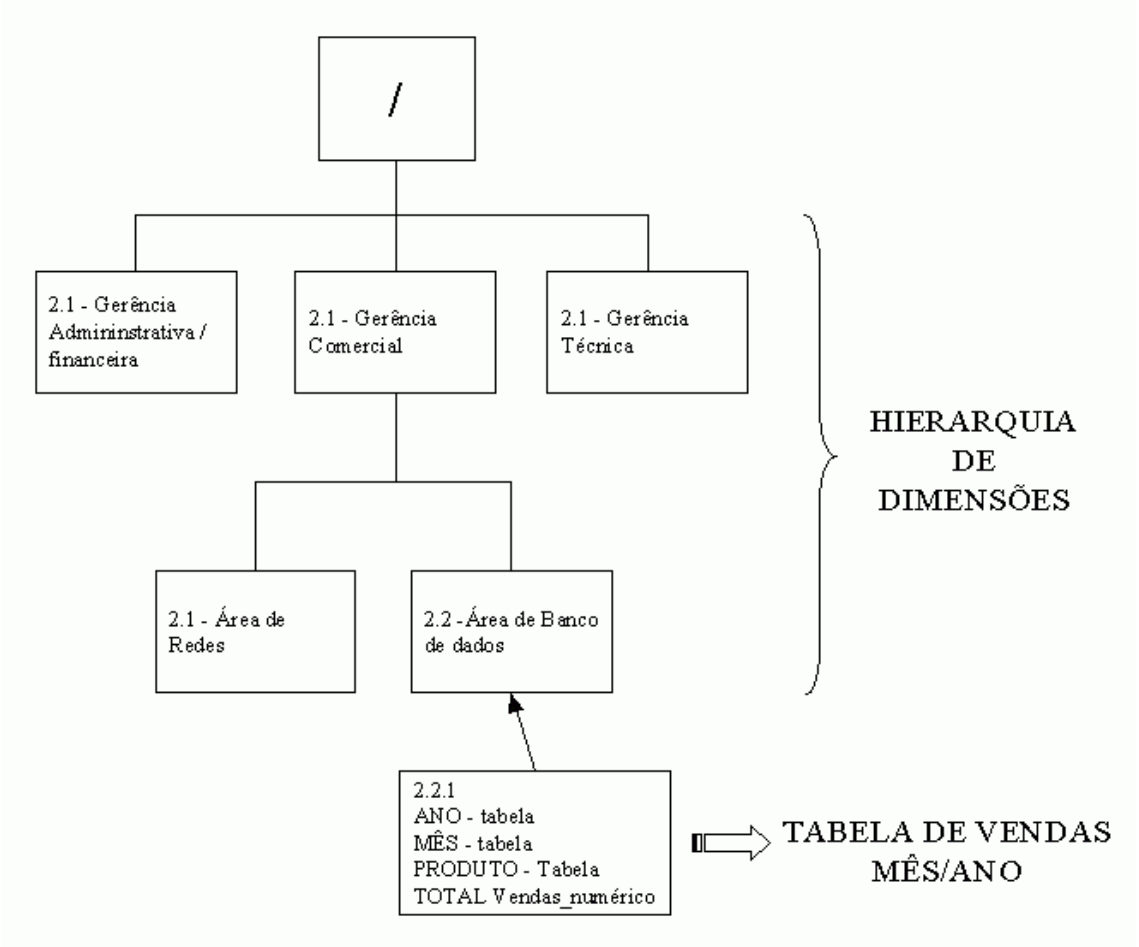

Figura 4.1 – Modelagem de um Pequeno Negócio

4.2 – O ADMINISTRADOR DO DATA WAREHOUSE

Imaginamos, de maneira clara, uma ferramenta que tivesse como objetivo específico a possibilidade da construção e manutenção da modelagem dimensional do negócio manipulada diretamente pelo usuário conhecedor de sua realidade corporativa ou departamental.

Esta ferramenta é denominada de administrador do data warehouse e, a seguir, acompanharemos através do modelo da Figura 4.1, os passos necessários para a criação e manutenção da hierarquia de dimensões e das tabelas de fatos envolvidas na modelagem.

Observe na Figura 4.2 (a) que o administrador do data warehouse é dividido em 4 (quatro) blocos de funções, na forma a seguir:

a) **Menu**: é dividido em 3 (três) rotinas. A primeira denominada **Ações** é composta pelas sub-rotinas NOVO, EXCLUIR, PROPRIEDADES, RENOMEAR, PROCURAR, IMPRIMIR e SAIR. São todas ações possíveis sobre a hierarquia de dimensões e tabela de fatos do exemplo tratado. A segunda denominada **Exibir** é composta pelas opções BARRA DE FERRAMENTAS, BARRA DE STATUS e ATUALIZA. A terceira denominada **Ajuda** é formada pela opção SOBRE O ADMINISTRADOR DO DATA WAREHOUSE;

b) **Botões de Atalho**: é composto por 4 (quatro) botões cujas funções são, na ordem: NOVO, EXCLUIR, PROPRIEDADES, PROCURAR E VISUALIZAR A IMPRESSÃO;

c) **Navegador do data warehouse**: é a área à esquerda da interface que exibe e permite a navegação sobre as dimensões e tabelas de fatos do data warehouse. Através do acionamento do botão direito do *"mouse"* pode-se obter

um menu *"pull-down"* que permite a utilização das opções relacionadas nos blocos anteriores;

d) **Área de Metadados e Fatos**: é a grande área àdireita da interface utilizada para inserção, manutenção e exibição dos metadados e fatos das tabelas de fatos assinaladas na área do item anterior. Permite a definição dos metadados e dos atributos das tabelas de fatos com a especificação de denominação, tipo básico, valores máximos, mínimos e a unidade de medida caso o atributo seja numérico.

4.2.1 – Inclusão da Hierarquia de Dimensões

Através da Figura 4.2 (a) podemos contemplar o estado inicial do data warehouse, estando representada apenas a dimensão raiz.

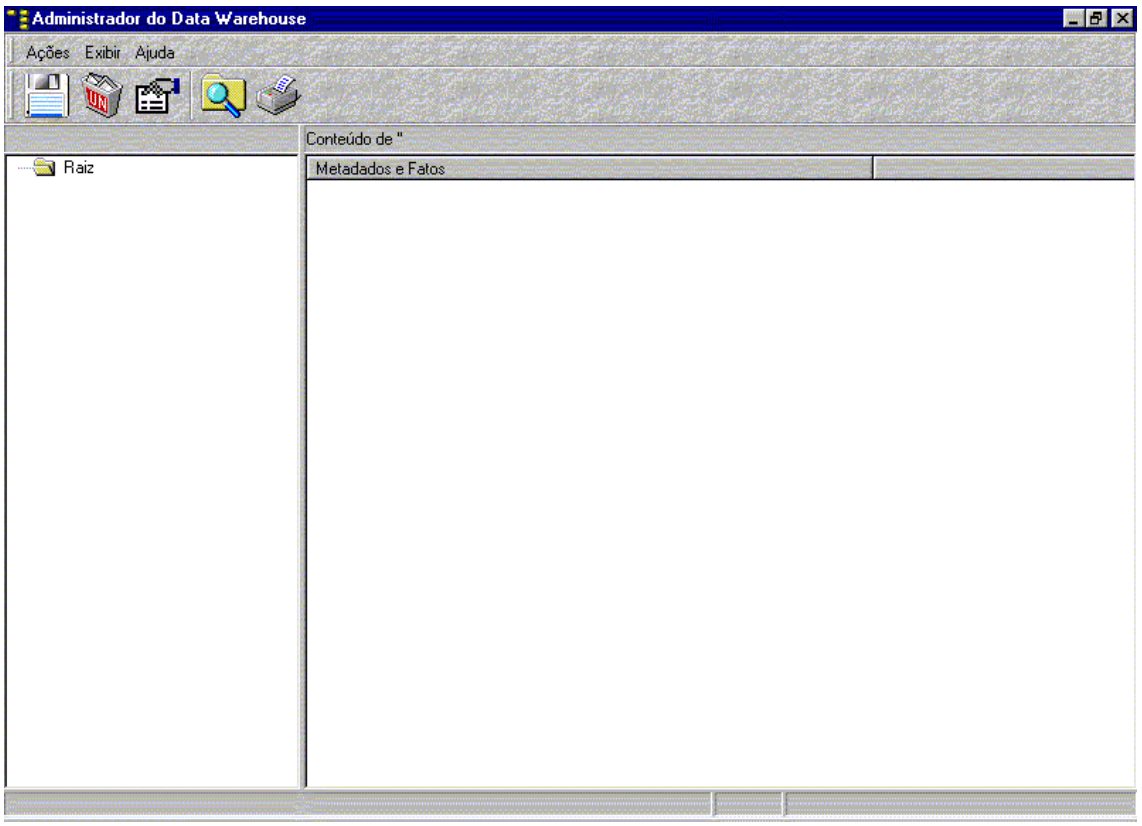

Figura 4.2 (a) – Data warehouse no momento inicial

Através da Figuras 4.2 (b) até (i) observamos a seqüência de ações necessárias para que as dimensões do negócio ilustrado em 4.1 sejam implementadas no data warehouse.

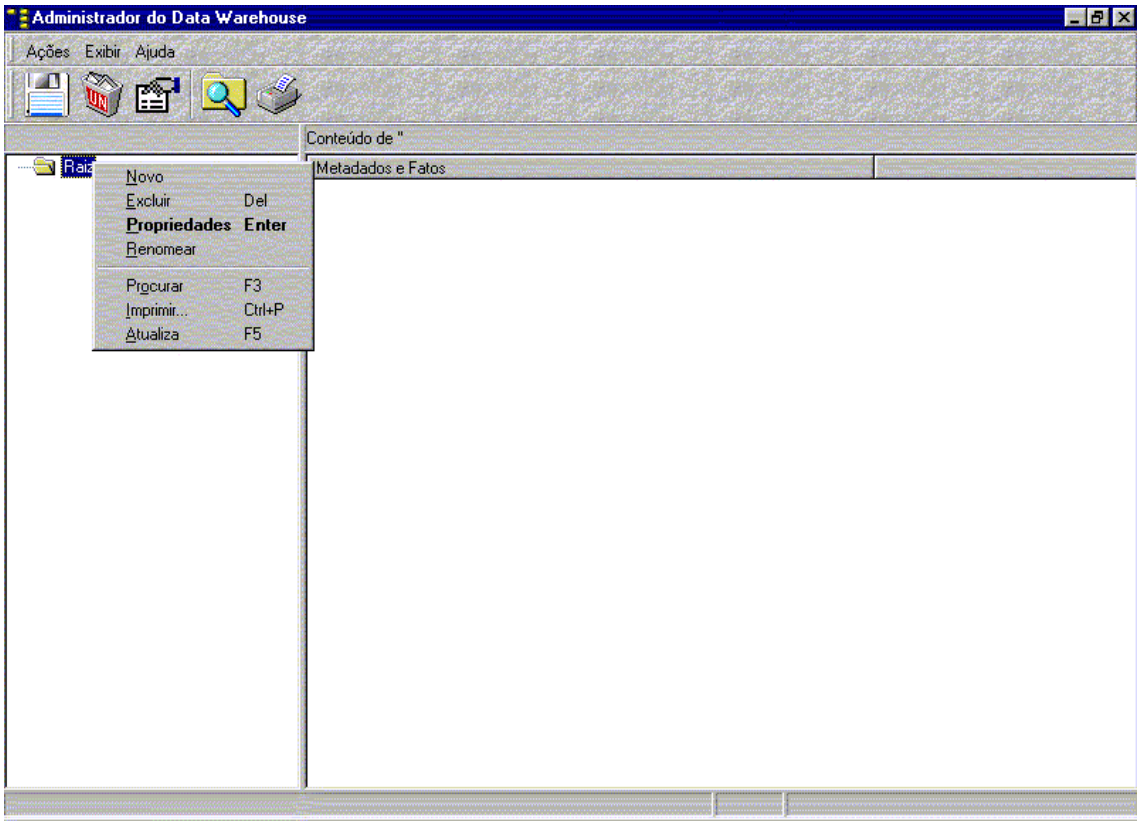

Figura 4.2 (b) – Preparando para inserir a primeira dimensão

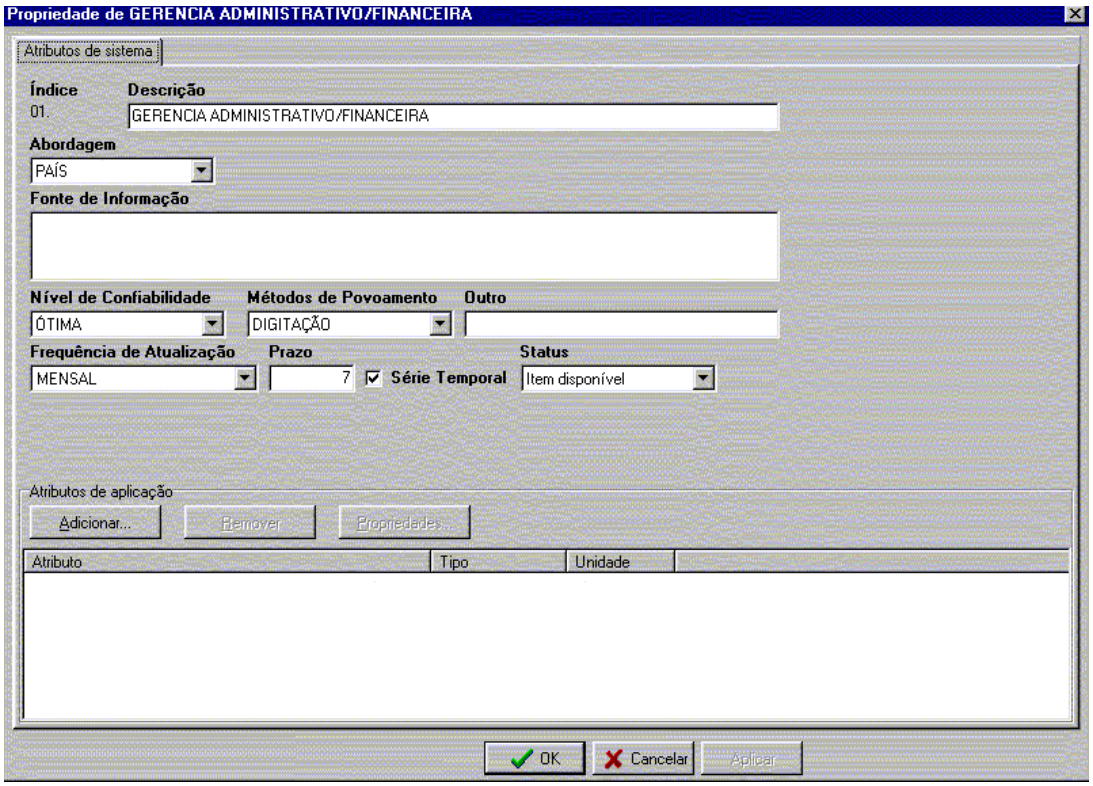

Figura 4.2 (c) – Metadados da dimensão Gerência Administrativo/Financeira

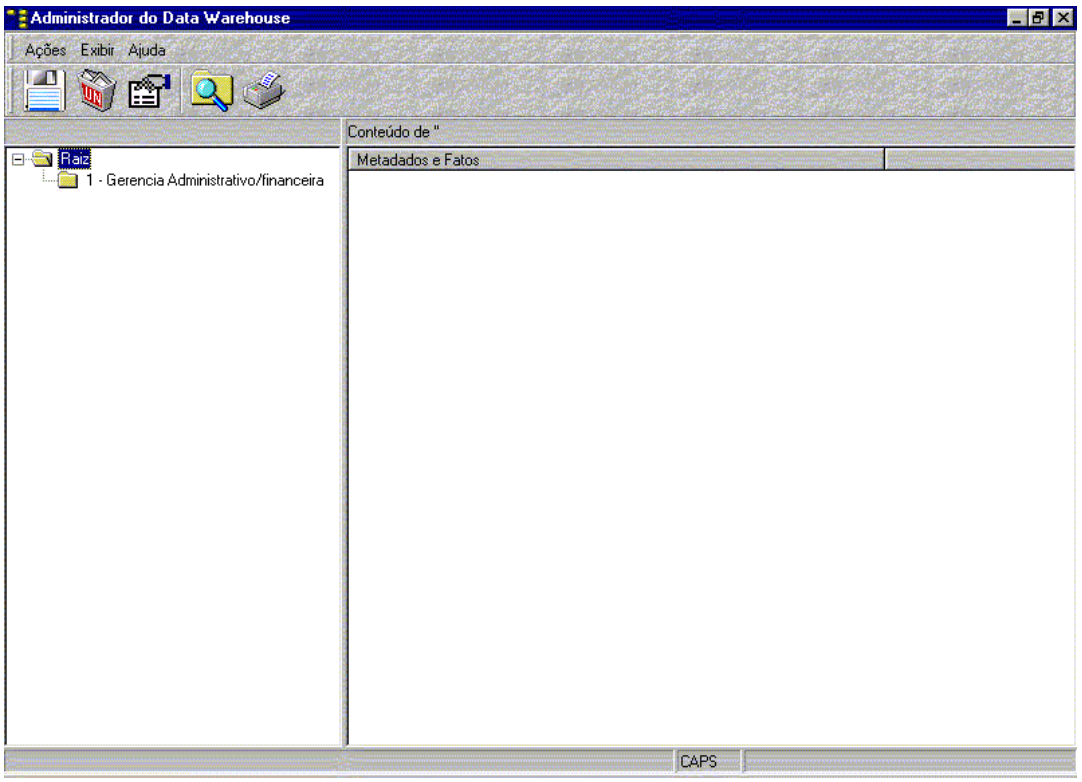

Figura 4.2 (d) – A dimensão Gerência Administrativo/Financeira inserida

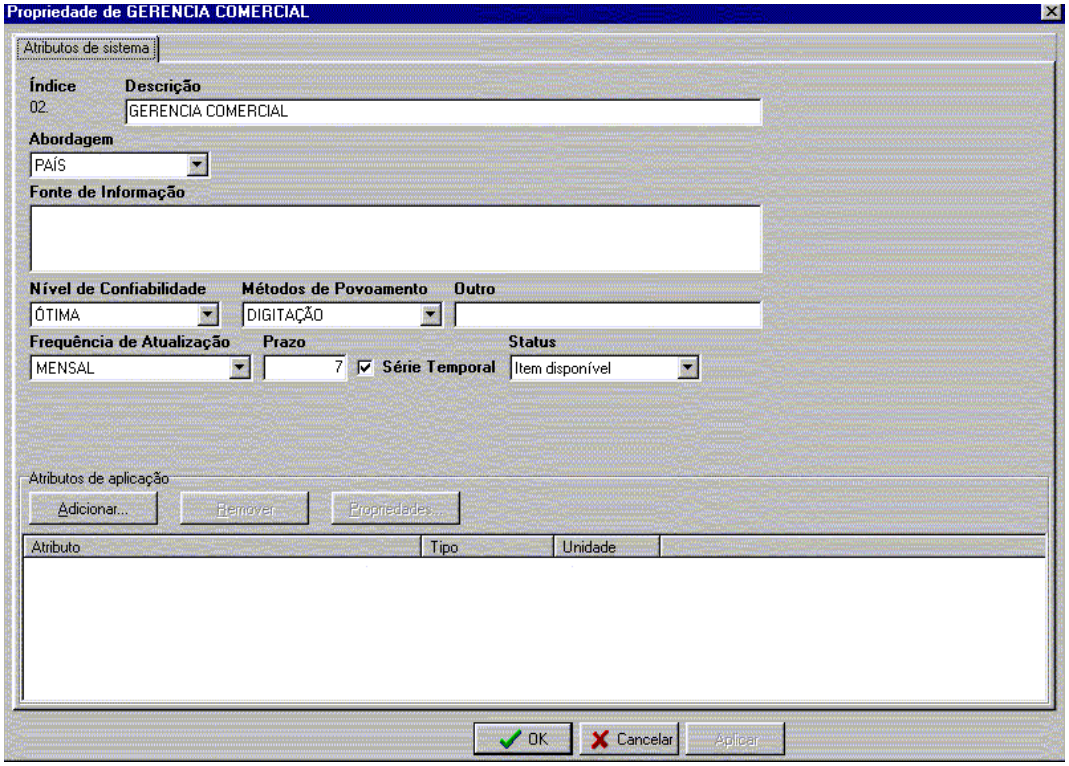

Figura 4.2 (e) – Metadados da dimensão Gerência Comercial

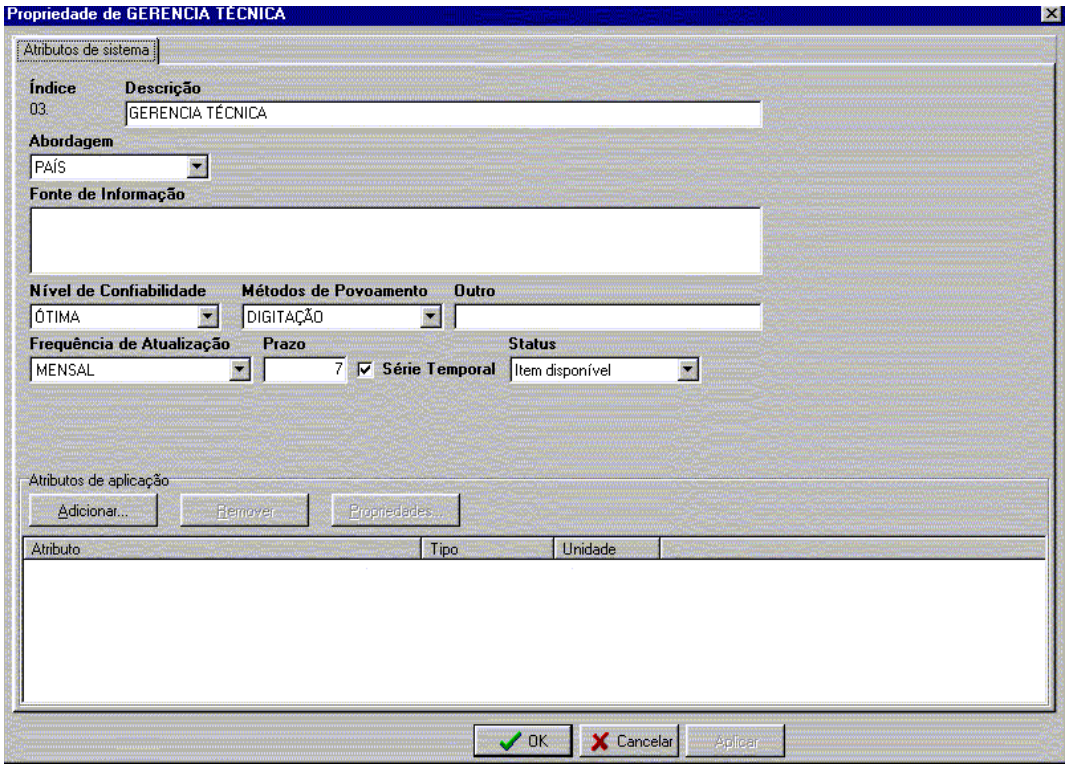

Figura 4.2 (f) – Metadados da dimensão Gerência Técnica

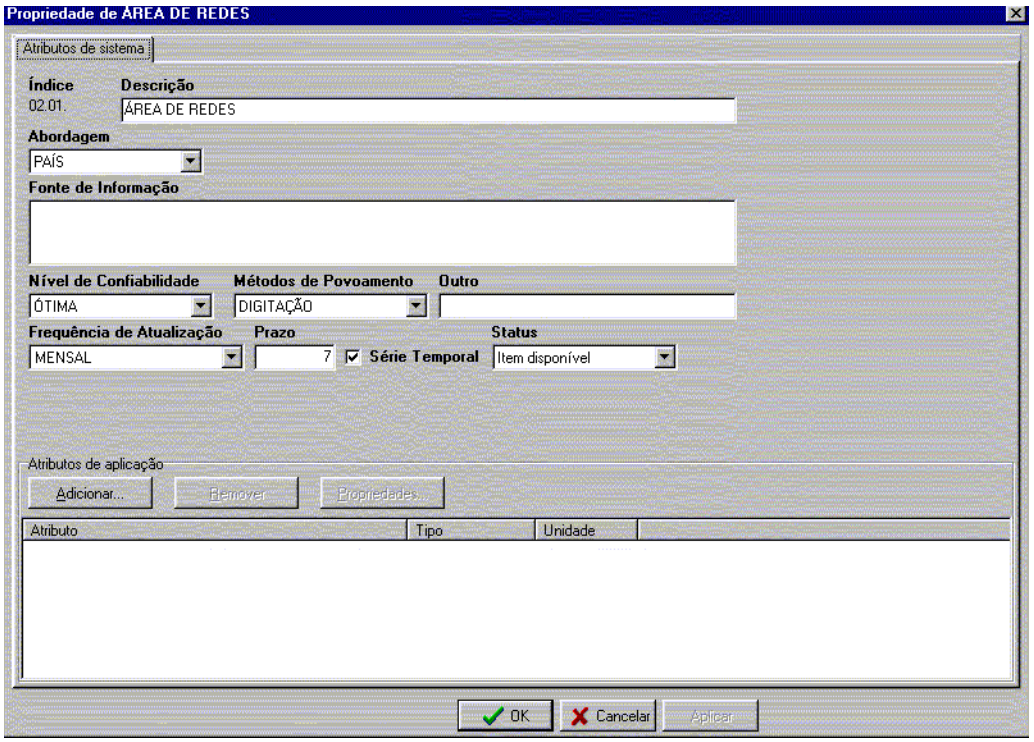

Figura 4.2 (g) – Metadados da dimensão Área de Redes

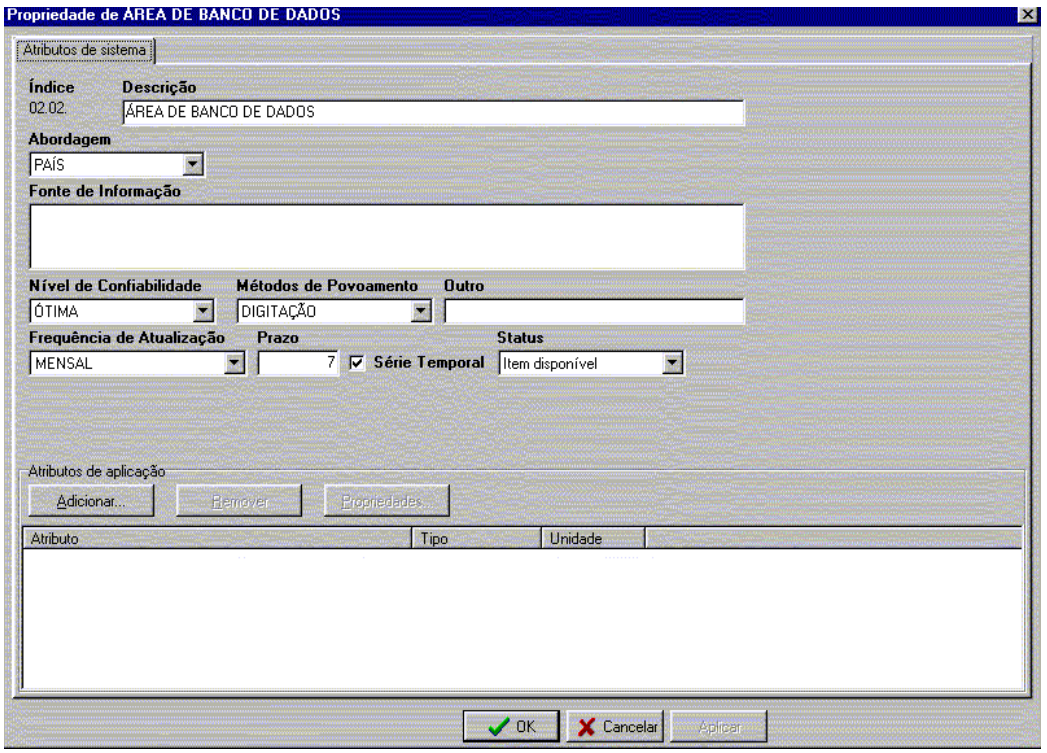

Figura 4.2 (h) – Metadados da dimensão Área de Banco de Dados

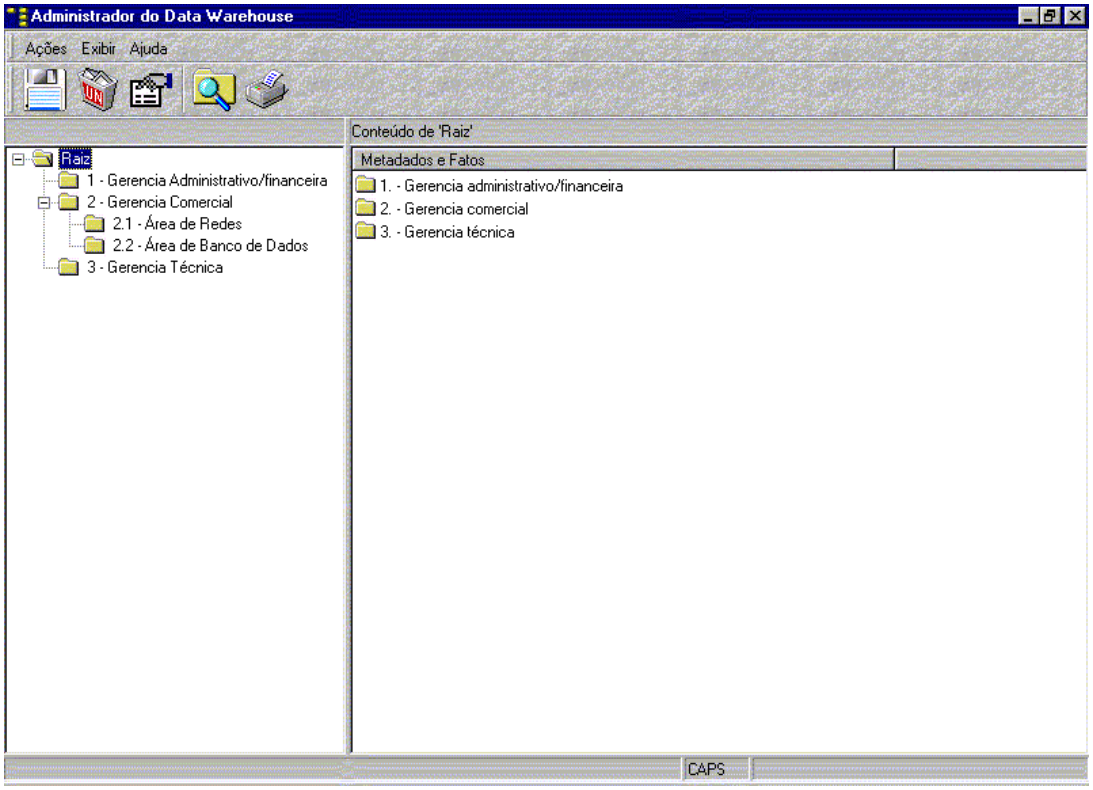

Figura 4.2 (i) – Visão final das dimensões do pequeno negócio da Figura 4.1

Observamos na Figura 4.2 (i) o negócio proposto completamente modelado através do administrador do data warehouse, à exceção da tabela de fatos VENDAS POR MÊS/ANO, que trataremos a seguir.

4.2.2 –Inclusão da Tabela de Fatos

A tabela de fatos, no exemplo tratado, é identificada pelo índice 2.2.1 – VENDAS POR MÊS/ANO e neste tópico será apresentada a sua inclusão no data warehouse através da ferramenta implementada. Para que isto ocorra, basta apontarmos para a sua dimensão imediatamente ascendente (2.2 – ÁREA DE BANCO DE DADOS) e pressionarmos o botão direito do *"mouse"* e escolhermos a opção NOVO do menu *"pull-down"* apresentado. Será aberto um formulário em que será possível incluir o conteúdo dos seus metadados e

definir a estrutura dos atributos que a compõem, conforme o apresentado nas Figuras 4.3 .

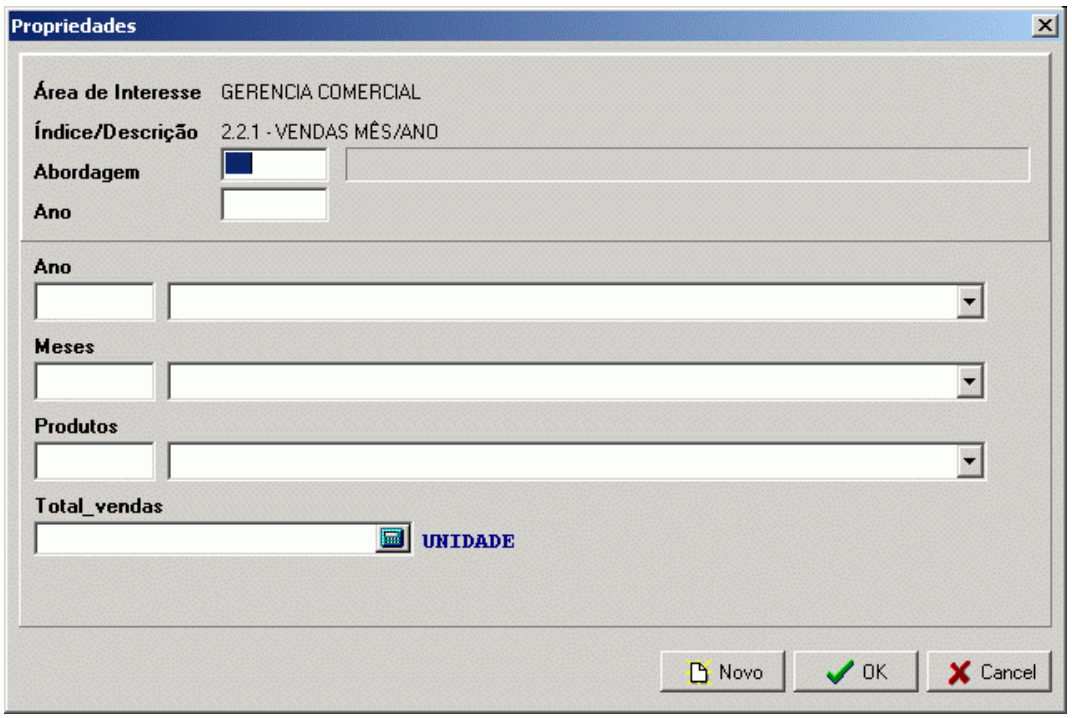

Figura 4.3 – Atributos da Tabela de Fatos VENDAS MÊS/ANO

4.2.3 – Listagem Geral da Estrutura do Data Warehouse

Na Figura 4.4 temos uma visão geral da estrutura de dimensões e fatos criada para a modelagem realizada.

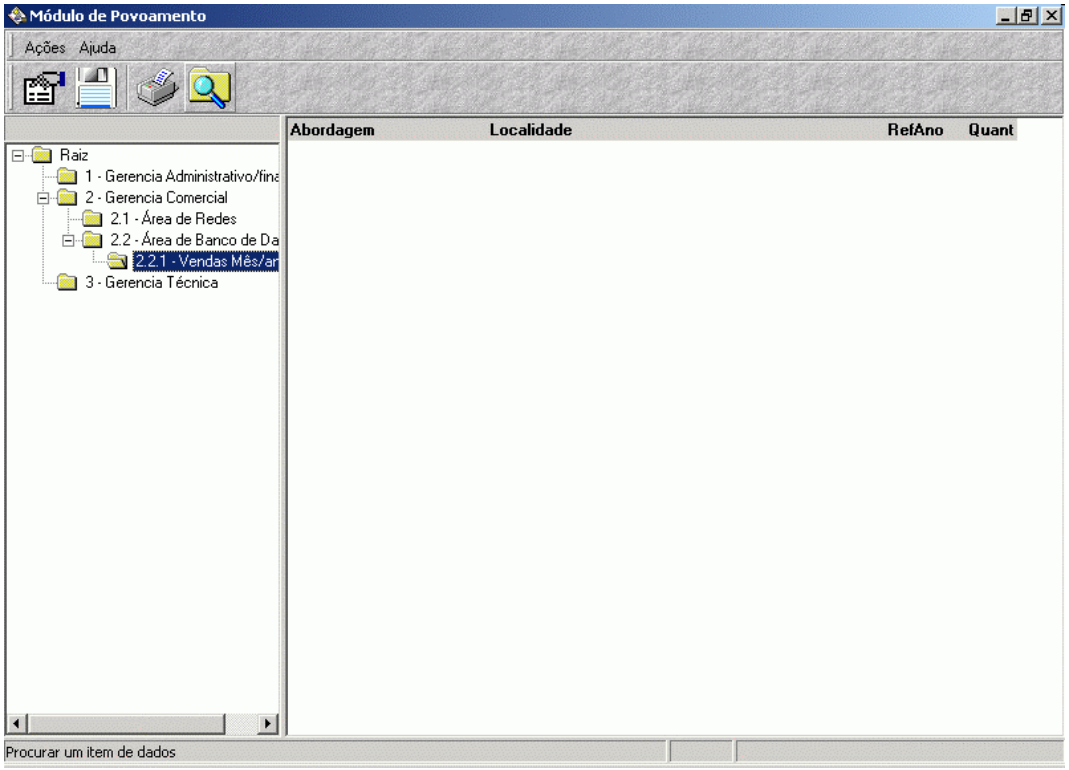

Figura 4.4 – Estrutura Geral de Dimensões e Fatos

4.3 – POVOADOR

Este é um módulo bastante simples, responsável pela instanciação dos valores no data warehouse. Como é de se supor não enfrenta as questões relacionadas à extração dos dados das fontes primitivas ou externas, limpeza e transformação apenas limitando-se à carga na sua forma mais simples. A Figura 4.5 apresenta uma etapa no povoamento dos dados do exemplo tratado.

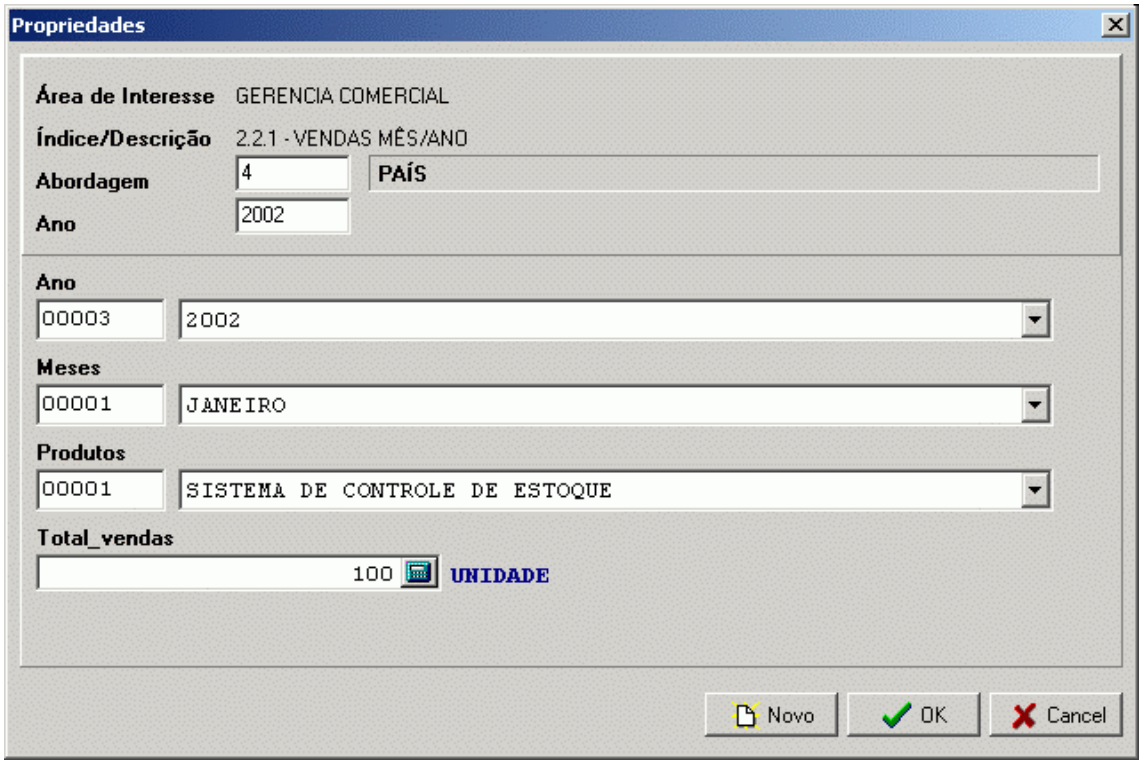

Figura 4.5 – Povoamento dos Fatos de um Evento de Vendas

4.4 – ADMINISTRADOR DE TABELAS

Este módulo tem funcionalidades combinadas com o administrador do data warehouse. Na verdade este complementa o conceito clássico de dimensões [Kimball & Ross 02] [Red Brick 95], produzindo e mantendo um conjunto da tabelas gerais que, na verdade, apresentam-se como alternativa para a inclusão das chaves das dimensões no esquema em estrela nas tabelas de fatos a estas relacionadas. As tabelas gerais são consideradas de sistema ou não. As tabelas de sistema são aquelas utilizadas para delimitar o conjunto de conteúdos possíveis para os metadados enquanto as demais delimitam o conjunto de conteúdos possíveis para as dimensões.

No que tange à recuperação de informações dentro do data warehouse, estes funcionam absolutamente em convergência ao apontado na bibliografia

pesquisada. Através dos mesmos pode-se obter as diferentes perspectivas do negócio.

No exemplo tratado utilizaremos a instanciação das tabelas de anos, meses e produtos para ilustrar o funcionamento da ferramenta em tela, de acordo com as Figuras 4.6 (a) (b) (c) (d).

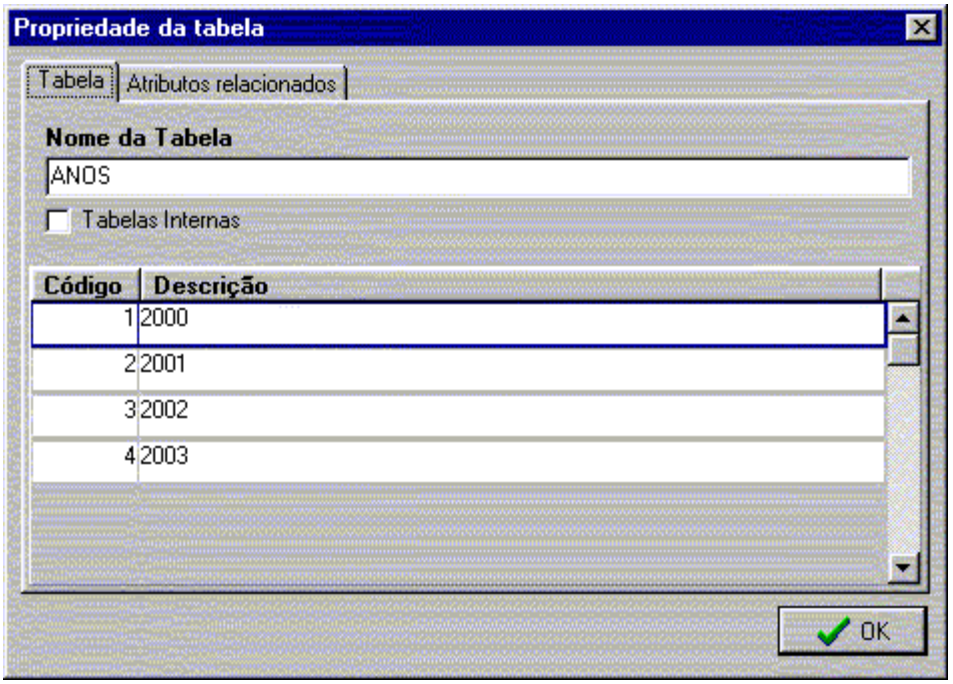

Figura 4.6 (a) – Apresentação das Tabela Geral de ANOS

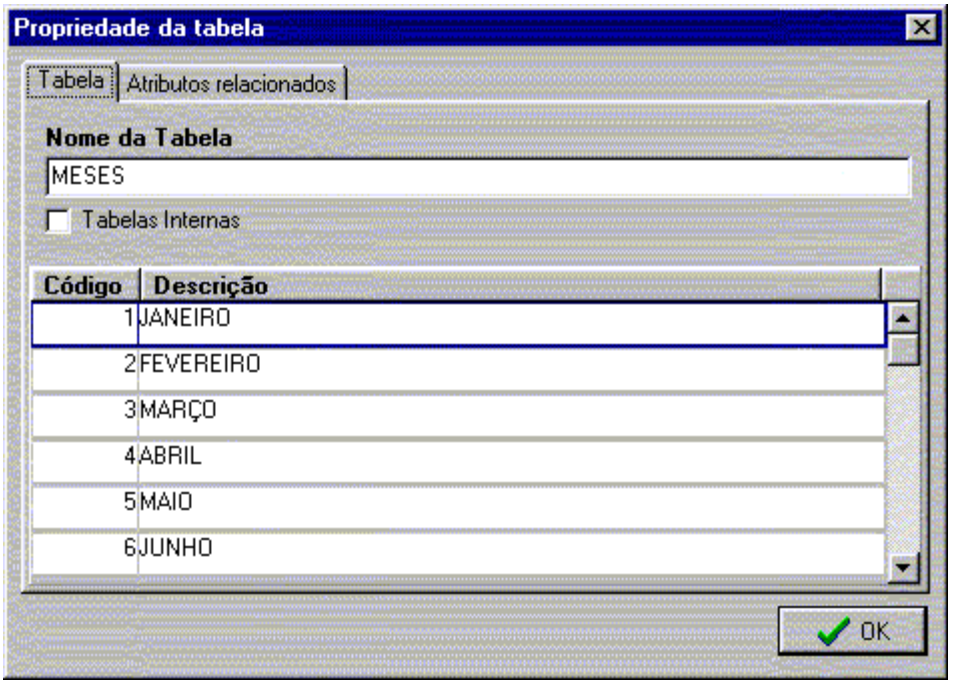

Figura 4.6 (b) – Apresentação das Tabela Geral de MESES

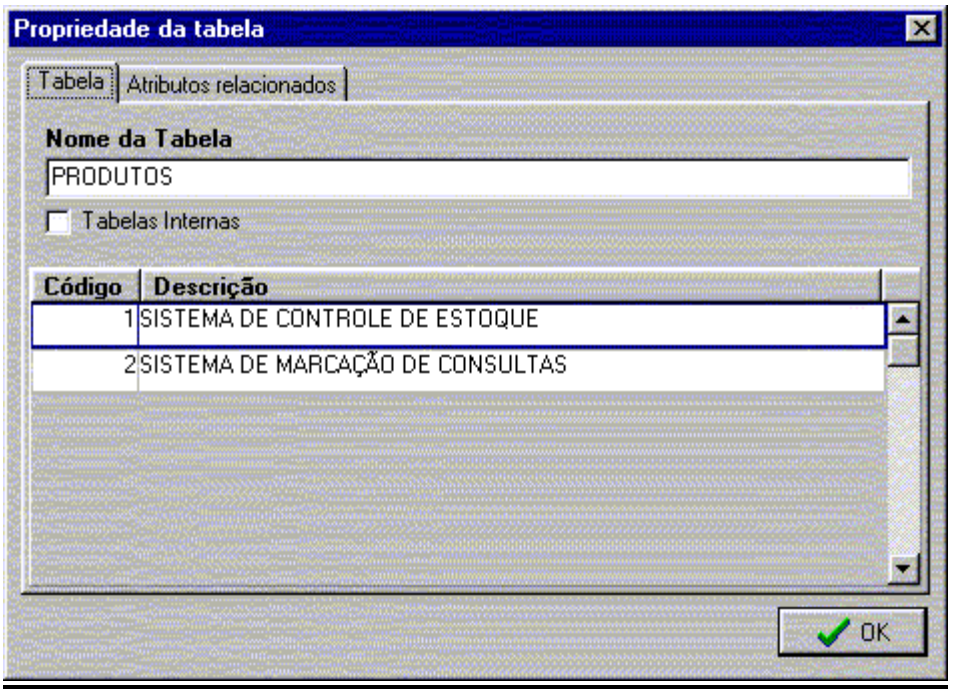

Figura 4.6 (c) – Apresentação das Tabela Geral de PRODUTOS

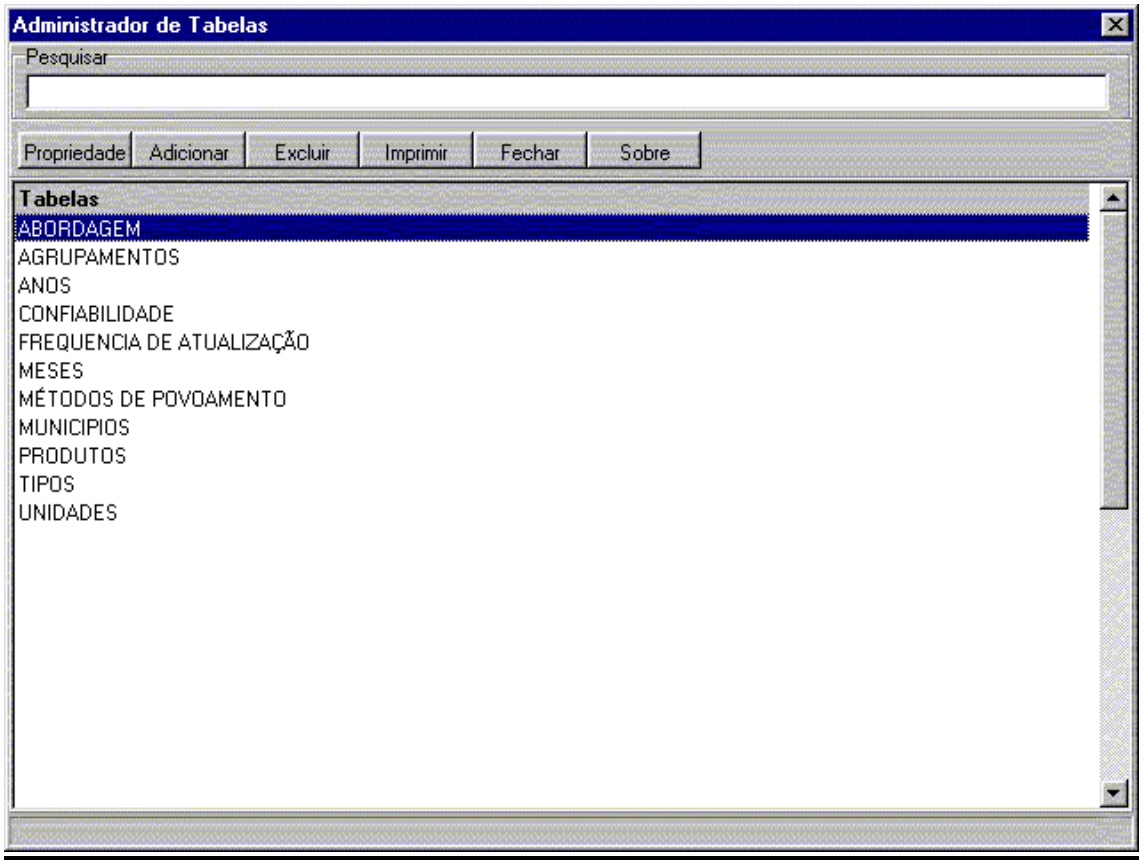

Figura 4.6 (d) – Apresentação das Tabelas Gerais

4.5 – CONSIDERAÇÕES FINAIS

Neste Capítulo apresentamos as ferramentas desenvolvidas para a implementação dos objetivos estabelecidos para este trabalho. Vimos que para a materialização do esquema proposto precisamos criar 3 (três) módulos distintos mas que trabalham sobre o mesmo banco de dados: o administrador de data warehouse, para criar e manter as dimensões e os fatos; o povoador para instanciar os fatos e; o administrador de tabelas para complementar o conceito de dimensões e permitir a instanciação dos atributos das dimensões dentro das tabelas de fato.

Baseamos a apresentação do protótipo em um modelo de negócio simples, mas podemos depreender que a sua versatilidade permitiria a sua utilização em um rol expressivo de outros negócios. O mais importante é que a representação da realidade do negócio é realizada de maneira intuitiva e familiar através de ferramentas de utilização bastante simples.

É bem verdade que uma série de especificações de data warehouse não estão representadas pela implementação e que há muito mais complexidade envolvida, tanto em etapas anteriores quanto em posteriores da própria estruturação do data warehouse. Dentro do âmbito das nossas limitações acreditamos que o esquema dimensional proposto é razoável, tendo surgido da inquietação de um profissional de Ciência da Computação, associada aos conceitos de data warehouse.

CAPíTULO 5

Conclusão

5.1 – OBJETIVOS ALCANÇADOS

Avaliando o comportamento e as características dos sistemas de informação e impulsionado pela idéia de aproximar os usuários finais das ferramentas de estruturação e manipulação de seus próprios dados, procuramos propor uma solução para modelagem de realidades de negócio que fosse de simples e familiar utilização e, de forma subjacente, que fosse implementada de uma maneira única sobre um mesmo modelo de dados, utilizando sempre as mesmas ferramentas e a mesma estrutura de banco de dados.

Fomos afastados imediatamente desta pretensão no que tange aos sistemas de informações transacionais pelos necessários requisitos operacionais do repositório de dados, haja vista a necessidade de garantia de sincronismo e integridade de dados providos por mecanismos de controle de concorrência, gerência de transações, recuperação pós-falha e outros. Assim, nos encaminhamos naturalmente aos sistemas de informações estratégicas e, por exigência da tecnologia para tratamento deste nível de informações, dos conceitos relacionados a data warehouse.

Dentro deste contexto e de acordo com os resultados alcançados acreditamos que conseguimos os seguintes objetivos:

a) Propusemos um esquema para modelagem dimensional implementável em sistemas gerenciadores de bancos de dados relacionais, que apresenta uma forma rígida – hierárquica – para acesso aos fatos. Isto garante uma compreensão natural e familiar do negócio manipulado por parte dos usuários finais, para os quais são estabelecidas balizas fixas para o caminhamento em direção aos dados armazenados;

b) Produzimos um projeto de banco de dados sobre o qual pode ser implementado diretamente o esquema dimensional concebido e que atende, sem modificações e de acordo com as simulações realizadas, a qualquer modelagem de negócios requerida;

c) Produzimos um conjunto de ferramentas de *"software"* que possibilita a interação natural do usuário final com a criação e manutenção da estrutura de seu armazém de dados.

5.2 – PROPOSTAS DE TRABALHOS FUTUROS

O desenvolvimento deste trabalho suscitou alguns pontos que, dentro de nossa própria observação, poderiam merecer maior estudo e reflexão e serem alvos de contribuições futuras. São eles:

a) O fato de não utilizarmos atributos na implementação do conceito de dimensões, seja através da hierarquia ou das tabelas gerais, pode causar críticas. Na verdade não conseguimos ainda estabelecer se a medida adotada em nosso trabalho foi a melhor opção, mas fomos encaminhados diretamente a ela procurando a abstração do conceito de chaves. Tal decisão contribui também para não haver a sistematização de todos os atributos relacionados a uma mesma dimensão sob uma mesma estrutura e torna a caracterização das dimensões mais pobre. Na verdade, eles podem se representados na solução, mas consideramos que não está na forma ideal. Um trabalho possível seria a criação de uma hierarquia de domínios para as tabelas gerais, dentro de uma mesma tabela, associada a uma hierarquia entre as próprias tabelas, refletindo uma tentativa de normalização da relação 1:N entre atributos de uma mesma

dimensão no esquema em estrela, simulando uma versão do esquema em flocos de neve;

b) Um outro trabalho poderia repousar sobre a opção de não trabalharmos com tipos atômicos para os fatos armazenados de forma distribuída, promovendo junção entre os mesmos quando da recuperação de informações. Poderíamos trabalhar com a criação de uma relação específica para cada tabela de fatos, através da utilização de cláusula SQL *"CREATE"*, formando os relacionamentos necessários com a relações At_Aplic e Tabelas. Um estudo sobre os resultados obtidos em nível de utilização de espaço em disco e desempenho na realização de junções poderia ser desenvolvido;

c) Observe que não são armazenados valores em Tipo_Tabela, mas apenas as referências para a relação Item_Tabela. Assim, para o processamento de consultas serão necessárias junções entre as relações Tipo e a relação Item_Tabela o que pode prejudicar o desempenho. Assim, poderíamos, ao invés de armazenar no Tipo_Tabela uma referência a Item_Tabela, armazenar os valores obtidos de Item_Tabela no momento do povoamento, o que eliminaria a necessidade de mais uma junção. No entanto, seria interessante estudar o impacto desta decisão no que tange ao espaço para armazenamento.

d) Um estudo geral sobre os requisitos de espaço para armazenamento e desempenho no processamento de consultas poderia ser desenvolvido, estabelecendo-se um comparativo com o obtido nos esquemas existentes;

e) A criação das ferramentas para a implementação das atividades relacionadas à extração, transformação e carga dos dados seria uma proposta

de trabalho bastante interessante uma vez que este aspecto da construção do data warehouse não foi contemplado neste trabalho.

CAPíTULO 6
BIBLIOGRAFIA

[Alter 92] ALTER, Steven. Information system: a management perspective. United States of América; Addison - Wesley Publishing Company, 1992.

[Andreatto 99] ANDREATTO, Ricardo. *Construindo um Data Warehouse e Analisando suas Informações com Data Mining e OLAP*. http://www.datawarehouses.hpg.ig.com.br/dw_cap4.htm pela Internet, acessado em 06/02/2003.

[Battisti 01] BATTISTI, Julio. *SQL Server 2000: Administração e Desenvolvimento: Curso Completo*. Axcel Books, 1ª Edição, 2001.

[Beuren et al 01] BEUREN, Ilse Maria; MARTINS, Luciano Waltrick. *Sistema de Informação Executiva: Suas características e Reflexões sobre sua Aplicação no Processo de Gestão*. Revista Contabilidade & Finanças FIPECAFI-FFA- USP. São Paulo; FIPECAFI, v. 15, n. 26. P – 6 – 24; maio/agosto 2001.

[Bliujute et al 98] BLIUJUTE, Rasa, SALTENIS, Simonas, SLIVINSKAS, Giedrius, JENSEN, Christian S., *Systematic Change Management in Dimensional Data Warehousing*. Proc. Third International Baltic Workshop on DB and IS. Abril 1998.

[Boggs 02] BOGGS, Wendy; BOGGS, Michael. *Mastering UML com Rational Rose 2000: A Bíblia*. Alta Books, 1ª Edição, 2002.

[Campos & Rocha 03 a] http://genesis.nce.ufrj.br/dataware/tutorial/arquit.html#tg_II3 pela Internet, acessado em 02/01/2003.

[Campos & Rocha 03 b] http://genesis.nce.ufrj.br/dataware/tutorial/dimens.html pela Internet, acessado em 06/02/2003.

[Campos 94] CAMPOS FILHO, Maurício Prates. *Os sistemas de informação e as modernas tendências da tecnologia e dos negócios*. Revista de Administração de Empresas. São Paulo, v.34, n-6, nov/dez 1994, p-33-45.

[Chauduri & Dayal 97] CHAUDURI, S.; DAYAL, U. *An Overview of Data Warehousing and OLAP Technology*. SIGMOD Record, Vol. 26, No 1. Setembro 1997.

[Chuck et al 98] CHUCK, Ballard; DIRK, Herreman; DON, Schav; RHONDA, Bell; EUNSAENG, Kim; ANN, Valencil. *Data Modeling Techniques for Data Warehousing*, Red Book, IBM International Technical Support Organization, http://www.redbooks.ibm.com. Fevereiro 1998.

[Data 03] http://www.terravista.pt/nazare/3187/BDDataWar.htm pela Internet, acessado em 03/01/2003.

[Dw Brasil 03] www.dwbrasil.com.br, pela Internet, acessado em janeiro/2003.

[Elmasri & Navathe 00] ELMASRI, Ramez; NAVATHE, Shamkant B. *Fundamentals of Database Systems*. United States of América: Addison – Wesley Publishing Company, 2000.

[Firestone 98] FIRESTONE, Joseph M. *Architectural Evolution in Data Warehousing and Distributed Knowledge Management Architecture*, White Paper, http://www.dkms.com/ARCHEV.html . Julho 1998.

[Flores 02 a] FLORES, Christian Feltrin. *Projeto e Desenvolvimento de Data Warehouse Hospitalar*. http://www.hcaa.com.br/antiga/dw/capa1131.htm pela Internet, acessado em 04/02/2003.

[Flores 02 b] FLORES, Christian Feltrin. *Projeto e Desenvolvimento de Data Warehouse Hospitalar*.http://www.hcaa.com.br/antiga/dw/capa1311.htm pela Internet, acessado em 03/01/2003.

[Fowler & Kendall 00] FOWLER, Martin; KENDALL, Scott. *UML Essencial*. Editora Bookman, 2ª Edição. 2000.

[Gardner 98] GARDNER, Stephen R. *Building the Data Warehouse*, Communications of the ACM, Vol 41, Nº 9. Setembro 1998.

[Golfarelli & Rizzi 98] GOLFARELLI, Matteo; RIZZI, Stefano. *A Methodological Framework for Data Warehouse Design*. Proc. DOLAP'98. 1998.

[Golfarelli et al 98] GOLFARELLI, Matteo; MAIO, Dario; RIZZI, Stefano, *Conceptual Design of Data Warehouses from E/R Schemas*. Janeiro 1998.

[Gupta 97] GUPTA, Vivek R. *An Introduction to Data Warehousing*, White Paper, System Services Corp., http://www.system-services.com. Agosto 1997.

[Han & Kamber 00] HAN, Jiawei; KAMBER, Micheline. *Data Mining – Concepts and Techniques*. Morgan Kaufmann Publis, 2000.

[Hand et al 01] HAND, David J; MANNILA, Heikkt; SMYTH, Padhraic. *Principles of Data Mining*. MIT PRESS, 2001.

[Harrison 98] HARRISON, Thomas H. *Intranet Data Warehouse*. São Paulo: Berkley Brasil, 1998.

[Inmon & Hackthorn 97] INMON, Wiliam H; HACKTHORN, Richard D. *Como usar o Data Warehouse.* Rio de Janeiro: INFOBOOK, 1997.

[Inmon & Welch 99] INMON, W, H.; WELCH, J. D.; Glassey, Katherine L. *Gerenciando Data Warehouse.* São Paulo: Makron Books, 1999.

[Inmon 97] INMON, W.H. *Como Construir o Data Warehouse.* Tradução da Segunda Edição, Editora Campus, 2ª Edição. 1997.

[Jacobson et al 00] JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. *UML: Guia do Usuário.* Editora Campus. 2000.

[Kenan 95] KENAN SYSTEMS CORPORATION, An Introduction to Multidimensional Database Technology. White Paper. 1995.

[Kimball 98 a] KIMBALL, Ralph. *Surrogates Keys*. Maio 1998.

[Kimball 98b] KIMBALL, Ralph. *Pipelining your Surrogates.* DBMS Magazine. Junho 1998.

[Kimball & Ross 02] KIMBALL, Ralph; ROSS, Margy. DATA WAREHOUSE TOOLKIT – O Guia Completo para a Modelagem Multidimensional. Rio de Janeiro: CAMPUS, 2002.

[Kimball 97] KIMBALL, Ralph., *A Dimensional Modeling Manifesto*. DBMS Magazine. Agosto 1997.

[Kimball et al 98] KIMBALL, Ralph; REEVES, Laura; ROSS, Margy; THORNWAITE, Warren. The Data Warehouse Lifecycle Toolkit : Experts Methods for Designing, Developing and Deploying Data Warehouses. John Wiley & Sons Inc. 1998.

[Lima 02] LIMA, Adilson da Silva. *Erwim 4.0: Modelagem de Dados*. Editora Érica, 1ª Edição, 2002.

[Melo 02] MELO, Ana Cristina. *Desenvolvendo Aplicações com UML*. Editora Brasport, 1ª Edição, 2002.

[Mendelzon 03] MENDELZON,Alberto. *Data Warehousing and OLAP*. http://www.cs.toronto.edu/~mendel/dwbib.html pela Internet, acessado em 04/02/2003.

[Ming & Bucchmann 97] MING, Chang Wu;BUCHMANN, Alejandro P. *Research Issues in Data Warehousing*. BTW, 1997.

[Orr 97] ORR, Ken. Data Warehousing Technologies – White Paper. The Ken Orr Institute, 1997.

[Pacheco & Teixeira 02] PACHECO, Xavier; TEIXEIRA, Steve. *Delphi 6: O Guia do Desenvolvedor*. Editora Campus, 1ª Edição, 2002.

[Peterson 94] PETERSON, Stephen. *Stars : A Pattern Language for Query Optimized Schema*, White Paper, Sequent Computer Systems Inc., http://c2.com/ppr/stars.html. 1994.

[Piatetsky-Shaphiro et al 96] PIATETSKY-SHAPHIRO,G; SMYTH,P; FAYYAD, Usama. *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[Poe et al 98] POE, Vidette; KLAUER, Patricia; BROBST, Stephen. *Building a Data Warehouse for Decision Support*. Prentice Hall PTR. 1998.

[Power 99] POWER, D. J. *A Brief History of Decision Support Systems*, White Paper, DSS Resources, http://dss.cba.uni.edu/dss/dsshistory.html. 1999.

[Quatrani 01] QUATRANI, Terri. *Modelagem Visual com Rational Rose 2000 e UML*. Editora Ciência Moderna, 1ª Edição, 2001.

[Ravat et al 99] RAVAT, Franck; TESTE, Olivier; ZURFLUH, Gilles. *Towards Data Warehouse Design*. Proc. 8th Int'l Conference on Information Knowledge Management (CIMK). Novembro 1999.

[Red Brick 95] RED BRICK SYSTEMS. *Star Schemas and Star Join Technology*. White Paper. Setembro 1995.

[Ribeiro 01] RIBEIRO, Ivanilson Lima. *Delphi 6 com SQL Server 2000.*Editora Érika, 1ª Edição, 2001.

[Rocha 00] ROCHA, André Barbosa, *Guardando Históricos de Dimensões em Data Warehouses*, Dissertação de

Mestrado, Departamento de Sistemas e Computação, Universidade Federal da Paraíba. Fevereiro 2000.

[Sen & Jacob 98] SEN, Arun; JACOB, Varghese S. Industrial-strength : Data Warehousing, Why Process is so Important- and so often ignored. Communications of the ACM, Vol. 41, Nº 9. Setembro 1998.

[Shapiro 01] SHAPIRO, Jeffrey R. *SQL Server 2000: Completo e Total*. Makron Boooks, 1ª Edição, 2001.

[Silberschatz et al 99] SILBERSCHATZ, Abraham; KORTH, Henry F. e SUDARSHAN, S. *Sistema de Banco de Dados São Paulo.* MAKRON BOOKS, 1999.

[Souza 99] SOUZA, Aídre da Cunha Guedes de. *Ambiente Computacional para Gerência Estratégica – Data Warehousing: Uma Survey*, Dissertação de Mestrado, Departamento de Sistemas e Computação, Universidade Federal da Paraíba. Abril 1999.

[Sprague & Watson 91] SPRAGUE, Ralph H., WATSON, Hugh J., “Sistema de Apoio à Decisão : Colocando a Teoria em Prática”. Editora Campus, 1991.

[Thomsen 97] THOMSEN, Erik. OLAP Solutions : Building Multidimensional Information Systems. John Wisley & Sons Inc., USA. 1997.

[Thomsen 97] THOMSEN, Erik. OLAP Solutions : Building Multidimensional Information Systems. John Wisley & Sons Inc., USA. 1997.

[Unicamp 98] http://www.revista.unicamp.br/infotec/informacao/inf54.htm pela Internet, acessado em 04/02/2003.

APÊNDICE A

Scripts para a Criação do Banco de Dados

```
CREATE TABLE [dbo].[At_Aplic] (
	[id_Atributo] [DInteiro] NOT NULL ,
	[Seq_Atributo] [int] NOT NULL ,
	[Desc_Atributo] [varchar] (100) NOT NULL ,
	[Tipo_Dado] [int] NOT NULL ,
	[Id_Tabela] [int] NULL ,
	[vl_min] [DNumero] NULL ,
	[vl_max] [DNumero] NULL ,
	[id_Unidade] [int] NULL
)
GO

CREATE TABLE [dbo].[At_Sistema] (
	[id_Atributo] [DInteiro] IDENTITY (1, 1) NOT NULL ,
	[id_Pai] [DInteiro] NULL ,
	[Indice] [DIndice] NULL ,
	[Id_Abordagem] [int] NULL ,
	[Nom_Nivel] [varchar] (100) NOT NULL ,
	[Fonte] [DString] NULL ,
	[id_Confiabilidade] [int] NULL ,
	[id_Povoamento] [int] NULL ,
	[Tam_Prazo] [DInteiro] NULL ,
	[Outra] [DString] NULL ,
	[id_Atulizacao] [int] NULL ,
	[Ser_Temporal] [char] (1) NOT NULL ,
	[Status] [char] (1) NULL
)
GO

CREATE TABLE [dbo].[Dado_Local] (
	[id_Atributo] [DInteiro] NOT NULL ,
	[Seq_Atributo] [int] NOT NULL ,
```

```
	[id_local] [int] NOT NULL ,
	[ref_temp] [datetime] NOT NULL ,
	[Num_Registro] [int] NOT NULL ,
	[Seq_dado] [int] IDENTITY (1, 1) NOT NULL
)
GO

CREATE TABLE [dbo].[Item_Tabela] (
	[Id_Tabela] [int] NOT NULL ,
	[Cod_Item] [int] NOT NULL ,
	[Vl_Item] [varchar] (100) NOT NULL
)
GO
```

```
CREATE TABLE [dbo].[Tabelas] (
	[Id_Tabela] [int] IDENTITY (1, 1) NOT NULL ,
	[Nom_Tabela] [DString] NOT NULL ,
	[Sistema] [char] (1) NOT NULL
)
GO

CREATE TABLE [dbo].[Tipo_Data] (
	[Seq_Dado] [int] NOT NULL ,
	[Valor] [datetime] NULL
)
GO

CREATE TABLE [dbo].[Tipo_Descritivo] (
	[Seq_Dado] [int] NOT NULL ,
	[Valor] [DString] NOT NULL
)
GO

CREATE TABLE [dbo].[Tipo_Numero] (
	[Seq_Dado] [int] NOT NULL ,
	[Valor] [DNumero] NOT NULL
)
GO

CREATE TABLE [dbo].[Tipo_Objeto] (
	[Seq_Dado] [int] NOT NULL ,
	[Valor] [image] NULL
)
GO

CREATE TABLE [dbo].[Tipo_Tabela] (
```

```
	[Seq_Dado] [int] NOT NULL ,
	[Cod_Item] [int] NOT NULL ,
	[Id_Tabela] [int] NOT NULL
)
GO

CREATE TABLE [dbo].[Tipo_Texto] (
	[Seq_Dado] [int] NOT NULL ,
	[Valor] [text] NOT NULL
)
GO
```

APÊNDICE B

Stored Procedures

Nome	CriaSeq_Anos
Parâmetros de Entrada	@Inicio – Ano inicial. @Fim – Ano final.
Descrição	Esta procedure atualiza a tabela SEQ_ANOS inserindo a seqüência de todos os anos compreendidos entre os parâmetros informados.

Código Fonte:

```
CREATE PROCEDURE CriaSeq_Anos @Inicio Int, @Fim Int AS
Declare
  @Cont Int
  Select @Cont = @Inicio
  While @Cont <= @Fim
  Begin
    If not Exists(Select * from Seq_anos Where Ano = @Cont)
    Begin
      Insert Into Seq_anos(Ano) Values(@Cont)
    End
    Select @Cont = @Cont + 1
  End
GO
```

Nome	GETID
Parâmetros de Entrada	@INDICE – Índice da dimensão ou fato.
Parâmetros de Saída	@ID – Identificador da dimensão ou fato.
Descrição	Retorna o identificador da dimensão ou fato em função do seu índice.

Código Fonte:

```
CREATE PROCEDURE GETID @INDICE VARCHAR(20), @ID INT OUTPUT
AS
DECLARE
  @RESTO  VARCHAR(20),
  @NUM    VARCHAR(20),
  @LEN    INT,
  @POS    INT
  SELECT @RESTO = @INDICE + '.'
  SELECT @POS = PATINDEX('%.%', @RESTO)
  SELECT @LEN = LEN(@RESTO)
  SELECT @NUM = SUBSTRING(@RESTO, 1, @POS - 1)
  SELECT @RESTO = SUBSTRING(@RESTO, @POS + 1, @LEN - @POS)
  SELECT @ID = ID_ATRIBUTO FROM VINDICE
    WHERE ID_PAI IS NULL AND NUM = @NUM
  WHILE LEN(@RESTO) > 0
  BEGIN
    SELECT @POS = PATINDEX('%.%', @RESTO)
    SELECT @LEN = LEN(@RESTO)
    SELECT @NUM = SUBSTRING(@RESTO, 1, @POS - 1)
    SELECT @RESTO = SUBSTRING(@RESTO, @POS + 1, @LEN - @POS)
    SELECT @ID = ID_ATRIBUTO FROM VINDICE
      WHERE ID_PAI = @ID AND NUM = @NUM
  END
  Return 1
GO
```

Nome	GETINDICE
Parâmetros de Entrada	@ID – Identificador da dimensão ou fato.
Parâmetros de Saída	@INDICE – Índice da dimensão ou fato.
Descrição	Retorna o índice da dimensão ou fato em função do seu identificador.

Código Fonte:

```
CREATE PROCEDURE GETINDICE @ID INT, @INDICE VARCHAR(20) OUTPUT AS
DECLARE
 @PAI INT, @NUM VARCHAR(120)

 SELECT @INDICE = ''
 WHILE @ID IS NOT NULL
 BEGIN
  SELECT @NUM = LTRIM(STR(NUM)), @PAI = ID_PAI
   FROM VINDICE
    WHERE ID_ATRIBUTO = @ID
  IF lEN(@NUM) = 1 SELECT @NUM = '0' + @NUM
  SELECT @INDICE = @NUM + '.' + @INDICE
  SELECT @ID = @PAI
 END
 Return 1
GO
```

Nome	PRO_ÍNDICE
Descrição	Atualiza os índices da hierarquia de dimensões.

Código Fonte:

```
CREATE PROCEDURE PRO_INDICE AS
DECLARE
  @id_Atributo INTEGER,
  @id_Pai      INTEGER,
  @Nom_Nivel   VARCHAR(100),
  @Indice      VarChar(20)

DECLARE Cur_Indice CURSOR FOR
  SELECT id_Atributo, id_Pai, Nom_Nivel
  FROM At_Sistema
OPEN Cur_Indice
FETCH NEXT FROM Cur_Indice
INTO @id_Atributo, @id_Pai, @Nom_Nivel
WHILE @@FETCH_STATUS = 0
BEGIN
  EXEC GETINDICE @id_Atributo, @Indice OUTPUT
  UPDATE AT_SISTEMA
  SET INDICE = @Indice
  WHERE id_Atributo = @id_Atributo

  FETCH NEXT FROM Cur_Indice
  INTO @id_Atributo, @id_Pai, @Nom_Nivel
END
CLOSE Cur_Indice
DEALLOCATE Cur_Indice
RETURN
GO
```

Nome	PROXIMO_NUM
Descrição	Retorna o próximo número de registro para instanciar um fato.

Código Fonte:

```
CREATE PROCEDURE [Proximo_Num] AS
Declare
  @Proximo int
Select @Proximo = (COALESCE(Max(Num_Registro), 0) + 1) from Dado_Local
Return @Proximo
GO
```

APÊNDICE C

VISÕES

Nome da Visão	Vabordagem
Descrição	Faz uma seleção sobre a relação ITEM_TABELA a fim de obter os código e descrição dos tipos de abordagem.

Código Fonte:

```
CREATE VIEW VAbordagem (Cod_Item, Vl_Item)  AS
    SELECT Item_Tabela.Cod_Item, Item_Tabela.Vl_Item
    FROM Item_Tabela
    WHERE Id_Tabela = 4
GO
```

Nome da Visão	VConfiabilidade
Descrição	Faz uma seleção sobre a relação ITEM_TABELA a fim de obter os código e descrição dos tipos de confiabilidade.

Código Fonte:

```
CREATE VIEW VConfiabilidade (Cod_Item, Vl_Item)  AS
    SELECT Item_Tabela.Cod_Item, Item_Tabela.Vl_Item
    FROM Item_Tabela
    WHERE Id_Tabela = 3
GO
```

Nome da Visão	VDados
Descrição	Faz uma seleção sobre a relação ITEM_TABELA a fim de obter o código e descrição dos tipos de Dados.

Código Fonte:

```
CREATE VIEW VDados (Cod_Item, Vl_Item)  AS
    SELECT Item_Tabela.Cod_Item, Item_Tabela.Vl_Item
    FROM Item_Tabela
    WHERE Id_Tabela = 5
GO
```

Nome da Visão	VFreqAtu
Descrição	Faz uma seleção sobre a relação ITEM_TABELA a fim de obter o código e descrição dos tipos de freqüência de atualização.

Código Fonte:

```
CREATE VIEW VFreqAtu (Cod_Item, Vl_Item)  AS
    SELECT Item_Tabela.Cod_Item, Item_Tabela.Vl_Item
    FROM Item_Tabela
    WHERE Id_Tabela = 80
GO
```

Nome da Visão	Vlocais
Descrição	Faz uma seleção sobre a relação ITEM_TABELA a fim de obter o código e descrição dos locais de abrangência.

Código Fonte:

```
CREATE VIEW VLocais  AS
        SELECT VMunicipio.Cod_Item, VMunicipio.Vl_Item, 'M' Abrangencia
        FROM VMunicipio union
           SELECT 0, 'Piauí', 'E'
GO
```

Nome da Visão	VPov
Descrição	Faz uma seleção sobre a relação ITEM_TABELA a fim de obter o código e descrição dos métodos de povoamento.

Código Fonte:

```
CREATE VIEW VPov (Cod_Item, Vl_Item)  AS
     SELECT Item_Tabela.Cod_Item, Item_Tabela.Vl_Item
     FROM Item_Tabela
     WHERE Id_Tabela = 2
GO
```

Nome da Visão	VPovoamento
Descrição	Retorna todos os dados povoados.

Código Fonte:

```
CREATE VIEW dbo.VPovoamento
AS
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
    Dado_Local.id_local, (CASE WHEN MONTH(ref_temp) <> 1 THEN '' ELSE
STR(YEAR(ref_temp)) END) ANO,
  Dado_Local.Num_Registro, Tipo_Descritivo.Valor
FROM Dado_Local INNER JOIN
  Tipo_Descritivo ON
  Dado_Local.Seq_dado = Tipo_Descritivo.Seq_Dado
union all
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
  Dado_Local.id_local, (CASE WHEN MONTH(ref_temp) <> 1 THEN '' ELSE
STR(YEAR(ref_temp)) END) ANO,
  Dado_Local.Num_Registro, Ltrim(Str(Tipo_Numero.Valor)) Valor
FROM Dado_Local INNER JOIN
  Tipo_Numero ON
  Dado_Local.Seq_dado = Tipo_Numero.Seq_Dado
union all
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
    Dado_Local.id_local, (CASE WHEN MONTH(ref_temp) <> 1 THEN '' ELSE
STR(YEAR(ref_temp)) END) ANO,
  Dado_Local.Num_Registro,
  Item_Tabela.Vl_Item AS Valor
FROM Dado_Local INNER JOIN
  Tipo_Tabela ON
```

```
   Dado_Local.Seq_dado = Tipo_Tabela.Seq_Dado LEFT OUTER JOIN
   Item_Tabela ON
   Tipo_Tabela.Id_Tabela = Item_Tabela.Id_Tabela AND
   Tipo_Tabela.Cod_Item = Item_Tabela.Cod_Item
union all
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
    Dado_Local.id_local, (CASE WHEN MONTH(ref_temp) <> 1 THEN '' ELSE
STR(YEAR(ref_temp)) END) ANO,
   Dado_Local.Num_Registro,
(ltrim(Str(day(valor))) + '/' + ltrim(Str(month(valor))) + '/' + ltrim(Str(year(Valor))))
as Valor
FROM Dado_Local INNER JOIN
   Tipo_Data ON Dado_Local.Seq_dado = Tipo_Data.Seq_Dado
union all
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
    Dado_Local.id_local, (CASE WHEN MONTH(ref_temp) <> 1 THEN '' ELSE
STR(YEAR(ref_temp)) END) ANO,
   Dado_Local.Num_Registro, ''
FROM Dado_Local INNER JOIN
   Tipo_Objeto ON
   Dado_Local.Seq_dado = Tipo_Objeto.Seq_Dado
union all
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
    Dado_Local.id_local, (CASE WHEN MONTH(ref_temp) <> 1 THEN '' ELSE
STR(YEAR(ref_temp)) END) ANO,
   Dado_Local.Num_Registro, convert(Varchar(8000), Valor)
FROM Dado_Local INNER JOIN
   Tipo_Texto ON Dado_Local.Seq_dado = Tipo_Texto.Seq_Dado
GO
```

Nome da Visão	VTipo_Data
Descrição	Retorna todos os dados povoados do tipo data.

Código Fonte:

```
CREATE VIEW dbo.VTipo_Data
AS
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
   Dado_Local.id_local, Dado_Local.ref_temp,
   Dado_Local.Num_Registro, Tipo_Data.Valor
FROM Dado_Local INNER JOIN
   Tipo_Data ON Dado_Local.Seq_dado = Tipo_Data.Seq_Dado
GO
```

Nome da Visão	VTipo_Descritivo
Descrição	Retorna todos os dados povoados do tipo caracter.

Código Fonte:

```
CREATE VIEW dbo.VTipo_Descritivo
AS
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
   Dado_Local.id_local, Dado_Local.ref_temp,
   Dado_Local.Num_Registro, Tipo_Descritivo.Valor
FROM Dado_Local INNER JOIN
   Tipo_Descritivo ON
   Dado_Local.Seq_dado = Tipo_Descritivo.Seq_Dado
GO
```

Nome da Visão	VTipo_Numero
Descrição	Retorna todos os dados povoados do tipo numérico.

Código Fonte:

```
CREATE VIEW dbo.VTipo_Numero
AS
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
   Dado_Local.id_local, Dado_Local.ref_temp,
   Dado_Local.Num_Registro, Tipo_Numero.Valor
FROM Dado_Local INNER JOIN
   Tipo_Numero ON
   Dado_Local.Seq_dado = Tipo_Numero.Seq_Dado
GO
```

Nome da Visão	VTipo_Objeto
Descrição	Retorna todos os dados povoados do tipo Objeto.

Código Fonte:

```
CREATE VIEW dbo.VTipo_Objeto
AS
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
   Dado_Local.id_local, Dado_Local.ref_temp,
   Dado_Local.Num_Registro, Tipo_Objeto.Valor
FROM Dado_Local INNER JOIN
   Tipo_Objeto ON
   Dado_Local.Seq_dado = Tipo_Objeto.Seq_Dado
GO
```

Nome da Visão	VTipo_Tabela
Descrição	Retorna todos os dados povoados do tipo Tabela.

Código Fonte:

```
CREATE VIEW dbo.VTipo_Tabela
AS
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
   Dado_Local.id_local, Dado_Local.ref_temp,
   Dado_Local.Num_Registro,
   Item_Tabela.Vl_Item AS Valor
FROM Dado_Local INNER JOIN
   Tipo_Tabela ON
   Dado_Local.Seq_dado = Tipo_Tabela.Seq_Dado LEFT OUTER JOIN
   Item_Tabela ON
   Tipo_Tabela.Id_Tabela = Item_Tabela.Id_Tabela AND
   Tipo_Tabela.Cod_Item = Item_Tabela.Cod_Item
GO
```

Nome da Visão	VTipo_Texto
Descrição	Retorna todos os dados povoados do tipo texto.

Código Fonte:

```
CREATE VIEW dbo.VTipo_Texto
AS
SELECT Dado_Local.id_Atributo, Dado_Local.Seq_Atributo,
   Dado_Local.id_local, Dado_Local.ref_temp,
   Dado_Local.Num_Registro, Tipo_Texto.Valor
FROM Dado_Local INNER JOIN
```

Tipo_Texto ON Dado_Local.Seq_dado = Tipo_Texto.Seq_Dado
GO

Nome da Visão	VUnidade
Descrição	Faz uma seleção sobre a relação ITEM_TABELA a fim de obter o código e descrição das unidades de medida usada pelos os dados do tipo numérico.

Código Fonte:

```
CREATE VIEW VUnidade (Cod_Item, Vl_Item)  AS
    SELECT Item_Tabela.Cod_Item, Item_Tabela.Vl_Item
    FROM Item_Tabela
    WHERE Id_Tabela = 79
GO
```