JOSÉ RAFAEL CARVALHO

MODELO DE METADADOS PARA SISTEMAS DE DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS UTILIZANDO XML

MARINGÁ

2007

JOSÉ RAFAEL CARVALHO

MODELO DE METADADOS PARA SISTEMAS DE DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS UTILIZANDO XML

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profª. Drª. Maria Madalena Dias

MARINGÁ

2007

Carvalho, José Rafael.
C3311m

Modelo de metadados para sistemas de descoberta de conhecimento em banco de dados utilizando XML./ José Rafael Carvalho. -- Maringá : [s.n] , 2007.
120f. : il.

Orientadora: Profª. Dra Maria Madalena Dias
Dissertação de mestrado - Universidade Estadual de Maringá – Programa de Pós – graduação em Ciência da Computação, 2007.

1.Banco de dados 2. Data warehouse 3. Metadados 4. XML I. Dias, Maria Madalena II. Universidade Estadual de Maringá – Programa de Pós graduação em Ciência da Computação

CDD 005.74

JOSÉ RAFAEL CARVALHO

MODELO DE METADADOS PARA SISTEMAS DE DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS UTILIZANDO XML

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Aprovado em 06/09/2007.

BANCA EXAMINADORA

Profª. Drª. Maria Madalena Dias
Universidade Estadual de Maringá – DIN/UEM

Profº. Dr. Donizete Carlos Bruzarosco
Universidade Estadual de Maringá – DIN/UEM

Profª. Drª. Marilde Terezinha Prado Santos
Universidade Federal de São Carlos – DC/UFSCar

AGRADECIMENTOS

Finalizando este curso de mestrado, agradeço a todas as pessoas que acreditaram em mim, todos que oraram, a fim de vencer as dificuldades. Em especial à minha família.

À orientadora do trabalho, pela própria arte de guiar o trabalho e paciência nos momentos difíceis.

Aos amigos de Maringá, pelo incentivo nos momentos mais críticos. A amiga Adriana, pelos diversos conselhos e ensinamentos técnicos, ao amigo Jô Sato, pessoa de valor inestimável. À Inês, secretária do mestrado, pela eficiência e colaboração.

Apesar de não estarmos mais juntos, agradeço à minha ex-namorada Clarissa, pela compreensão de tantas horas direcionadas unicamente para este curso.

Aos amigos do trabalho que contribuíram com suas idéias, entre eles, Jean.

RESUMO

Com o advento da tecnologia de *Data Warehousing*, adotada pelas empresas com o intuito de disponibilizar gerenciamento e integração de bancos de dados, não se deve esquecer da importância dos metadados, uma vez que eles são os responsáveis por manter informações sobre os dados que estão integrados um *data warehouse* (DW). A construção de um DW é uma das atividades do processo KDD (*Knowledge Discovery in Database* – Descoberta de Conhecimento em Banco de Dados). Neste processo, o DW é utilizado para armazenar os dados extraídos de bases de dados provenientes de sistemas transacionais. Assim, é fundamental a presença de metadados para registrar as informações sobre as extrações e transformações realizadas nos dados contidos no DW, bem como a forma de acesso para busca de conhecimentos que darão suporte à tomada de decisão. O fato desses dados serem, geralmente, provenientes de diversas fontes de dados, torna interessante que o formato dos metadados seja independente de plataforma. A interoperabilidade pode ser resolvida com o uso da tecnologia ***Extensible Markup Language*** (XML), possibilitando o acesso aos metadados por qualquer sistema computacional em diferentes plataformas. Assim, neste trabalho é apresentado um modelo de metadados, no formato XML, e a especificação de um gerenciador de metadados, implementado em Java, para dar suporte ao modelo proposto.

Palavras Chaves: Descoberta de Conhecimento em Banco de Dados, *Data Warehouse*, Metadados, XML.

ABSTRACT

With the appearance of Data Warehousing technology, adopted by enterprises with intention of turn available databases management and integration, should not forget about the importance of metadata, once they are responsible for maintain the information about the data that are integrate in a data warehouse (DW). The construction of a DW is one of the activities of KDD (Knowledge Discovery in Database) process. In this process, the DW is used to store the extracted data of databases coming from transactional systems. So, the presence of metadata to register information about the extractions and transformations done in the data contained in a DW is fundamental, as well as the access form to search of knowledge that will support the decision-making. The fact of the data be, usually, coming from several data sources, it turns interesting that the metadata format be independent of platform. The interoperability can be solved with the use of Extensible Markup Language (XML) technology, making possible the access to the metadata by any computation system in different platforms. So, in this text is presented a metadata model, in the XML format, and the specification of a metadata manager, implemented in Java, to give support to the proposed model.

Keywords: Knowledge Discovery in Database, Data Warehouse, Metadata, XML.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

CSS	*Cascading Style Sheets*
CWM	*Common Warehouse Metamodel*
DM	*Data Mart*
DW	*Data Warehouse*
DS	*Data Set*
DTD	*Document Type Description*
KDD	*Knowledge Discovery in Database*
MDC	*Meta Data Coalition*
MOF	*Meta Object Facility*
OIM	*Open Information Model*
OLAP	*On-Line Analytic Processing*
OLTP	*On-Line Transaction Processing*
OMG	*Object Management Group*
SGBD	Sistema Gerenciador de Banco de Dados
SOA	*Service Oriented Architecture*
SQL	*Struct Query Language*
UEM	Universidade Estadual de Maringá
UML	*Unified Modeling Language*
XMI	XML *Metadata Interchange Format*
XML	*Extensible Markup Language*
XSL	*Extensible Style Language*
W3C	*World Wide Web Consortium*

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Com o passar do tempo, as empresas que haviam investido maciçamente na área de computação, perceberam que seus sistemas de informações convencionais não podiam lhe trazer informações em nível estratégico, uma vez que tais sistemas são voltados para resolver os problemas cotidianos. Surgiu assim, a necessidade da construção de um *Data Warehouse* (DW) para que, a partir de uma grande quantidade de dados históricos, seja possível obter informações antes não disponíveis, como por exemplo, descobrir uma fraude ou mesmo personalizar o atendimento aos clientes para melhor atendê-los.

Um DW deve oferecer suporte à integração de dados independente do formato, ou seja, englobar todos os dados da empresa, logo, seus sistemas legados. O objetivo desta integração é facilitar a descoberta de conhecimento e a utilização desta importante ferramenta na administração de empresas e instituições (DIAS, 2001).

Os sistemas de informação tornaram-se mais exigentes no sentido de fornecer informações que não sejam usadas em nível operacional apenas, mas também em nível estratégico. Neste nível mais elevado, onde decisões sobre o futuro da organização estão em jogo, previsões erradas quanto ao mercado podem colocar em risco a sobrevivência da organização. A fim de auxiliar na tomada de decisão, surge a necessidade de Sistemas KDD (*Knowledge Discovery in Database*) que, a partir de grande quantidade de dados históricos, torna possível extrair informações antes não conhecidas. Como exemplo deste tipo de informação, pode-se citar venda combinada de produtos que pode levar a organização a mudar os produtos na prateleira, deixando próximos os que costumam ser vendidos juntos.

Em sistemas KDD, os conhecimentos para tomada de decisão são obtidos com a aplicação de técnicas de mineração de dados, que pode ser realizada sobre os dados contidos em bancos de dados de sistemas transacionais, após serem integrados e transformados. Porém, o ideal é que seja construído um DW, onde os dados já estarão preparados para servirem de entrada na aplicação dessas técnicas, podendo ser utilizados também por ferramentas OLAP (*On-Line Analitycal Processing*) (DIAS, 2001).

Diante da importância do DW para um sistema KDD, é necessário conhecer as informações sobre os dados que compõem esse “armazém de dados”, como eles são

integrados e transformados para serem gravados no DW, além das informações necessárias para o acesso aos dados do DW. Essas informações deverão ser registradas em um metadados. Portanto, é necessário definir um modelo de metadados para um sistema KDD e desenvolver um componente de *software* para gerenciamento do metadados definido. O modelo proposto contém o mínimo possível de informações sobre os dados, porém, ao mesmo tempo relata as informações mínimas necessárias para o bom funcionamento de um sistema KDD. Os metadados oferecem suporte à construção e acesso às informações do DW, além de outras informações necessárias no processo de descoberta de conhecimento em banco de dados (processo KDD).

O uso do formato XML na definição do modelo de metadados proposto possibilita maior independência de plataforma computacional. Desta forma, a aplicação ganha uma característica fundamental para qualquer *software*, ou seja, a desejada independência de qualquer empresa fabricante do produto.

Para efeito de validação do modelo definido, foi desenvolvido um componente de *software* para gerenciar o registro e acesso de informações no metadados, conforme o modelo de metadados proposto. A implementação deste componente de *software* foi realizada na linguagem Java, por ser livre e fornecer portabilidade.

1.2 OBJETIVOS

O principal objetivo deste trabalho é a definição de um modelo de metadados para sistemas KDD.

Os objetivos específicos deste trabalho são:

- Definir um modelo de metadados em XML que atenda as necessidades de sistemas KDD;
- Desenvolver um componente de *software* para gerenciamento do metadados proposto.

1.3 JUSTIFICATIVA

O processo KDD envolve etapas como (FELDENS et al., 1998): pré-processamento, mineração de dados e pós-processamento. Durante a primeira etapa um DW pode ser construído para armazenamento dos dados provenientes de diversos sistemas transacionais, Na etapa de mineração de dados, esse DW é usado para busca de conhecimentos para tomada de decisão. Da mesma forma, ele pode ser construído para ser usado por ferramentas OLAP.

Diante da importância do DW, é necessário saber as informações sobre os dados que compõem esse "armazém de dados". Esses "dados sobre dados", conhecidos como metadados, representam os dados que informam, por exemplo, informações sobre como os dados foram integrados no DW, histórico das extrações realizadas e formatos e localizações dos dados de origem, da área de estágio e do DW. Enfim, é praticamente impossível pensar em DW sem considerar a questão dos metadados, pois seria difícil controlar uma grande quantidade de dados sem qualquer informação de onde vieram e como estão armazenados. Portanto, torna-se necessária a definição de um modelo de metadados que suporte a construção de um DW e o acesso às informações e conhecimentos nele contidos, bem como o desenvolvimento de um gerenciador do metadados proposto.

1.4 CONTEXTUALIZAÇÃO

A construção de um DW envolve a identificação das bases de dados de origem, a seleção de tabelas e atributos e a definição de transformações necessárias sobre os dados.

Para facilitar o acesso às informações de um DW, é necessário que haja um meio de identificar onde e como essas informações estão armazenadas no DW.

Arquiteturas de DW propostas na literatura, tais como Menolli (2004) e Singh (2001), e arquiteturas para sistemas KDD, tais como apresentadas em Gupta et al. (2004), mostram a necessidade do uso de uma estrutura de metadados. Além disso, outros trabalhos como Dias (2001) e Castoldi (1999) enfatizam a importância de metadados na busca de conhecimento em banco de dados. Valentin (2006) propôs uma arquitetura para sistemas KDD que contém um componente de software para gerenciamento de metadados.

As ferramentas para construção de DW, OLAP e de mineração de dados, tais como Oracle Warehouse Builder (OWB, 2006), Oracle9i OLAP (ORACLE9i, 2006), MineSet

(MINESET, 2006), na certa usam uma estrutura de metadados, mas não é de conhecimento público como essa estrutura foi definida e é utilizada.

Portanto, é necessário definir uma estrutura de metadados padrão que possa ser usada em diversas plataformas computacionais e por diferentes ferramentas de DW, OLAP e mineração de dados, no contexto de sistemas KDD. Além disso, um componente de software responsável pelo gerenciamento desta estrutura de metadados facilitará o desenvolvimento de sistemas KDD e de outras aplicações relacionadas a esta área.

1.5 METODOLOGIA DE DESENVOLVIMENTO DA PESQUISA

A metodologia empregada neste trabalho é fundamentada no estudo de várias tecnologias envolvidas: sistemas de informação, KDD, DW, metadados, XML e Java. Diante do estudo destas tecnologias, foi possível alcançar uma base sólida que propiciou a definição de um modelo de metadados para um ambiente KDD.

Esta pesquisa pode ser classificada como pesquisa aplicada e experimental. Segundo Silva e Menezes (2000, pg. 20) uma pesquisa é tida como aplicada porque "objetiva gerar conhecimentos para a aplicação à solução de problemas específicos". Ainda Silva e Menezes (2000, pg. 21) definem o porquê de esta pesquisa ser considerada uma pesquisa experimental: "determina-se um objeto de estudo, selecionam-se as variáveis capazes de influenciá-lo, definem-se as formas de controle e de observação dos efeitos que a variável produz no objeto".

O objeto de estudo nesta pesquisa é metadados, abordando o seu uso no controle da construção e do acesso a um DW. As variáveis capazes de influenciar no objeto de estudo são as demais tecnologias envolvidas.

1.5.1 Processo de Desenvolvimento da Pesquisa

As etapas principais de pesquisa consideradas neste trabalho são: escolha do tema, revisão da literatura, estudo da procedência dos dados, definição do modelo de metadados para um ambiente de KDD e desenvolvimento do gerenciador de metadados.

A Figura 1.1 ilustra as etapas do processo de desenvolvimento da pesquisa, sendo que à esquerda são referenciadas as etapas principais da pesquisa, enquanto que à direita estão relacionados os elementos necessários para a concepção de cada etapa.

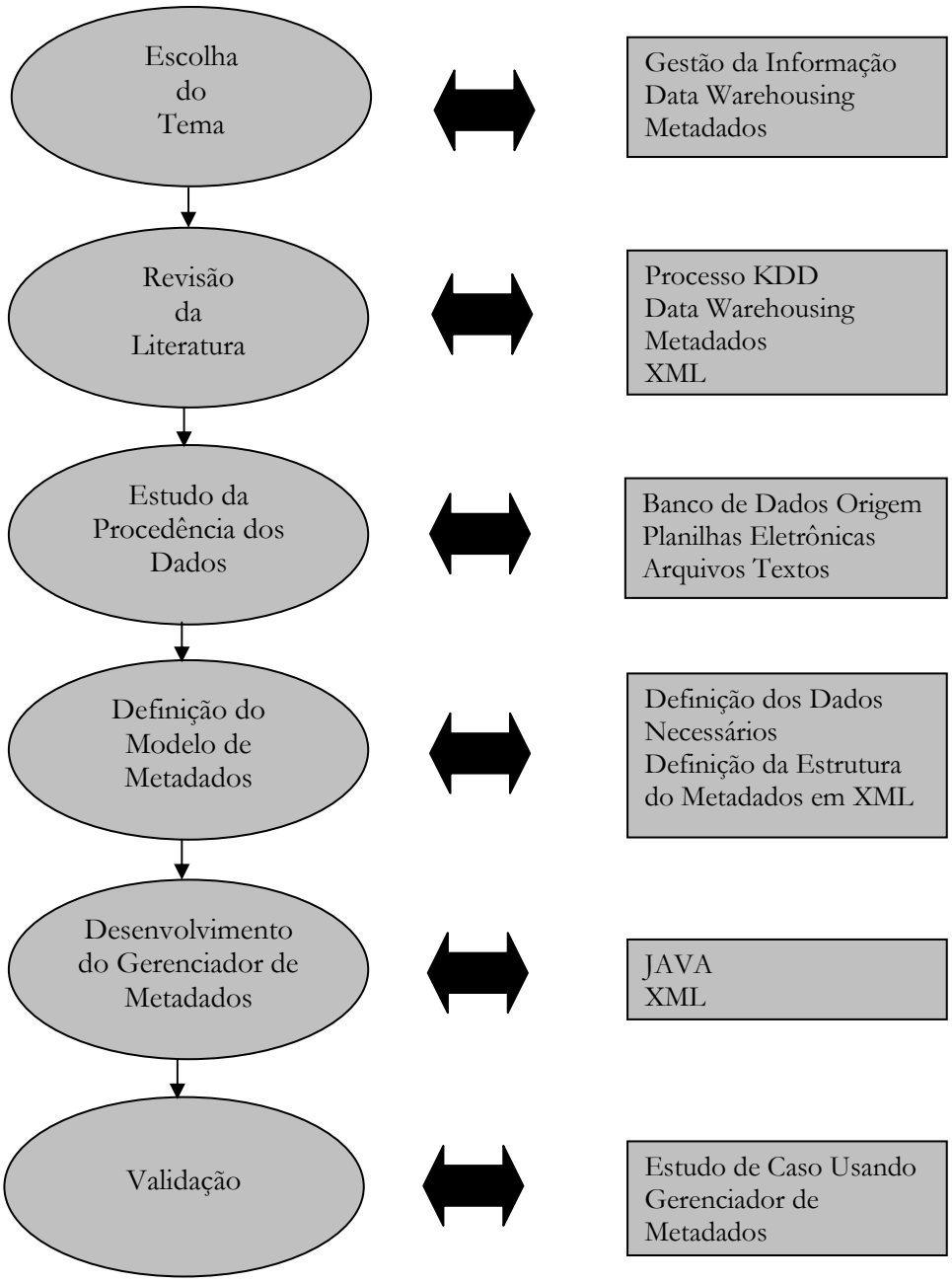

Figura 1.1: Processo de Desenvolvimento da Pesquisa

1) Escolha do Tema:

A escolha do tema é decorrente de um problema comumente encontrado nos projetos de DW, esse problema é a falta de um modelo de metadados que propicie o registro de informações mínimas necessárias em ambientes desse tipo.

Nessa etapa, foram definidas as tecnologias específicas a serem aplicadas na concepção de um protótipo que gerencie o componente metadados de um *Data Warehouse*.

2) Revisão da Literatura:

Essa etapa envolve a revisão de conceitos e características das seguintes tecnologias utilizadas na pesquisa: processo KDD, *Data Warehousing,* metadados e XML.

Toda a revisão de conceitos e características das tecnologias mencionadas forneceu uma noção geral do ambiente envolvido, servindo assim como um alicerce para a definição de um modelo de metadados que possa atender às necessidades desse ambiente.

3) Estudo da Procedência dos Dados:

Na construção de um DW é fundamental identificar os dados necessários a sua composição. Para isso, é necessário realizar um estudo para saber a origem desses dados, se eles são provenientes de um banco de dados, de uma planilha eletrônica ou mesmo de um arquivo texto.

4) Definição do Modelo de Metadados:

Tendo como base o modelo de metadados genérico definido por Tannenbaum (2002), os padrões de metadados OIM (*Open Information Model*) (PEREIRA, 2000) e CWM (*Common Warehouse Metamodel*) (MARINO, 2001), a estrutura de metadados definida por Castoldi (1999) e a arquitetura de DW definida por Menolli (2004), foi definido um modelo de metadados em XML.

5) Desenvolvimento do Gerenciador de Metadados:

Para a realização da validação do modelo de metadados proposto e tornar possível o uso deste modelo em um sistema KDD, foi desenvolvido um componente de software, implementado em Java.

6) Validação:

Para validação do gerenciador de metadados desenvolvido, foi realizado um estudo de caso onde é mostrado como o XML é criado e acessado.

1.6 ORGANIZAÇÃO DO TRABALHO

Além deste capítulo, este trabalho está dividido em 5 capítulos.

No segundo capítulo são apresentados conceitos e características de descoberta de conhecimento em banco de dados, *data warehouse*, metadados, XML e alguns trabalhos relacionados.

No terceiro capítulo é descrito detalhadamente o modelo de metadados proposto.

No quarto capítulo é apresentada a especificação do componente de software desenvolvido para gerenciamento dos metadados proposto.

No quinto capítulo é demonstrado o estudo de caso realizado utilizando o protótipo desenvolvido em Java para gerenciamento de metadados.

No sexto capítulo são tecidos comentários sobre as conclusões deste trabalho e relacionadas sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos necessários sobre as áreas de pesquisa envolvidas, que são: processo de descoberta de conhecimento em banco de dados, DW, metadados e XML. Também são descritos sucintamente alguns trabalhos relacionados que contribuíram para a pesquisa realizada.

2.1 PROCESSO DE DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS

Segundo Manilla (1996 apud DIAS, 2001) o processo de descoberta de conhecimento é um método semi-automático complexo e iterativo, sendo que, nesse processo são envolvidas várias etapas e ferramentas de suporte para que essas possam ser executas com sucesso. Para efeito de definição das etapas, diversos autores propõem metodologias para uma melhor divisão das mesmas, entre elas:

1) Feldens et al. (1998) propõem uma metodologia de três etapas bem definidas: pré-processamento, mineração de dados e pós-processamento. Sendo que as tecnologias de mineração de dados e DW apresentam papéis importantíssimos neste processo, além das questões de visualizações dos dados. A seguir são detalhadas essas etapas:
   - Pré-processamento: análise feita na organização com propósito de se concentrar no projeto de mineração de dados, análise dos dados existentes, integração das fontes de dados, transformações necessárias nesses dados, entre outros. Por exemplo, para integrar essas fontes de dados, o sistema KDD precisa saber o endereço de cada fonte de dados, que deve estar registrado nos metadados.
   - Mineração de dados: após a realização das tarefas pertinentes à etapa de pré-processamento, os dados são gravados em DW (dados integrados de toda a organização), *Data Mart* (DM) (dados por departamento) ou *Data Set* (DS) (conjunto de dados). Nesse ponto percebe-se que o DW é apenas um elemento de um processo mais amplo. Dessa forma, os dados contidos em um DW, DM ou DS, são usados como entrada na aplicação de algoritmos de mineração de dados. Nessa etapa, os

metadados devem informar como os dados do DW podem ser obtidos pelos algoritmos de mineração de dados.

- Pós-processamento: após a aplicação de algoritmos de mineração de dados, a informação extraída pode sofrer filtragem, estruturação ou mesmo classificação. O conhecimento descoberto pode ser filtrado por alguma medida estatística, ou mesmo ser estruturado de forma hierárquica. Nos metadados podem ser registrados os mecanismos utilizados na análise dos resultados, durante a etapa de pós-processamento.

2) Groth (1998) apresenta uma metodologia dividida em cinco etapas: preparação de dados, definição de um estudo, construção de um modelo, entendimento do modelo e predição.

3) Complementando a definição de Groth (1998), Lans (1997, apud DIAS, 2001) menciona a existência de um passo que antecede a preparação de dados, sendo denominado definição de objetivos.

4) Dias (2001) também elaborou um processo KDD que reúne as principais etapas definidas pelos autores acima citados. Este processo, apresentado na Figura 2.1, é dividido em seis passos, descritos a seguir:
   - Definição de Objetivos: nesta etapa são definidos os objetivos de negócio a serem alcançados com a aplicação da mineração de dados, assim como, o que deverá ser feito com os seus resultados, um exemplo é a mudança do plano de *marketing* da organização.
   - Preparação de Dados: esta etapa é responsável pelas tarefas de seleção e transformação dos dados, que são armazenados em um DW, *Data Mart* ou *Data Set*. Para auxiliar a efetivação desta etapa, deve ser mantido um catálogo de metadados sobre as fontes de dados, assim como, sobre o que está no DW, *Data Mart* e *Data Set*. Há de ressaltar que a realização desta etapa exige pleno conhecimento dos dados operacionais e seus relacionamentos, disponibilidade de tempo do analista e/ou do usuário e de alguns cuidados na escolha de subconjuntos de atributos e de dados, principalmente em relação à qualidade dos dados de origem.

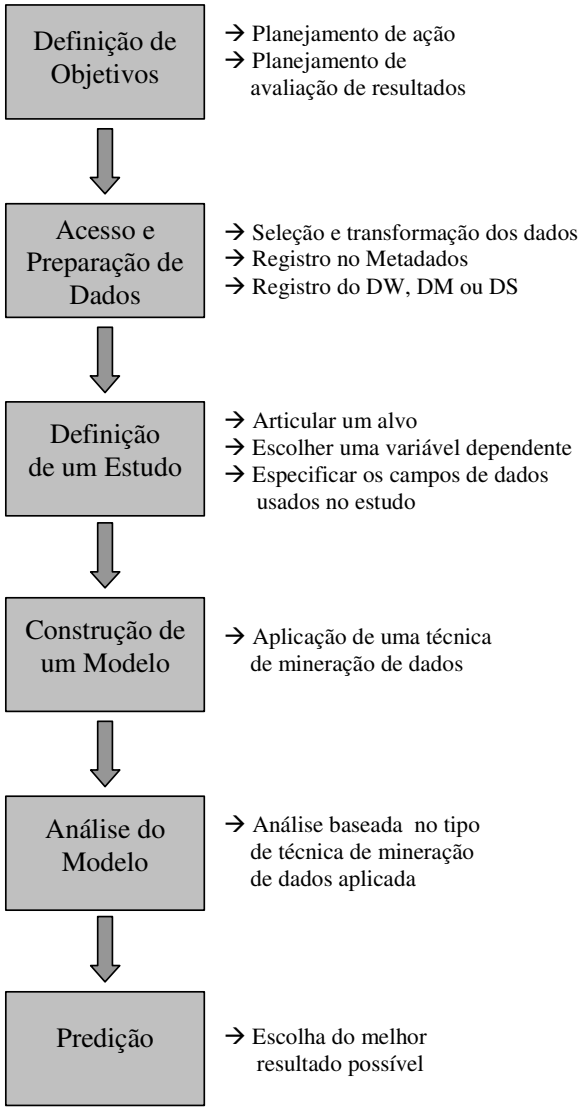

Figura 2.1: Processo de Descoberta de Conhecimento (DIAS, 2001, pg. 18)

- Definição de um estudo: etapa que se caracteriza pela definição de um estudo, podendo envolver a articulação de um alvo, escolha de uma variável dependente ou uma saída que caracterize um aspecto do alvo e a especificação dos campos de dados que serão utilizados no próprio estudo. As atividades executadas nesta etapa complementam os objetivos de negócio, definidos anteriormente, após a obtenção de um conhecimento mais detalhado sobre os dados operacionais existentes.
- Construção de um Modelo: com base nos dados transformados e no estudo definido, é construído um modelo aplicando uma técnica de mineração de dados.
- Análise do modelo: conforme o tipo de modelo usado para representar os dados, existem diversas formas de entendê-lo, dependendo dos indicadores agregados. Cada modelo pode gerar resultados em diferentes formatos, o que faz com que a sua

interpretação esteja ligada diretamente à técnica de mineração de dados usada pelo modelo.

- Predição: esta etapa é responsável pela escolha do melhor resultado possível que deve estar baseada na análise de dados históricos, na tarefa de suporte à decisão e nos objetivos traçados. Para essa escolha, o usuário necessita ter um bom entendimento sobre o negócio da organização e sobre o conhecimento descoberto.

2.2 *DATA WAREHOUSE*

Para Kimball et al. (1998), um DW apresenta como objetivo principal a integração de dados de diferentes fontes e formatos, sendo que não é construído com o intuito para suportar o processo funcional ou operacional da empresa, ou seja, não é o fim, mas o meio para facilitar o uso da informação.

Segundo Inmon (1997), o DW é caracterizado como sendo uma coleção de dados orientados por assunto, integrada, não-volátil, variante no tempo e, o mais importante, fornece suporte à tomada de decisão de âmbito administrativo.

Embora o conceito de *Data Warehousing* seja recente, ele se baseia em idéias que vinham sendo aplicadas em vários sistemas de informação há muitos anos (Inmon, 1997).

Porém, há de se ressaltar que há uma distinção clara entre os bancos de dados tradicionais e os DWs, uma vez que os banco de dados tradicionais são transacionais (relacional, orientado a objetos, de rede, ou hierárquico), enquanto que os DW's têm a característica distinta de que são direcionados principalmente para aplicações de apoio às decisões. Assim, os DWs são otimizados para a recuperação de dados, e não para o processamento rotineiro de transações.

Complementando, os DWs proporcionam acesso aos dados para análise complexa, descoberta de conhecimento e tomada de decisão, sendo que eles fornecem suporte às demandas de alto desempenho de dados e informações de uma organização (ELMASRI e NAVATHE, 2005). Na Tabela 2.1 são listadas algumas diferenças entre os dados transacionais e do DW.

Tabela 2.1: Diferenças entre BD transacionais e DW, adaptado de Singh (2001).

Características	BD's Operacionais	DW
Objetivo	Operações diárias do negócio	Analisar o negócio
Uso	Operacional	Informativo
Tipo de processamento	OLTP	OLAP
Unidade de trabalho	Inclusão, alteração, exclusão	Carga e consulta
Número de usuários	Milhares	Centenas
Tipo de usuário	Operadores	Comunidade gerencial
Interação do usuário	Somente pré-definida	Pré-definida e *ad-hoc*
Condições dos dados	Dados operacionais	Dados analíticos
Volume	Megabytes – gigabytes	Gigabytes – terabytes
Histórico	60 a 90 dias	5 a 10 anos
Granularidade	Detalhados	Detalhados e resumidos
Redundância	Evita-se	Ocorre
Estrutura	Estática	Variável
Manutenção desejada	Mínima	Constante
Acesso a registros	Dezenas	Milhares
Atualização	Contínua (tempo real)	Periódica (em *batch*)
Integridade	Transação	A cada atualização
Número de índices	Poucos/simples	Muitos/complexos
Intenção dos índices	Localizar um registro	Aperfeiçoar consultas

2.2.1 Características de um *Data Warehouse*

As principais características de DW, entre elas, orientação por assunto, integração, não-volátil, variação no tempo e armazenamento, são descritas a seguir (INMON, 1997) (INMON, 2000) (MACHADO, 2000) (CIFERRI, 2002) (MENOLLI, 2004):

1) Orientação por Assunto:

Os dados de DW são focados nos assuntos relevantes ao processo de tomada de decisão de uma organização, enquanto que os dados transacionais focam no que acontece no

seu dia-a-dia. Alguns exemplos de assuntos utilizados em projetos de DW são: produtos, atividades, contas, clientes, entre outros.

2) Integração:

A integração do DW diz respeito à consistência de denominações, das unidades dos valores contidos nas tabelas e outras padronizações aplicadas aos dados no sentido de que esses sejam transformados até estarem em um estado uniforme. Isso é devido ao fato que um DW em geral é composto por diversas fontes de dados, para que essas fontes possam interagir é necessária uma representação única para todos esses tipos de dados.

3) Variação no tempo:

Os dados contidos em um DW são temporais, por estarem relacionados a períodos de tempo bem definidos, o que auxilia na análise e na confirmação de acontecimentos sazonais dentro de uma determinada atividade ou ramo de negócio.

A variação no tempo significa que todos os dados no DW são precisos em algum instante no tempo, enquanto que, no ambiente operacional, os dados estão corretos no momento de acesso. Dessa forma, os dados do DW mostram uma imagem fiel da época em que foram gerados.

4) Não-volátil:

Como já citado anteriormente, os dados no DW não sofrem modificações, são sempre inseridos e nunca alterados. Em contrapartida, os dados de um banco de dados transacional, sofrem freqüentes alterações.

5) Armazenamento:

Realizadas as etapas de extração, limpeza, integração, há a necessidade de armazenar os dados no DW. Durante o armazenamento ainda são efetuados processos adicionais, como: verificação de restrições de integridade, ordenação dos dados, geração de agregações, construção de índices e também a condensação dos dados visando diminuir o volume de dados a ser armazenado no DW.

2.2.2 Modelagem Dimensional

Para Ballard et al. (1998) e Harrison (1998, apud MENOLLI, 2004), a modelagem dimensional é a técnica em que o modelo proporciona uma representação do banco de dados consistente com o modo que o usuário visualiza e navega pelo DW, combinando tabelas com dados históricos em séries temporais, cujo contexto é descrito através de tabelas de dimensões.

Segundo Song et al. (2001), o modelo dimensional apresenta duas vantagens principais na sua utilização em DW. A primeira vantagem, diz respeito ao provimento, por parte do modelo, de uma análise espacial multidimensional em ambientes de bases de dados relacionais, sendo analisados dados factuais utilizando dimensões. A segunda vantagem está no fato de que um típico modelo dimensional desnormalizado apresenta uma estrutura simples, acarretando assim, uma simplificação do processamento de consulta de usuários finais e a elevação do desempenho.

Essas vantagens permitem que as ferramentas OLAP analisem e gerenciem dados, proporcionando aos executivos um grande ganho de desempenho através do rápido acesso a uma grande variedade de visões de dados organizados através da base de dados multidimensional (MENOLLI, 2004).

Um modelo dimensional apresenta três elementos principais (MENOLLI, 2004):

- Fatos: coleção de dados, sendo que cada fato representa um item de negócio, uma transação ou evento de negócio. Machado (2000, pg 64) afirma que "fato é tudo aquilo que reflete a evolução dos negócios do dia-a-dia de uma organização".
- Dimensões: elementos que estão presentes em um fato, como exemplo, tempo, localização ou cliente. Podendo ser organizadas de maneira hierárquica, compostas por vários níveis, por exemplo, uma dimensão chamada "região" pode ser composta dos níveis "região", "estado" e "cidade".
- Medidas: atributos numéricos que representam um fato, uma vez que cada medida é composta pela combinação de dimensões que estão em um fato. Um exemplo para elucidar o elemento "Medidas" é a quantidade em valor monetário das vendas de um determinado vendedor em um mês.

Quanto à visualização, a modelagem dimensional permite que os dados sejam vistos na forma de cubo, sendo que cada dimensão do cubo representa o contexto de um determinado fato e a intersecção entre as dimensões representa as medidas (MENOLLI,

2004). Diferentemente da matemática, onde o cubo possui apenas três dimensões, na modelagem dimensional, o cubo pode apresentar quantas dimensões forem necessárias para representar um determinado fato, daí o motivo desse modelo também se chamar modelo multidimensional (MACHADO, 2000).

Entre os modelos, o mais comum é o modelo estrela, sendo que tal modelo possui uma entidade central denominada tabela de fatos e um conjunto de entidades menores, as tabelas de dimensões, relacionadas com a tabela de fatos. Segundo Barquini (1996), as tabelas de fatos podem utilizar até 95% da área destinada ao DW.

Além do modelo estrela, outro modelo dimensional que existe é o modelo floco de neve, sendo que este modelo pode ser conseguido pela decomposição de uma ou mais dimensões de maneira hierárquica (KIMBALL, 1996). É importante salientar que a diferença básica entre o modelo estrela e floco de neve é que no último as dimensões são separadas em hierarquias, tendo uma tabela para cada nível, enquanto que no primeiro as mesmas dimensões com todos os níveis se dispõem em uma única tabela.

Segundo Kimball (1996), no modelo estrela o tempo de acesso é mais eficiente do que no floco de neve, uma vez que, neste último, são necessárias mais junções entre as tabelas, portanto, um maior gasto de tempo. Por outro lado, o modelo floco de neve é esteticamente mais semântico (melhor estruturado) para se ver em relação ao modelo estrela, porém deve-se salientar que a principal preocupação de um DW é realizar consultas com o máximo de eficiência possível.

Na modelagem dimensional existem operações que permitem a navegação entre os dados, a seguir estão descritas as operações básicas (KIMBALL, 1996) (MACHADO, 2000):

- *Drill Down*: navegação de um nível genérico para um nível mais detalhado.
- *Drill Up*: navegação de um nível detalhado para um nível mais genérico.
- *Drill Across*: combinação de várias tabelas de fato que compartilham as mesmas dimensões em um único relatório.
- *Slice*/*Dice*: a operação *Slice* fatia o cubo para melhor visualização, por exemplo, onde livros e artigos são referenciados juntos, com o *Slice* seria possível visualizar separadamente, livros de artigos. Já a operação *Dice* se refere à mudança de perspectiva de visão, ou seja, em um dado momento o foco é um determinado assunto, com o uso do *Dice* é possível mudar essa visão para outro assunto diferente, alterando o foco. Um exemplo, em um determinado momento é visualizado em uma Universidade quantos artigos foram publicados em certo

período de tempo, com a operação *Dice* pode-se visualizar o número de artigos publicados nessa Universidade, ou seja, o foco deixa de ser a Universidade e passa a ser a publicação de artigos;

- *Pivot*: alternância na ordem de apresentação das dimensões do cubo de dados, ou seja, alternar linhas e colunas.

2.2.3 Granularidade de um *Data Warehouse*

De acordo com Inmon (1997), granularidade está diretamente relacionada com o grau de detalhe em que os dados serão armazenados no DW, sendo que a granularidade afeta diretamente o desempenho das consultas e o volume de dados. Além disso, nenhum outro aspecto de projeto terá importância se a granularidade não for conduzida de forma adequada e se a quantificação do volume de dados não for criteriosamente projetada e implementada.

É importante salientar que se a granularidade for muito baixa, o desempenho do sistema pode cair em níveis inaceitáveis, isso devido ao fato de possuir um número muito grande de registros armazenados. Em contrapartida, no caso da granularidade ser muito alta, os dados podem ser tão resumidos que praticamente não restará nada para o usuário fazer em termos de sumarizações e agregações, trabalhando mais em operações de *pivot* (pivô de dados), que é a alternância na ordem de apresentação das dimensões do cubo de dados.

2.2.4 *Data Mart*

Enquanto o DW permite uma visão global do negócio de uma organização, o *Data Mart*, também conhecido como DW departamental, é relacionado a um departamento ou a uma área específica do negócio, ou seja, mostra apenas uma parte do todo (departamento da organização).

A necessidade da utilização dos *Data Marts* é visível se for considerado que os sistemas de DW vão tornando-se bastante volumosos à medida que os dados históricos são transferidos para suas bases de dados. Nesse contexto, percebe-se a utilidade dos *Data Marts*, uma vez que eles são especializações dos DW*s* em sistemas menores, subdivididos em cada setor dentro da empresa.

As vantagens de um DW estruturado em *Data Marts* são (INMON, 1997):

- Personalização dos dados: as informações contidas em um *Data Mart* são referentes, em sua maioria, ao departamento onde estão definidos;
- Volume reduzido de informações: uma vez que não contém todos os dados da empresa, não ocupam muito espaço de armazenamento, ao contrário de um DW sem *Data Marts*;
- Histórico limitado: pode-se optar por armazenar um período menor do que o do sistema de DW, uma vez que é reduzido sensivelmente o tamanho do *Data Mart*, isso devido à organização dos dados alcançados;
- Dados sumarizados: a massa de dados deve ser reduzida aos totais mensais, ao invés de diminuir a granularidade e trabalhar com dados diários, por exemplo.

A Figura 2.2 mostra a relação de DW com vários *Data Marts*, sendo visível quatro departamentos de uma organização constituindo os *Data Marts*, que por sua vez são integrados em um único DW.

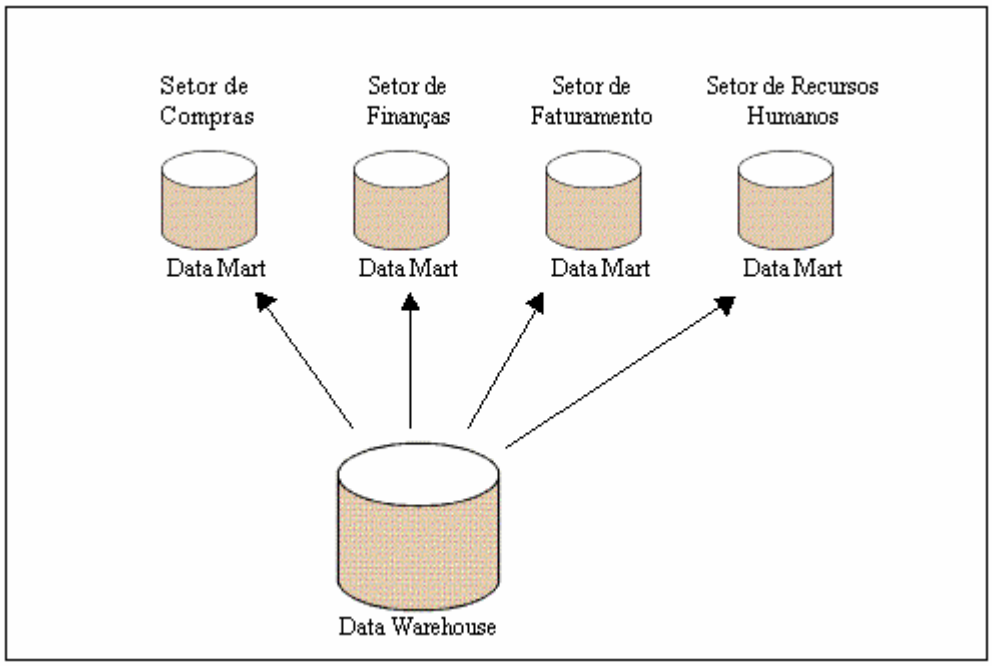

Figura 2.2: Modelo conceitual de *Data Marts* adaptado de Inmon (1997)

2.2.5 Arquitetura de *Data Warehouse*

De acordo com Singh (2001), arquitetura é um conjunto de regras que validam uma estrutura para um produto ou projeto de sistema. Para Menolli (2004), mais importante que as

ferramentas que serão usadas na concepção de um DW, são os próprios fundamentos da arquitetura.

Existem algumas propostas de arquitetura de ambiente de DW, entre elas: duas camadas, três camadas, multicamadas, *data marts* independentes, *data marts* incrementais, entre outras. Em destaque a arquitetura de multicamadas, devido ao modelo de metadados proposto neste trabalho estar relacionado à arquitetura multicamadas proposta por Menolli (2004), ilustrada na Figura 2.3. De acordo com Menolli (2004), uma arquitetura em camadas apresenta como vantagem principal o fato da independência das camadas, uma vez que, o que acontece na camada antecessora é transparente à camada superior, então, se houver a necessidade de alterar ou inserir novos componentes na camada antecessora não haverá problema, desde que se mantenha a maneira de como os dados são dispostos à camada superior.

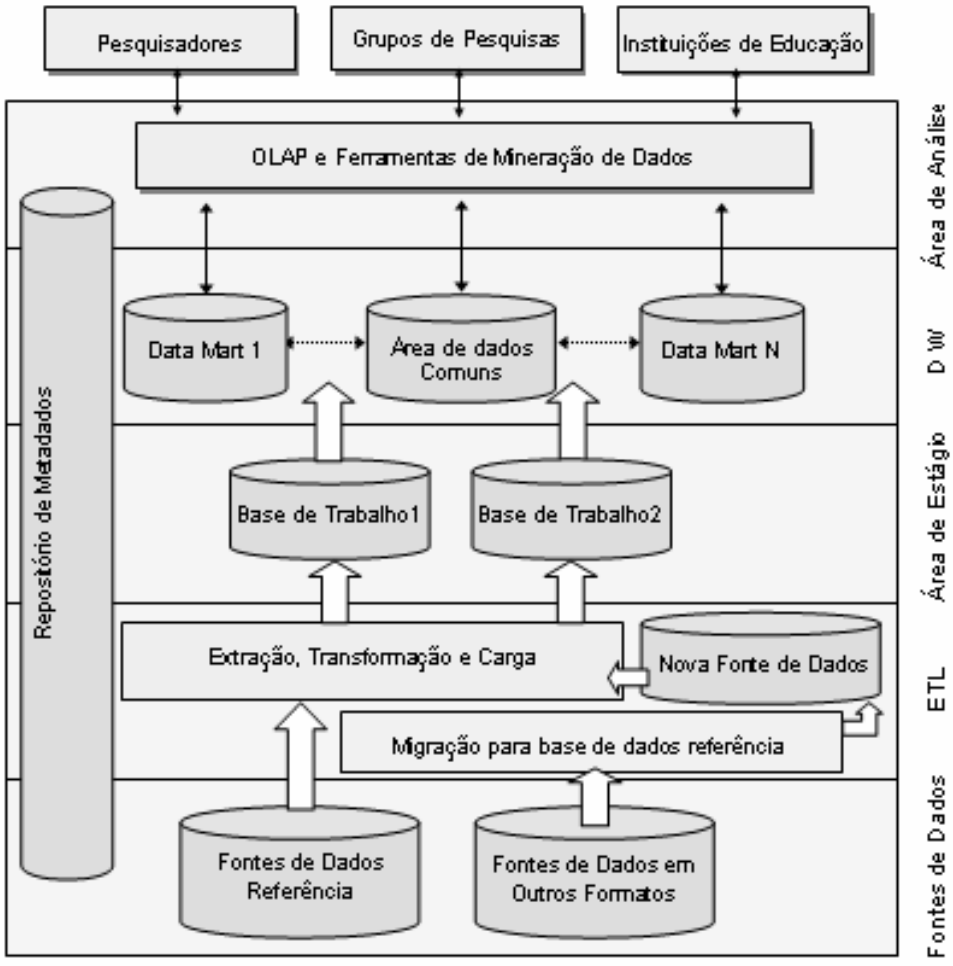

Figura 2.3: Arquitetura de DW proposta por Menolli (2004, pg. 61)

A arquitetura de DW proposta por Menolli (2004) tem como principal fundamento a substituição de fontes de dados de diversos formatos por novas bases de dados analíticas padronizadas, facilitando a integração dos dados. As flechas grossas simbolizam o carregamento de dados enquanto que as setas finas representam o acesso aos dados.

Menolli (2004) apresenta as funcionalidades básicas de cada camada:

- Nas camadas Fontes de Dados e ETL ocorre o processo de extração dos dados dos sistemas operacionais que é baseado na padronização dos dados, onde todos os dados devem estar em um mesmo formato para iniciar a limpeza, transformações e integração na área de estágio, facilitando a integração de dados de diferentes fontes de dados, que é um dos maiores problemas no desenvolvimento de um DW;
- A camada "Área de estágio" representa uma área de armazenamento intermediária dos dados que utiliza a modelagem normalizada em conjunto com a modelagem usada em modelos dimensionais, o que facilita a integração dos dados da área de estágio para o DW;
- Na camada DW é onde se encontra o DW, que é a área de armazenamento dos dados e segue o modelo proposto por Kimball et al. (1998), utilizando-se da implementação *BUS*, a qual é baseada em modelos dimensionais mantidos em diversos *data mart*s, contendo tabelas de dimensões e de fatos previamente estruturadas para manter a integração entre os dados dos diversos *data mart*s;
- Na área de análise figuram os elementos responsáveis pela busca de dados no DW, como ferramentas OLAP, mineração de dados e visões materializadas, sendo que são estes que disponibilizarão as informações para o usuário.

Na área de estágio existem passos que devem ser executados, desde as fontes de origem até à formatação dos dados, os quais são (MENOLLI, 2004):

1) Separação dos dados por granularidade: sendo necessária porque os sistemas de origem em muitos casos incluem dados em diferentes níveis de detalhes;
2) Limpeza dos valores de atributos: tratamento das seguintes situações: valores descritivos inconsistentes, decodificações ausentes, códigos inválidos, dados inválidos, dados ausentes, entre outros;
3) Transformação dos dados: transformações relativas às tarefas como: cálculo aritmético, conversões de tempo, tratamento de valores nulos, conversões de tipos de dados, unidades monetárias, entre outros;

4) Troca das chaves operacionais por chaves substitutas: chaves trocadas para facilitar a normalização dos dados;
5) Garantia da qualidade dos dados: verifica a consistência dos dados e de valores contidos nas tabelas.

Finalizando sobre a área de estágio, os passos relatados acima trazem como benefícios dados de qualidade e padronizados, facilitando a integração e a carga de dados no DW.

Na Figura 2.3 é destacável a importância dos metadados, pois eles se relacionam com todas as camadas integrantes da arquitetura de DW. Por exemplo, no processo de extração, transformação e carga dos dados, antes dessas ações ocorrerem, o próprio ambiente já utilizou o metadados, pois é ele que informará onde se encontram os dados, *drivers*, usuários e senhas necessários.

Há de se ressaltar que, um dos principais fundamentos da arquitetura de DW é a flexibilidade, uma vez que é indispensável que uma organização tenha a possibilidade de modificar e analisar seus dados de acordo com suas necessidades.

O modelo de metadados proposto é um componente integrante da arquitetura de DW proposta por Menolli (2004).

2.2.6 Ferramentas OLAP

Uma organização pode ser dividida em dois departamentos distintos quanto ao tipo de sistemas computacionais utilizados (VALENTIN, 2006) (CHAUDHURI e DAYAL, 1997):

- Departamento Operacional: sistemas computacionais referentes a controle, faturamento, produção, comunicação, registros, entre outros. Sendo que esses sistemas executam inúmeras operações e acessam dados de diversas fontes, dessa forma, são basicamente estruturados em transações, partindo daí a categoria OLTP (*On-Line Transacional Processing*). As operações envolvendo essas transações são otimizadas para obter um maior desempenho em sua execução, além de apresentar recursos de tratamento de concorrências e tolerância a falhas;

- Departamento Gerencial: departamento no qual a preocupação é com a saúde da organização, portanto, necessita de relatórios e informações para apoiar o processo de decisão. Dessa forma, este departamento necessita de sistemas computacionais

que analisam enormes quantidades de dados e geram gráficos, planilhas, ou seja, qualquer recurso que transforme os dados em informações. Esses sistemas são denominados como OLAP (*On-Line Analytical Processing*), uma vez que são otimizados para ambientes onde o tempo de resposta das pesquisas e o volume de dados são fatores determinantes.

A Figura 2.4 mostra a divisão de uma organização em duas visões departamentais (operacional e gerencial).

Complementando, ferramentas OLAP permitem análise e gerenciamento, possibilitando aos executivos um enorme ganho de desempenho através do rápido acesso a uma grande diversidade de visões de dados organizados através da base de dados multidimensional (VALENTIN, 2006). Há de ressaltar que essas ferramentas em geral necessitam que os dados estejam pré-definidos.

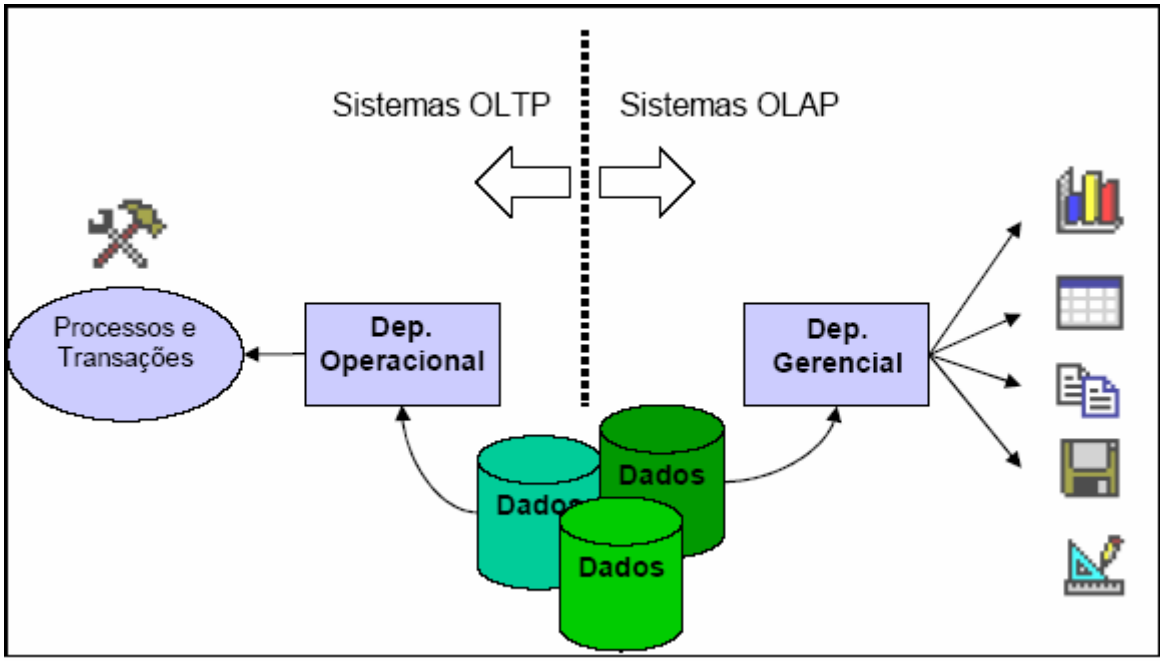

Figura 2.4: Divisão de uma organização em duas visões departamentais: operacional e gerencial (VALENTIN, 2006, pg. 17)

2.3 METADADOS

O próprio conceito sobre metadados se diverge entre os autores, mas em um ambiente de DW sua presença é primordial, para não dizer indispensável. Diante de uma grande quantidade de dados armazenada no DW, é extremamente importante saber informações sobre

a procedência desses dados, o modelo de origem desses dados, o histórico da extração desses dados, entre tantas informações ditas fundamentais para "conhecimento" com o que se está trabalhando. Nesta seção, são apresentadas características principais de metadados, categorias e classificação de metadados, necessários para a concepção de um modelo de metadados.

2.3.1 Conceitos de Metadados

Segundo Kimball et al. (1998), os metadados são dados sobre os próprios dados, sendo que neles se localizam informações diversas que são utilizadas e alimentadas por vários componentes da arquitetura. Há de considerar que essas informações contidas nesse repositório (metadados) são essenciais para o pleno funcionamento dos componentes e para a sua manutenção.

É importante salientar que, independente da arquitetura utilizada para a concepção de DW, é indispensável para o "bom" funcionamento da mesma um componente de metadados, uma vez que eles possuem as informações necessárias para o conhecimento do ambiente de DW por inteiro.

De acordo com Singh (1997), os metadados são o principal componente de um DW. Na literatura, a definição mais comumente percebida sobre metadados é que eles representam "dados sobre dados". O termo "meta" é proveniente da Grécia, sendo sua tradução "sobre", o que leva a expressão "dados sobre dados" (TANNENBAUM, 2002).

Também é encontrado na literatura o termo "documentação" para a definição de metadados, sendo que essa documentação é o dado de alto nível que descreve o dado de baixo nível, dessa forma, é o instrumento que transforma dados "crus" em conhecimento. Esse conhecimento pode, por exemplo, estar na forma de definição de um campo que informa se uma dada cadeia de *bits* é um endereço de clientes, parte de uma imagem fotográfica ou mesmo parte do código de um programa de computador (TANNENBAUM, 2002).

Tannenbaum (2002, pg. 88) vai além e compara o termo "informação" com "conhecimento", uma vez que o termo "informação" é definido pela autora como: "conhecimento, item de conhecimento, notícia". Para a autora, a própria tecnologia da informação (TI), desde seu surgimento, vem separando os termos "conhecimento" e "informação", sendo que os "zeros e uns" processados são claramente somente dados; Aplicações processando dados geram "informações", e os usuários de negócio converteram

essas “informações” em “conhecimento”. A Figura 2.5 mostra o fluxo da geração do “conhecimento”, sem, no entanto, mencionar o uso de metadados.

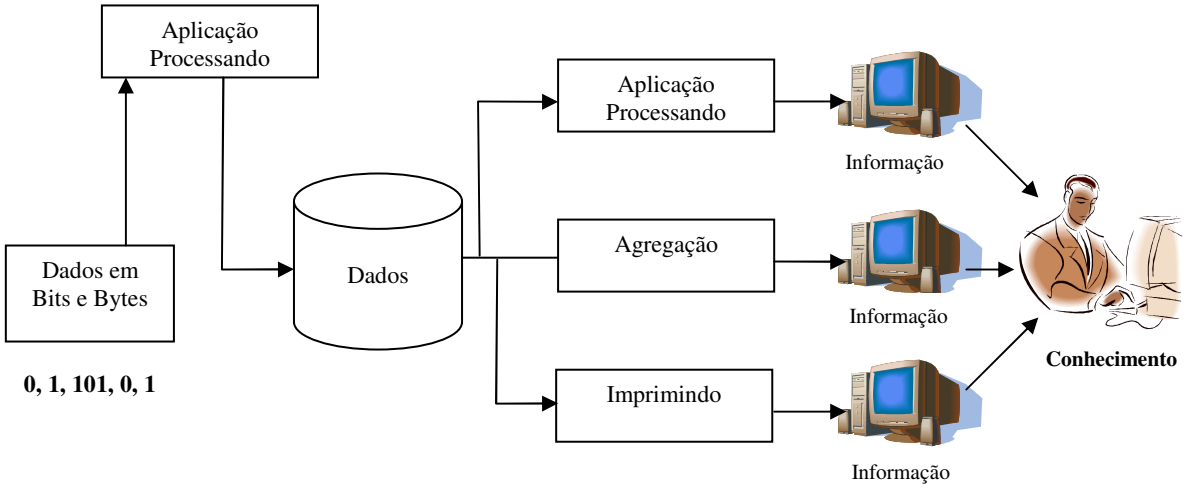

Figura 2.5: Dados para o conhecimento, traduzido de Tannenbaum (2002, pg. 89)

A Figura 2.6, já exibe como seria o fluxo da geração do “conhecimento” considerando o uso de metadados, isso significa que o usuário do negócio agora tem em suas mãos, além da própria “informação”, a “informação” da própria “informação”, agregando um maior conhecimento ao seu negócio em geral. Isso não era possível na sistemática definida na Figura 2.5. A Tabela 2.2 mostra, diante das diversas interpretações, como os metadados estão relacionados ao estágio do uso dos dados dentro de uma hierarquia evolucionária de gestão do conhecimento.

Detalhando um pouco mais sobre a Tabela 2.2, organizações em nível mais baixo da hierarquia gerenciam dados brutos, enquanto que organizações mais avançadas são capazes de administrar seus recursos de informação em níveis de Informação, Conhecimento ou Sabedoria.

Em nível de Informação, o foco está nos relacionamentos entre os componentes do sistema e os papéis individuais que eles assumem no sistema. Uma organização que está envolvida ao ponto em que ela pode mostrar explicitamente as regras de negócio que gerenciam seu comportamento, está em nível de Conhecimento. Uma organização atinge o nível mais avançado da hierarquia quando monitora ativamente seus sistemas para garantir que seu comportamento esteja tal como planejado de forma que pode detectar e diagnosticar qualquer comportamento anormal do seu sistema (DE LUCCA, 2003).

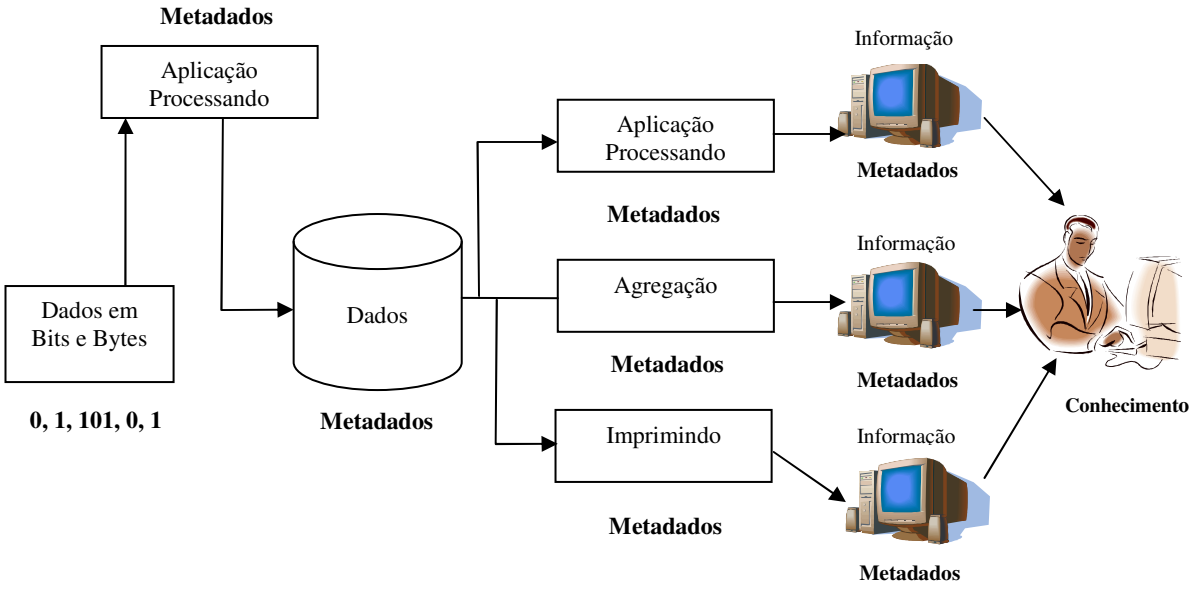

Figura 2.6: Adição de metadados na evolução do conhecimento(traduzido de Tannenbaum (2002, pg. 90))

Tabela 2.2: Níveis de Gerenciamento de Conhecimento (DE LUCCA, 2003).

Estágio	**Recurso a ser administrado**	**Definição de Metadados**
Dados	Valores dos dados	Informação necessária para administrar o recurso dos dados
Informação	Valores dos dados e o contexto da informação	Informação necessária para administrar o recurso da informação
Conhecimento	Valores dos dados, contexto da informação e regras de negócio	Informação necessária para administrar as regras e políticas de negócio da organização
Saber	Valores de dados, contexto da informação, regras de negócio, monitoramento das regras de negócio e métricas de avaliação	Informação necessária para administrar o comportamento da organização de acordo com suas regras e políticas de negócio

Contudo, pode-se dizer que a finalidade principal dos metadados é documentar e organizar dados de forma estruturada com o objetivo de minimizar a duplicação de esforços e facilitar a manutenção e uso dos dados.

2.3.2 Categorias de Metadados

Diante da grande variedade de conceitos que se tem em relação a metadados, a sua taxonomia não poderia ser diferente. A forma mais evidente, mas não deixando de ser uma visão simplista, é que os metadados podem ser divididos em duas categorias básicas: metadados técnicos e metadados de negócio (SANTOS, 2003):

- Metadados técnicos: descrição dos dados necessários para as diversas ferramentas que precisam armazenar, manipular ou movimentar dados. Essas ferramentas podem se tratar de um banco de dados relacional, ferramentas CASE (*Computer Aided Software Engineering*), ferramentas de pesquisa em banco de dados, ferramentas OLAP (*On-Line Analitical Processing*), entre outras;
- Metadados de negócio: descrição de dados necessários pelos usuários de negócios, para entender o contexto do negócio e o significado dos dados, sendo que atualmente existem ferramentas só para efeito de documentação.

Complementando, metadados técnicos fornecem aos desenvolvedores e usuários técnicos de sistemas de suporte à decisão, a confiança de que o dado está correto, sendo que esses metadados são críticos para a manutenção e o crescimento contínuo de um DW. Já os metadados de negócios representam a ligação entre o DW e os usuários de negócios, uma vez que esses dados fornecem uma espécie de mapa aos usuários para que eles possam acessar os dados, tanto no DW como nos *Data Marts.*

2.3.3 Classificação de Metadados

Vários trabalhos relativos aos metadados fazem uso de classificações com o intuito de identificar a natureza de seus elementos e o papel desempenhado por cada elemento dentro do contexto de informação que está sendo considerado. O objetivo de classificar metadados de forma geral é enfocar o modo que os mesmos são obtidos (automática ou manual); se são ou

não baseados no conteúdo do recurso; se são relacionados ou não ao tipo de mídia; se são dependentes ou não de um domínio qualquer de informação.

Na seqüência é descrita uma classificação de metadados proposta por Kashyap et al. (1995). Um aspecto que se destaca nesta proposta está relacionado aos metadados descritivos de conteúdo: a classificação subentende que esses metadados também podem ser classificados como dependentes de conteúdo no caso de serem extraídos por processos de análise automática sobre o conteúdo do recurso (KASHYAP et al., 1995) (BARRETO, 1999).

- **Metadados dependentes de conteúdo**: são aqueles termos extraídos por processos automáticos, inteiramente guiados pelo tipo da mídia, que tomam como base o conteúdo do dado original. Um extrator automático de metadados para arquivos de e-mail, por exemplo, pode filtrar informações tais como: quem enviou a mensagem, a data, o assunto, para quem foi enviada a mensagem, etc.;
- **Metadados descritivos de conteúdo**: são termos descritivos que são associados ao conteúdo do recurso. Podem ser específicos da mídia em questão como, por exemplo, os termos que descrevem a cor predominante, o formato e a textura de uma imagem, ou independentes da mídia tais como os termos que classificam o conteúdo (ou assunto) de um documento. Também podem ser classificados em:
    - **Dependentes de domínio**: são termos que empregam conceitos relacionados a algum domínio específico como, por exemplo, aqueles que descrevem a interpretação de uma imagem de satélite (vegetação, uso territorial, etc.);
    - **Independente de domínio**: são termos descritivos genéricos que não são baseados em conceitos específicos extraídos do domínio de informação no qual o recurso analisado esteja inserido. Podem ser enquadrados nesta categoria os metadados que descrevem a estrutura de um documento multimídia, por exemplo;
    - **Independentes de conteúdo**: são aqueles termos descritivos que não podem ser extraídos diretamente do conteúdo do recurso ao qual se refere, como exemplos podem ser citados a data/hora da última modificação e o local de armazenamento de um documento.

Além da classificação geral, é destacável a importância da classificação funcional devido ao fato do seu enfoque estar voltado aos metadados necessários para a descrição de um documento eletrônico composto por itens de multimídia. A proposta de classificação

funcional de metadados é utilizada no projeto HyperStorM (*Hypermidia Document Storage and Modeling*) de Bohm e Rakow (1994). O Projeto provê uma infra-estrutura genérica para o gerenciamento (com base em metadados) de diversos tipos de documentos estruturados, sendo composto por seis etapas:

- **Metadados para a representação de tipos de mídia**: provêem informações relacionadas à apresentação do dado multimídia, ao idioma, ao formato e às técnicas de compressão;
- **Metadados descritivos de conteúdo**: são os metadados usados para descrever o conteúdo de um documento multimídia. Podem ser dependentes da mídia;
- **Metadados para classificação de conteúdo**: correspondem a informações adicionais que podem ser derivadas do conteúdo do documento auxiliando em algum tipo de classificação que possa ser utilizada em sistemas de filtragem. No contexto médico, por exemplo, estas informações podem ser úteis para a determinação das pessoas autorizadas a lerem um documento específico;
- **Metadados para a composição do documento:** são metadados que descrevem os relacionamentos entre os componentes do documento e o papel de cada componente dentro de um determinado contexto. Possibilitam a implementação de métodos envolvendo sua composição, tal como *ObtemPróximoComponente*;
- **Metadados para histórico do documento**: provêem informações históricas e de acompanhamento relativas à natureza evolucionária e dinâmica de um documento, observando seus componentes individuais. São exemplos de metadados desta categoria: “aprovadoPor”, “nãoAprovado” e “dataDaÚltimaAtualização” ;
- **Metadados para a localização do documento**: são aqueles que possibilitam o acesso a um determinado documento.

2.3.4 Importância dos Metadados

A importância dos metadados é percebida no início do desenvolvimento de um projeto de DW, sendo aconselhável começar primeiro com os metadados (HURWITZ, 1996).

Partindo do princípio que as aplicações de âmbito operacional da empresa são desenvolvidas em tempos diferentes, não é raro o surgimento de dados inconsistentes ou redundantes. Além disso, de acordo com Come (1999), as organizações costumam apresentar um outro problema comum: elas possuem múltiplas fontes de dados, sendo que cada uma

dessas fontes tem seu próprio conjunto de regras pré-definidas, convenções para nomes, formatos de arquivos e etc. Neste caso, é praticamente impossível para o usuário, e mesmo para o administrador, saberem quais fontes de dados usarem em diferentes circunstâncias. Diante desta problemática, é que aparece a importância dos metadados, para assim, trazer informações necessárias tanto para um usuário comum quanto para o próprio administrador.

Segundo Come (1999), os metadados devem isolar o usuário da complexidade de acessar informações distribuídas, enquanto facilita a atualização e sincronização de vários bancos de dados. Porém, se não funcionar, os usuários voltarão a se encontrar com os problemas que o DW pretendia resolver, ou seja, diferentes respostas para a mesma questão e a resultante falta de confiança na informação obtida.

Contudo, é visível que sem uma administração efetiva de dados, um DW não atingirá o seu objetivo de integração de dados de uma determinada organização, assim, os metadados constituem o principal recurso para a administração de dados do DW e se definem como componente formal extremamente importante no processo de *Data Warehousing*. Sem os metadados, os dados passam a não ter significado, tornando o ato de localizar informações contidas em um DW uma tarefa difícil, fazendo um paradoxo, seria semelhante a procurar o telefone de uma pessoa sem a ajuda de uma lista telefônica (COME, 1999).

Dessa forma, usuários seriam como turistas deixados em uma nova cidade sem qualquer informação sobre a mesma, e os administradores do DW seriam como os administradores que não têm idéia do tamanho dessa cidade e em que velocidade ela está crescendo. Ou ainda, um DW sem metadados adequados é um armário-arquivo cheio de papéis, mas sem pastas ou etiquetas (COME, 1999).

Diante de sua importância, o interesse sobre o assunto de metadados vem crescendo. Segundo De Lucca (2003):

- Existe uma necessidade crescente de melhores formas de encontrar e avaliar informações na *Internet* e nas *Intranets*;
- Os sistemas de gerenciamento de conhecimento integram informações de diversas fontes e aplicações que precisam ser mais facilmente pesquisadas e mantidas.

Sabendo da importância dos metadados para um DW, os próprios devem estar "aptos" a responderem perguntas básicas em um ambiente de grande quantidade de dados, tais como (COME, 1999):

- Que tabelas, atributos e chaves o DW contém?

- Qual é a origem de cada conjunto de dados?
- Que transformação lógica foi usada na carga do dado?
- Como os metadados têm mudado ao longo do tempo?
- Quais *aliases* (nome alternativo que normalmente é mais fácil de memorizar) existem e como eles se relacionam?
- Com qual freqüência os dados são carregados?
- Qual o volume de dados existente?

Perguntas que são de extrema importância para um administrador saber responder, sendo que essas respostas os próprios metadados, desde que bem estruturados, trarão com os detalhes necessários.

2.3.5 A espécie de Informação dos Metadados

Uma ferramenta fundamental no gerenciamento de um DW é um repositório de metadados, principalmente no momento de converter dados em informações para o negócio. Um repositório de metadados, desde que bem elaborado, deve conter informações sobre a origem do dado, regras de transformação, nomes e *aliases*, formatos de dados e etc. Assim, este repositório de metadados deve trazer bem mais que a descrição sobre colunas e tabelas, deve, principalmente, conter informações que agregam valor ao próprio dado. Essas informações básicas são listadas a seguir (COME, 1999):

- Fontes de dados: independente da aplicação (sistema ou um processo), é indispensável que a origem dos dados seja identificada;
- Destino dos dados: de igual importância quanto à origem dos dados, é também ter conhecimento sobre o seu destino, principalmente quando esse dado é utilizado como fonte para outras operações;
- Formato dos dados: informações a respeito, por exemplo, do tamanho do campo e tipo de dado;
- Nomes e *aliases*: é sabido que cada dado deve ter um nome, nome este que pode ser tanto técnico quanto relativo a uma área de negócios, sendo possível a utilização de *aliases* por tornarem o DW muito mais amigável e entendível, principalmente para os usuários de negócios, além de que, são bastante úteis

quando diferentes departamentos desejam usar seus próprios nomes para identificar um mesmo dado;

- Definições de negócios: parte fundamental dos metadados, pois é necessário que o entendimento de cada elemento de dado seja suportado dentro do contexto do negócio, sendo inclusive, importante garantir a consistência dessas informações, a fim de que os usuários possam encontrar rapidamente uma definição para a informação que precisam;
- Regras de transformação: são as regras de negócio de uma forma codificada;
- Atualização dos dados: normalmente, o histórico das atualizações é mantido pelo próprio banco de dados, mas ter um elemento de metadado que possa, por exemplo, identificar a última atualização de um dado pode ser muito útil para usuários que desejam determinar o estado de atualidade desse dado ou verificar a consistência de uma dimensão tempo em um DW;
- Requisitos de teste: o repositório de metadados é o local correto para manter os critérios de julgamento de um dado ou validação de uma tabela por uma rotina de teste, devendo manter um padrão para esses procedimentos de testes;
- Indicadores de qualidade: necessidade de indicadores para indicar a qualidade de um elemento de dado, uma vez que, a fonte do dado, a quantidade de processamento aplicado a ele e muitos outros fatores podem afetar a qualidade do dado;
- Processos automáticos (*triggers* - disparador de eventos em programas de computador): é comum a presença de *triggers* que procuram manter a consistência do banco de dados durante as atualizações, devendo estar liberados para a consulta de usuários e desenvolvedores, a fim de evitar a criação de uma situação que possa "disparar" um processo fora do seu contexto normal de utilização;
- Gestão das informações: a gestão está associada com propriedade e responsabilidade sobre os dados, devendo ter definida acessibilidade à informação, ou seja, para quem é a responsabilidade pelos dados e pela entrada de metadados em um DW;
- Acesso e segurança: deve haver o cuidado com a segurança das informações mantidas nos metadados, assim, os metadados devem conter informações suficientes para identificação sobre quem pode ler, atualizar, excluir ou inserir informações no banco de dados e sobre quem controla esses direitos de acesso.

2.3.6 Fontes de Metadados

Quanto às fontes de metadados, de acordo com Come (1999), basicamente existem duas fontes bem distintas: a formal e a informal. Sendo que essas fontes compreendem os metadados técnicos e de negócio de uma organização.

Segundo Come (1999), os metadados de fontes formais são aqueles que já foram discutidos e documentados, sendo normalmente armazenados em ferramentas ou documentos que são mantidos, distribuídos e reconhecidos por toda a organização. Fontes formais de metadados, que por sua vez, alimentam tantos os metadados técnicos como os de negócio. Os metadados informais consistem do conhecimento corporativo, políticas e orientações que não estão em um formato padrão, sendo, portanto, a espécie de informação que as pessoas "apenas sabem" e que faz parte do "conhecimento da empresa". No entanto, não são formalmente documentadas, mas ao mesmo tempo, é tão importante quanto às fontes formais de metadados.

Os metadados informais fornecem algumas das informações mais importantes, já que tendem a estar relacionados aos negócios. Sendo importante observar que, normalmente, muitos dos metadados de negócio são informais. No entanto, para se atingir um resultado, é necessário que esses metadados sejam capturados, documentados, formalizados e refletidos no DW, transformando, dessa forma, metadados informais em metadados formais.

Diante do fato de que empresas não são iguais, é muito difícil especificar onde os metadados informais podem ser encontrados, mas de maneira geral as fontes poderiam ser (COME, 1999): gestão dos dados, regras de negócios, definições de negócios, transformações, sumarizações, entre outros.

2.4 PADRÕES DE METADADOS

Nos últimos anos, a questão da padronização vem sendo discutida pelos comitês e órgãos internacionais no sentido de garantir interoperabilidade entre ferramentas e repositórios. Porém, a troca de metadados entre ferramentas é uma tarefa crítica visto que as ferramentas codificam e armazenam seus metadados de forma proprietária, segundo esquemas conceituais também proprietários. Sendo assim, o problema da interoperabilidade ocorre em dois níveis: conceitual e codificação. Neste contexto, os padrões OIM (*Open Information*

Model) e CWM (*Common Warehouse Model*) têm se destacado como padrões para representação e troca de metadados. Ambos os padrões especificam metamodelos, que podem ser vistos como esquemas conceituais para representação de metadados. No quesito intercâmbio de metadados, o padrão OIM utiliza uma especificação proprietária em XML, enquanto que o padrão CWM utiliza o padrão XML *Metadata Interchange* (XMI). A seguir eles são descritos sucintamente.

2.4.1 OIM (*Open Information Model*)

O padrão de metadados OIM (*Open Information Model*) surgiu da parceria de múltiplas empresas, algumas inclusive líderes de mercado, com o objetivo de prover suporte à interoperabilidade entre ferramentas de desenvolvimento, por meio da utilização de um modelo de informação compartilhado. Este padrão foi concebido de forma a permitir o acompanhamento de todas as fases de desenvolvimento de um sistema de informação, desde a fase de análise até a fase de implantação.

De acordo com Marino (2001), a versão 1.0 do OIM foi adotada em julho de 1999 como padrão pelo *Meta Data Coalition* (MDC), uma coalizão que atualmente consiste de mais de 50 membros, incluindo Microsoft, Ardent software, Brio Tecnnologies, Evolutionary International (ETI), Informática, Platinum, SAS Institute e Viasoft, e que tem por finalidade a definição e a implementação de um formato de padrão de intercâmbio de metadados, bem como os mecanismos de suporte necessários nos contextos de análise e projeto de sistemas de informação e de ambientes de data warehousing. O padrão OIM é baseado nos padrões de indústria UML (*Unified Modeling Language*), XML (*eXtensible Markup Language*) e SQL (*Structured Query Language*), e busca prover o suporte a tecnologias de computação diversas como CASE, componentes, intranet, bancos de dados e *data warehousing*. Atualmente, OIM encontra-se na versão 1.1, ainda uma proposta. O padrão OIM é uma especialização dos conceitos abstratos de UML2 em submodelos que descrevem metadados de domínios específicos, tais como (MARINO, 2001):

- **Modelo de Análise e Projeto** cobre o domínio da modelagem orientada a objeto e projeto de sistemas de software;
- **Modelo de Objetos e Componentes** cobre os diferentes aspectos envolvidos no ciclo de vida de desenvolvimento de componentes;

- **Modelo de Engenharia de Negócios** provê os tipos de metadados necessários à captura dos objetivos e da infra-estrutura organizacional de um negócio bem como dos processos e regras que governam o negócio;
- **Modelo de Gerenciamento do Conhecimento** busca prover os mecanismos necessários para a captura, organização e uso dos recursos de informação de uma empresa de forma a adicionar valor ao negócio;
- **Modelo de Bancos de Dados e *Data Warehousing*** provêem tipos de metadados para o gerenciamento de esquemas no contexto de projeto de banco de dados, reuso de esquemas e *data warehousing*.

Segundo Pereira (2000), os tipos de metadados especificados pelo padrão OIM são concebidos em submodelos voltados para uma determinada área de aplicação. Sendo que cada submodelo pode ser estendido para incorporar características específicas de cada aplicação. Entre esses submodelos se destacam o DBM (*Database Information Model*) e TFM (*Transformation Information Model*). O modelo DBM descreve as informações sobre os esquemas das bases de dados, já o TFM é uma extensão do DBM. Na ação de mover os dados dos sistemas transacionais (OLTP) para o DW/DM, esses dados devem estar adequadamente transformados para facilitar as consultas, sendo que estas transformações, bem como os dados acessados por elas, são mantidos no modelo TFM. Os objetivos do modelo TFM são (MDC OIM, 1999 apud PEREIRA, 2000, pg. 45):

- Permitir que as ferramentas de transformação armazenem em um único local informações relativas às transformações executadas;
- Apresentar um mecanismo de armazenamento de metadados para que aplicações documentem suas transformações de maneira consistente;
- Possibilitar o uso do modelo de informação DBM, tais como a descrição de tabelas e colunas, para a construção ou extensão dos modelos de transformação.

O armazenamento das transformações permite a reutilização das informações sobre as mesmas, um formato padrão para seu armazenamento que possibilita o compartilhamento dessas transformações entre as ferramentas (PEREIRA, 2000).

2.4.2 CWM (Common Warehouse Metamodel)

De acordo com Marino (2001), CWM é um padrão de metadados cujo intuito é permitir a integração de sistemas de DW, *e-business* e sistemas de negócios inteligentes em ambientes heterogêneos e distribuídos, através de uma representação e de um formato de troca de metadados. O padrão CWM é parte dos esforços do grupo OMG (*Object Management Group)* no sentido de prover um *framework* arquitetural orientado a objeto e padronizado para aplicações distribuídas, de forma a suportar reusabilidade, portabilidade e interoperabilidade de componentes de software orientados a objetos em ambientes heterogêneos.

Proposto pelo grupo OMG em conjunto com fornecedores líderes de mercado como IBM, Oracle, Unisys, Hyperion Solutions (Essbase Software), o padrão CWM foi adotado como um padrão OMG em junho de 2000 e é baseado nos seguintes padrões OMG (MARINO, 2001): UML, MOF (*Meta Object Facility*), uma metalinguagem e um padrão de repositório de metadados, e XMI (*XML Metadata Interchange*), um padrão baseado em XML para troca de metadados entre ferramentas e repositórios orientados a objetos.

Assim como o padrão OIM, o padrão CWM é definido em UML 1.3 e organiza os tipos de metadados por assunto, conforme o metamodelo mostrado no Figura 2.7, (Marino, 2001), (CWM, 2001):

- **Object Model:** providencia a construção básica de classes em todos os pacotes do CWM. O *Object Model* utiliza da UML somente as funcionalidades necessárias para a criação e descrição do CWM;
- **CWM Foundation** provê os tipos de metadados para representação de conceitos e estruturas que são compartilhados por outros pacotes CWM:
    - **Business Information**: disponibiliza serviços de informação do negócio para outros pacotes;
    - **Data Types:** providencia suporte para a construção dos metamodelos, especificando os tipos dos dados que eles necessitam;
    - **Expressions:** disponibiliza as funções/expressões matemáticas como “sum(a,b)” e “a+b”, respectivamente, sendo a semântica de uma função particular/operação específica da ferramenta de implementação, portanto, não á capturado pelo CWM;
    - **Keys Indexes:** especificam instâncias por ordenações alternadas representadas no pacote “Foundation”, sendo que essas instâncias podem ser compartilhadas por vários modelos de dados;

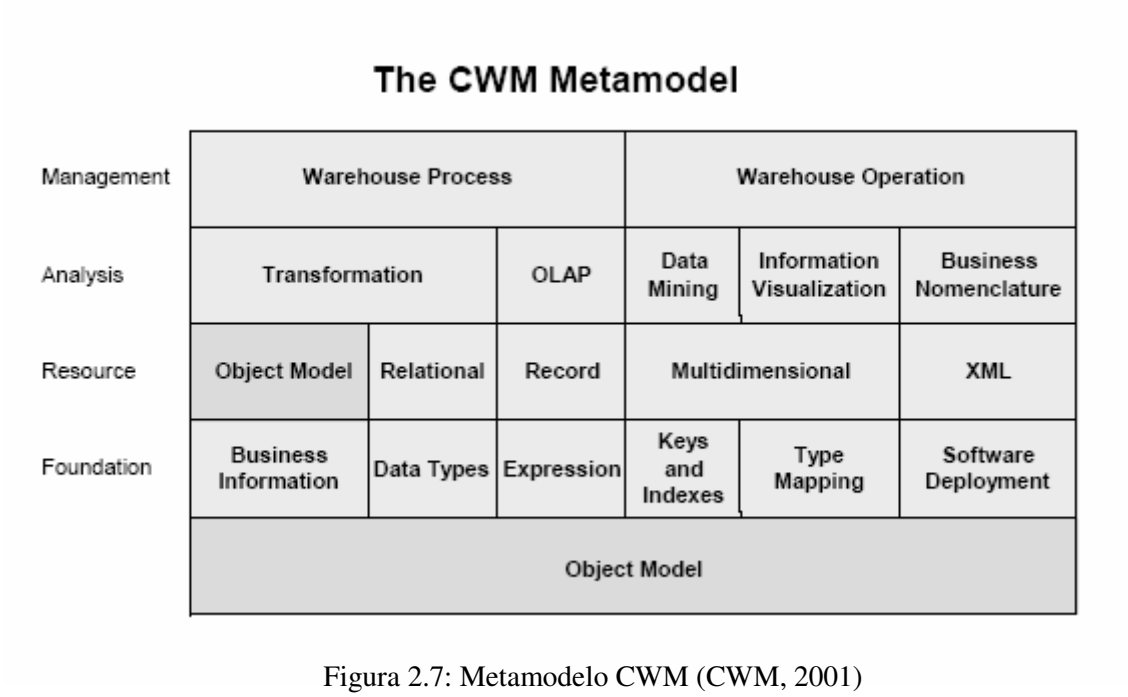

Figura 2.7: Metamodelo CWM (CWM, 2001)

- **Software Deployment:** contém classes que gravam qual *software* é usado no DW. Um pacote de *software* é representado como um objeto *SoftwareSystem*, que é um *subtype* de um *Subsystem*. Um *SoftwareSystem* pode referenciar um ou mais *Typesystems* que definem os tipos de dados suportados pelo *SoftwareSystem*. O mapeamento entre os tipos de dados e diferentes *TypeSystems* pode ser gravado como *Type Mappings*;
- **Type Mappings:** pacote responsável por suportar o mapeamento entre tipos de dados em diferentes sistemas. O propósito desse mapeamento é indicar os tipos de dados em diferentes sistemas que são suficientemente compatíveis e que os valores dos dados podem ser trocados entre esses sistemas.

- **Resource:** é responsável pelo suporte aos dados de sistemas legados e não legados, incluindo dados relacionais, orientados a registros e XML:
  - **Relational:** provê os tipos de metadados para descrever dados acessíveis através de uma interface relacional e segue o padrão SQL:1999;
  - **Record:** provê os tipos de metadados para descrição dos conceitos básicos de um registro e suas estruturas, ou seja, dados gravados em arquivos e banco de dados;
  - **Multidimensional:** corresponde uma representação genérica de um banco de dados multidimensional (MOLAP);
  - **XML:** provê os tipos de metadados para descrever fontes de dados em XML e é baseado na versão XML 1.0.

- **Analisys:** define um metamodelo para dados transformados, OLAP e informação de visualização:
    - **Transformation:** provê os tipos de metadados para descrever transformações entre diferentes tipos de fontes de dados;
    - **OLAP:** define um metamodelo dos construtores OLAP essenciais presentes nas aplicações e ferramentas OLAP;
    - **Data Mining:** é uma aplicação de um processo matemático ou estatístico com o propósito de extrair o conhecimento oculto em grande volume de dados;
    - **Information Visualization:** metadados para fornecer suporte ao problema do domínio ou informação publicada, ou seja, informação visualizada;
    - **Business Nomenclature:** usuários de DW necessitam de um bom conhecimento das informações e ferramentas existentes no DW, essas informações são extraídas deste pacote.
- **Management**: consiste na representação de metamodelos padrão de processos e resultados de operações em um DW (extração e cargas diárias):
    - **Warehouse Process:** provê os tipos de metadados para documentar o fluxo de processos utilizados para executar as transformações;
    - **Warehouse Operation** contém classes para o registro das operações diárias de um processo de DW.

Resumindo, o pacote "Foundation" é responsável pela especificação de vários elementos e serviços integrantes de um DW (expressões, informação do negócio, etc), enquanto que o pacote "Resource" é responsável pelo suporte aos dados de sistemas legados e não legados, incluindo dados relacionais, orientados a registros e XML. Já o pacote "Analysis" define um metamodelo para dados transformados, OLAP, informação de visualização, e a camada "Management" consiste na representação de metamodelos padrões de processos e resultados de operações em um DW (extração e cargas diárias). A Figura 2.8 exibe a dependência entre os pacotes principais do padrão CWM.

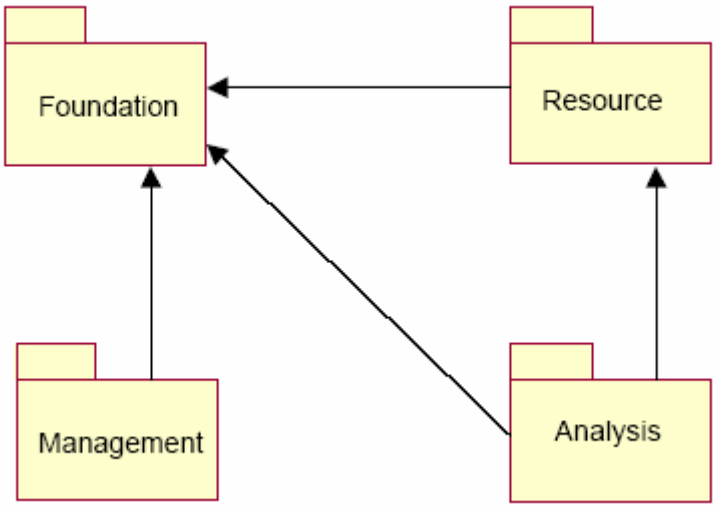

Figura 2.8: Dependência dos Pacotes do Metamodelo CWM (CHANG, 2000)

2.5 *EXTENSIBLE MARKUP LANGUAGE* (XML)

Uma das primeiras linguagens a conseguir um maior espaço para a programação *Web* foi a SGML (*Standard Generalized Markup Language*), ou traduzindo do inglês, Linguagem de Marcação Padrão Generalizada. De acordo com Pitts-Moultis e Kirk (2000), esta linguagem foi inventada pelo engenheiro da *Sun Microsystems*, Jon Bosak, sendo definida em 1986 pela ISO (*International Standards Organization*) como padrão ISO 8879, estava criado um poderoso padrão para linguagens de marcação, porém sua utilização seria um tanto complexa. Sendo que essa complexidade seria a grande causadora da pouca utilização do padrão SGML.

Ainda hoje, a principal linguagem de programação utilizada para desenvolvimento de páginas e aplicações no ambiente WWW é a HTML (*HiperText Markup Language*). A HTML foi concebida em 1992 por Tim Berners Lee e Robert Caillau no CERN (Centro Europeu de Pesquisas de Física de Partículas), sendo um linguagem descendente do SGML (PITTS-MOULTIS e KIRK, 2000). A linguagem original definia estritamente a estrutura lógica de um documento e não a sua aparência física. Porém, com a pressão dos usuários (principalmente da indústria), as versões posteriores da HTML foram forçadas a prover cada vez mais controle da aparência do documento.

O XML é um subconjunto do SGML. Segundo Buss e Kawassaki (2000), a linguagem XML simplifica significativamente o SGML, mantendo os principais pontos, mas tornando

seu aprendizado mais simples. Ela está no processo de padronização pela W3C (*World Wide Web Consortium*), que é o órgão que define os padrões para a *World Wide Web*.

Para Langiano et al. (2002, pg. 1), "as inovações oferecidas pela linguagem XML estão diretamente relacionadas com o contexto de Sistemas Distribuídos, sendo que essa tecnologia pode ser utilizada em plataformas diferentes, permitindo que seus dados sejam acessados por várias aplicações".

2.5.1 Definição

XML (*eXtensible Markup Language*) é uma linguagem que permite ao projetista criar seus próprios elementos de acordo com a aplicação que está sendo modelada, dando importância ao conteúdo e à estrutura da informação, sem se preocupar com o modo de apresentação de tal documento (LANGIANO et al., 2002).

De acordo com Tannenbaum (2002), XML é uma tecnologia que surgiu para atender às necessidades não atendidas pelo HTML, sendo mais promissora para a resolução de problemas referentes ao contexto *Web*. Esse paradigma da computação é o responsável por permitir que dados organizacionais sem estruturas, como relatórios internos e mensagens de *e-mail,* que são classificados como rudimentares, se tornem estruturas de metadados. Essa estrutura de metadados é a primeira separação do estilo de apresentação do conteúdo. De fato, o "parceiro" natural do XML é o XSL (*Extensible Style Language*), que por sua vez trata dos aspectos de formatação de documentos *Web* (TANNENBAUM, 2002).

O XML é um padrão para publicação e intercâmbio de documentos, recomendado pelo consórcio W3C. Como outras linguagens de marcação, o XML lida com instruções embutidas no corpo de documentos chamadas *tags*, que permitem a descrição dos dados. O XML tem como base linguagens mais antigas como SGML e HTML, sendo atualmente empregada na representação de estruturas de dados estruturados e semi-estruturados e seu intercâmbio na *Web* (LEAL, 2003) (TANNENBAUM, 2002).

A meta do XML é fornecer muitos dos benefícios de SGML não disponíveis em HTML e fornecê-los em uma linguagem que seja mais fácil de aprender e utilizar do que a SGML completa. Esses benefícios incluem *tags* definidas pelo usuário (conjunto extensível de *tags*), elementos encadeados e uma validação opcional do documento em relação a um esquema (LEAL, 2003).

Além disso, a utilização de HTML como uma linguagem de descrição de conteúdo apresentou várias inadequações, citando entre elas (LEAL, 2003):

- O conjunto de *tags* HTML é fixo e sua extensão para cobrir as exigências de novas aplicações acabava gerando a quebra do padrão ou a necessidade de um longo processo de padronização;
- As marcações HTML misturam anotações estruturais e visuais, produzindo documentos que são difíceis de serem processados por agentes de busca de informações na *Web*;

Em contrapartida, o XML atende a ambos os problemas, deixando que os autores do conteúdo definam e usem o conjunto de *tags* que melhor espelhe as propriedades estruturais e conceituais do conteúdo que eles querem publicar.

O esquema XML é passível de comparação com um esquema de banco de dados. Sendo que um esquema é um modelo para descrever a estrutura da informação. É um termo proveniente da área de banco de dados para escrever a estrutura de dados em tabelas relacionais.

Considerado no contexto XML, um esquema descreve um modelo para uma classe de documentos, sendo que o modelo descreve o possível arranjo de *tags* e textos dentro de um documento válido.

No contexto geral, um esquema também poderia ser visto como um acordo em um vocabulário comum para aplicações que realizam troca de documentos.

Diante de tudo isto, pode-se definir como uma característica importantíssima do XML o fato dele ser uma linguagem de marcação, sendo essa a primeira qualidade que distingue os documentos formatados com uma linguagem de marcação genérica do restante, sendo que o restante é normalmente lógico em sua ordem e altamente estruturado, incluindo regras específicas que afirmam onde várias estruturas do documento devem e precisam ser colocadas. De fato, documentos SGML e XML possuem uma estrutura em forma de árvore, que pode ser vista em qualquer um dos analisadores sintáticos (PITTS-MOULTIS e KIRK, 2000). Sendo que todos os demais componentes, como conteúdo e formatação, são construídos em torno dessa estrutura em árvore.

Segundo Langiano et al. (2002), o formato estruturado torna a descrição do conteúdo do documento mais fácil e mais amigável à formatação das seções dentro do mesmo. Sendo mais simples separar o conteúdo da apresentação e aplicar formatos específicos para fins específicos do mesmo grupo de dados.

A reformatação pode ser tranqüilamente alcançada, pois o desenvolvedor não precisa percorrer cada linha de código e mudar formatos específicos para itens específicos como eram exigidos no HTML, uma vez que a estrutura em forma de árvore torna mais fácil redesenhar documentos de marcação genéricos ou extrair dados para públicos diferentes (LANGIANO et al., 2002).

Através da flexibilidade do XML, é possível ter uma ótima representação das informações provenientes de um banco de dados. Desta forma, pode-se utilizar o XML para construir sites de comércio eletrônico que utilizem o XML para montar as suas páginas de descrição dos produtos e ainda que realizem as operações de consultas personalizadas, utilizando-se apenas das informações contidas nos documentos XML (MATOS JR., 2001).

Outras características consideráveis da linguagem XML são (LANGIANO et al., 2002), (MOURA, 2005), (PITTS-MOULTIS e KIRK, 2000):

- Inteligência: o XML é inteligente para mudar qualquer nível de complexidade. A marcação pode ser alterada de uma forma mais geral como "<CAO> Lassie </CAO>" para uma mais detalhada, como "<CAO> <VENHA_PARA_CASA> <COLLIE> Lassie </COLLIE> </VENHA_PARA_CASA> </CASA>". As idéias são bem marcadas para que "<VENDO_DOIS> duplo </VENDO_DOIS>" e "<MAIS_LICOR> duplo </MAIS_LICOR>" sejam sempre valores diferentes. A informação conhece a si mesma, não sendo necessário mais nenhuma idéia.
- Manutenção: o XML é simples de entender, pois contém somente idéias e marcações. As folhas de estilos e os *links* vêm separados e não embutidos no próprio documento. Sendo que cada um pode ser alterado separadamente quando necessário, com fácil acesso, portanto, facilitando a manutenção do código.
- Simplicidade: a especificação da SGML tem 300 páginas, enquanto que o XML apresenta em sua especificação 33 páginas, sendo retiradas idéias obscuras e desnecessárias, substituindo por idéias concisas, indo assim, direto ao assunto.
- Portabilidade: O XML é amplamente portável, sendo a razão da sua existência a força e a portabilidade. SGML tem a força, enquanto que o HTML tem a portabilidade, já o XML tem ambas as qualidades. XML pode ser navegada com ou sem o seu DTD (*Document Type Definition*, ou Definição de Tipo de Documento – normas que definem como as *tags* são estruturadas nos documentos XML), tornando o *download* mais rápido. O que um navegador precisa para ver um documento XML é ter noção que ele próprio e a folha de estilos controlam a

aparência, sendo que se uma validação estrita é necessária, o seu DTD pode acompanhá-lo e fornecer detalhes precisos de sua marcação.

- Integração de dados de diferentes fontes: a habilidade para consultar múltiplos bancos de dados, normalmente incompatíveis entre si, atualmente é muito difícil. A linguagem XML permite a combinação de dados estruturados oriundos de diferentes fontes. Componentes podem ser utilizados para a integração de dados oriundos de diferentes servidores e outras aplicações em uma camada intermediária (servidor de aplicações). Então, esses dados podem ser distribuídos aos clientes ou a outros servidores para posterior agregação, processamento e distribuição.
- Dados oriundos de múltiplas aplicações: a extensibilidade e flexibilidade de XML permitem que sua descrição se aplique a uma grande variedade de aplicações heterogêneas. Como o XML possui em sua própria estrutura a descrição dos dados, a descrição e o processamento no *front-end* podem ser efetuados sem que este possua uma descrição prévia do arquivo.
- Escalabilidade: dada a distinção existente no XML entre marcação e interface, pode-se inserir, em meio à descrição dos dados, procedimentos para a produção de diferentes visualizações dos resultados. Diante desse fato, transfere-se a carga de processamento para as máquinas cliente, o que reduz o tráfego no servidor e o tempo de resposta do *browser*. Além disso, o XML permite o envio de fragmentos de estruturas a título de atualização, acrescentando a escalabilidade do servidor, acarretando uma carga de processamento muito menor.
- Compressão: o XML apresenta boas taxas de compressão, devido à natureza repetitiva utilizada pelos delimitadores para descrever a estrutura de dados. A necessidade de compressão de arquivos XML é dependente da aplicação, sendo função da quantidade de dados a serem transferidos entre o servidor e o cliente. O XML pode utilizar o padrão de compressão em servidores e clientes HTTP 1.1.
- Múltiplas formas de visualização dos dados.
- Atualizações elementares dos arquivos, sem necessidade de reenvio de todos os dados, acarretando aumento da escalabilidade do servidor.
- Entrega de dados pela *Web*, pois é utilizado o protocolo http sem a necessidade de mudanças nas redes já instaladas.
- Suporte de diferentes fabricantes.

- Padrão aberto, alcançando interoperabilidade e suporte técnico desejados.
- Processamento e manipulação de dados no cliente, pois o padrão DOM (*Document Object Model*) permite a manipulação de arquivos XML a partir de *scripts* ou outras linguagens de programação.

Carvalho (2005) afirma que o padrão XML não é vinculado a nenhuma linguagem de programação, sendo assim, alcança a interoperabilidade de plataformas de programação e sistemas operacionais.

As Figuras 2.9 e 2.10 mostram arquivos XML gerados através de dois SGBD's distintos, sendo que a Figura 2.9 representa o XML do SGBD SQL Server 2000, enquanto que a Figura 2.10 retrata dado do SGBD MySQL. Analisando a parte em destaque de cada figura fica evidente que o XML é idêntico, a diferença é apenas referente aos dados naturais. Isto comprova como o XML resolve a questão da interoperabilidade entre tecnologias distintas, sendo perfeitamente possível juntá-los em um único arquivo, unindo o SGBD SQL Server 2000 juntamente com o SGBD MySQL.

De acordo com Leal (2003), o grupo de pesquisas em esquemas XML da W3C (*World Wide Web Consortium*) recebeu diversas propostas de linguagens. Porém, ainda não há nenhuma linguagem de esquema XML perfeita nos dias atuais, sendo que cada linguagem tem o seu diferencial.

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://tempuri.org/">
  - <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    - <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="pt-BR">
      - <xs:complexType>
        - <xs:choice maxOccurs="unbounded">
          - <xs:element name="Detran">
            - <xs:complexType>
              - <xs:sequence>
                  <xs:element name="Cpf" type="xs:string" minOccurs="0" />
                  <xs:element name="Nome" type="xs:string" minOccurs="0" />
                  <xs:element name="Dt_nasc" type="xs:dateTime" minOccurs="0" />
                  <xs:element name="Rg" type="xs:string" minOccurs="0" />
                  <xs:element name="Validade" type="xs:dateTime" minOccurs="0" />
                  <xs:element name="Multa" type="xs:decimal" minOccurs="0" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  - <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-
      com:xml-diffgram-v1">
    - <NewDataSet xmlns="">
      - <Detran diffgr:id="Detran1" msdata:rowOrder="0">
          <Cpf>11111111111</Cpf>
          <Nome>Michael Andretti</Nome>
          <Dt_nasc>1976-09-19T00:00:00.0000000-03:00</Dt_nasc>
          <Rg>0987654321</Rg>
          <Validade>2005-12-12T00:00:00.0000000-02:00</Validade>
          <Multa>0</Multa>
        </Detran>
      </NewDataSet>
    </diffgr:diffgram>
</DataSet>
```

Figura 2.9: XML da tabela de um SGBD SQL Server 2000 (CARVALHO, 2005)

Figura 2.10: XML da tabela de um SGBD MySQL (CARVALHO, 2005)

Uma linguagem de esquema XML apresenta como proposta prover regras de construções de marcação XML com que os esquemas são escritos. Sendo que o propósito de um esquema é definir e descrever uma classe de documentos XML usando estas regras para restringir e documentar o significado, uso e relações das partes constituintes: tipos de dados, elementos, atributos e anotações. Dessa forma, podemos dizer que a linguagem de esquema XML pode ser usada para definir, descrever e catalogar vocabulários para as classes de documentos XML (LEAL, 2003).

Finalizando, Löffler et al. (2002) utilizam dados em XML para realizar buscas em um sistema distribuído de arquitetura cliente/servidor, denominado *IFinder*. Esse sistema efetua buscas referentes a arquivos de áudio e vídeo por meio de metadados que estão descritos em XML. Essa pesquisa pode se dar por uma palavra chave, nome do dono da voz do áudio ou mesmo algum identificador de segmento de vídeo.

2.5.2 XMI (XML *Metadata Interchange Format*)

Tannenbaum (2002, pg 251) afirma que "o XMI apresenta como propósito integrar as tecnologias XML, UML e MOF, resultando em verdadeira troca pela *Internet*, não só de modelos UML, mas, também os exemplos desses modelos, sendo na maioria dos casos, os metadados". A mesma autora afirma também que, discordando do padrão das *tags* para troca de padrões, o comitê de padrões da W3C (desenvolvedores do XML) criou a especificação XMI.

O principal objetivo do XMI é permitir a facilidade na troca de metadados entre ferramentas de modelagem, baseadas na OMG (*Object Management Group*), UML e depósitos de metadados, baseado em OMG MOF, em ambientes heterogêneos distribuídos, integrando dessa forma (OMG XMI, 2001), (OMG MOF, 2002):

- XML: *eXtensible Markup Language*, o padrão W3C;
- UML: *Unified Modeling Language*, o padrão da modelagem OMG;
- MOF: *Meta Object Facility*, o padrão OMG para a modelagem e depósito de metadados.

Sendo, então, visível que a integração desses três padrões no XMI une a melhor tecnologia da OMG e da W3C em relação a metadados e à modelagem, possibilitando que desenvolvedores de sistemas distribuídos possam associar modelos de objetos e outros metadados através da *Internet*. Dessa forma, o "casamento" entre a tecnologia padrão referente a Objetos (UML) e padrão para trocas de dados através da *Internet* (XML), permite aos desenvolvedores trabalhar em um ambiente colaborativo de desenvolvimento para construir, armazenar e trocar modelos por meio da *Internet*, sem levar em consideração plataforma, linguagem de programação ou mesmo ferramenta utilizada.

O uso do XMI é bastante apropriado devido à variedade de recursos oferecidos para a manipulação e transformação de dados no formato XML, o que facilita a geração de outros arquivos a partir do XMI (SANTOS, 2004).

A Figura 2.11 apresenta um trecho de exemplo de código XMI, que representa um modelo contendo as classes "Departamento", "Instrutor", "Professor" e "Posdoc" e uma associação entre as classes "Instrutor" e "Departamento".

O padrão XMI é pretendido ser utilizado como uma ponte universal entre as ferramentas de desenvolvimento orientadas a objeto, evitando desta forma a criação de uma

variedade de formatos proprietários, cada um específico de cada fornecedor, à medida que cada ferramenta passe a importar e exportar metadados no formato XMI (MARINO, 2001).

A ausência de um padrão aberto tipo XMI tornariam as ferramentas necessitadas de mecanismos de tradução (importação e exportação) das informações em ambos os sentidos e entre todas as ferramentas necessárias para o desenvolvimento de aplicações. Sendo esta uma solução comprometida devido ao número de "pontes" a serem desenvolvidas (N*N-N pontos de tradução, sendo N o número de ferramentas) (ORENTEIN, 2001).

A Figura 2.12 mostra a importância do padrão XMI, pois sem a utilização da tecnologia, as 4 ferramentas precisariam de 12 pontos de tradução (pontes) para se comunicarem entre si. Finalizando, um documento XMI é um documento escrito em XML, que utiliza as *tags* definidas pela especificação do XMI para representar as informações, sendo que a composição de um documento XMI deve obedecer às regras definidas por esse padrão (AMARAL, 2002).

```
<?xml version="1.0" encoding="UTF-8" ?>
<XMI version="1.1" xmlns:UML="org.omg/UML1.3">
<XMI.header>
  <XMI.model xmi.name="Modelo Departamento" href="Departamento.xml" />
  <XMI.metamodel xmi.name="UML" href="UML.xml" />
</XMI.header>
<XMI.content>
  <UML:Class name="Departamento" xmi.id="1" />
  <UML:Class name="Instrutor" xmi.id="2" />
  <UML:Class name="Professor" xmi.id="3" generalization="Instrutor" />
  <UML:Class name="Posdoc" xmi.id="4" generalization="Instrutor" />
  <UML:Association>
      <UML:Association.connection>
        <UML:AssociationEnd name="instrutores" type="Instrutor" />
        <UML:AssociationEnd name="membroDe" type="Departmento" />
      </UML:Association.connection>
    </UML:Association>
</XMI.content>
</XMI>
```

Figura 2.11: Exemplo resumido de um arquivo XMI (SANTOS, 2004, pg. 58)

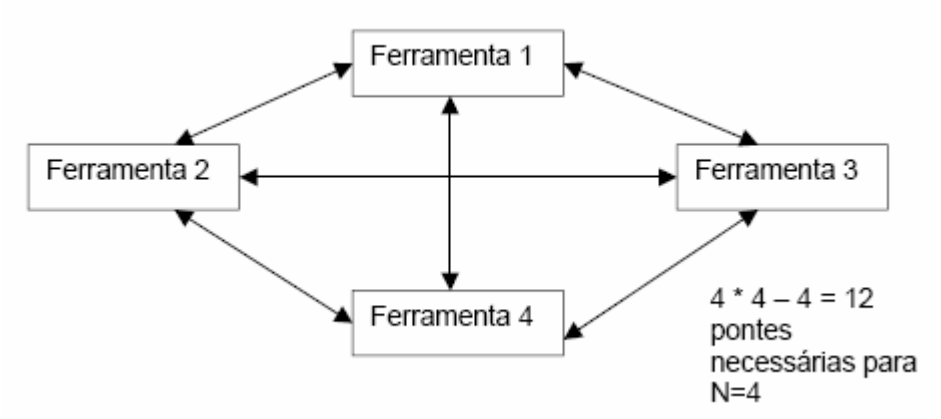

Figura 2.12: Exemplo de aplicação sem a utilização do padrão XMI (ORENTEIN, 2001, pg 4)

2.6 COMPARAÇÕES ENTRE PADRÕES

Segundo Marino (2001) os padrões OIM, CWM e XMI, descritos anteriormente, atuam em níveis diferentes no sentido de prover a interoperabilidade entre ferramentas e repositórios. Os padrões OIM e CWM atuam no nível conceitual, especificando metamodelos que podem ser vistos como os esquemas conceituais para os metadados, incorporando aspectos de aplicações específicas. O padrão XMI, por sua vez, atua no nível físico, preocupando-se em estabelecer um conjunto de regras capaz de gerar documentos XML a partir da especificação de modelos segundo o padrão MOF. Portanto, os padrões OIM e CWM abordam os aspectos de semântica, enquanto que o padrão XMI aborda os aspectos de sintaxe.

O padrão CWM, foi projetado para lidar somente com metadados no contexto de data warehousing e provê um *framework* para representação e troca de metadados sobre as fontes de dados (origem e destino) envolvidas e os processos responsáveis pela criação e gerenciamento destas fontes. Já o padrão OIM apresenta um escopo mais amplo, uma vez que foi projetado para acompanhar todas as fases de desenvolvimento de sistemas de informação, apresentando um conjunto de pacotes para áreas específicas, destacando-se o pacote *Database and Warehousing Model*, especificamente voltado para *data warehousing* (MARINO, 2001).

Para Pereira (2000), ambos os modelos, CWM e OIM, proporcionam o agrupamento de transformações únicas, ou seja, mapeamento de um único objeto: tabela, visão ou coluna para outro objeto em pacotes de transformação. Vetterli, Vaduva e Staudt (2000) evidenciam a grande diferença entre tais modelos: o OIM limita-se aos dados relacionais, enquanto o CWM não está vinculado a um modelo particular, ou seja, o CWM é independente de qualquer modelo prioritário.

2.7 TRABALHOS RELACIONADOS

Nesta seção são descritos os trabalhos que ofereceram alguma contribuição para o desenvolvimento desta pesquisa, principalmente em relação aos itens necessários para uma boa documentação de metadados. É descrito um modelo de metadados no contexto geral, proposto por Tannebaum (2002) e apresentado um modelo de metadados específico para sistemas KDD, proposto por Castoldi (1999).

2.7.1 Modelo de Metadados de Tannenbaum

Tannenbaum (2002) mostra um exemplo de um modelo de metadados. O passo inicial é identificar os "beneficiados" com o uso do metadados. Os "beneficiados" são identificados nas quatro primeiras colunas na Tabela 2.3, sendo no caso:

- Desenvolvedores;
- Gerentes de Projeto TI (Tecnologia da Informação);
- Usuários Finais;
- Catálogo SGBD (*Software* de banco de dados como o *Oracle* e DB2).

Já a quinta coluna "Fonte do Metadado" se responsabiliza por informar de onde é a procedência de um requisito do metadado.

A sexta coluna "Categoria do Metadado" procura classificar o metadado, sendo três possíveis alternativas (TANNENBAUM, 2002):

- Específico: metadado que se origina de uma fonte equivalente para a própria categoria dos "beneficiados", como exemplo, pode-se citar: metadados de gerenciamento de configuração (produção de programas e configuração de aplicações), metadados para modelagem orientada a objetos ou mesmo metadados que fornecem suporte para tomada de decisão. É importante salientar que a categorização depende da origem e de quem usará os metadados. Por exemplo, o requisito "Versão da Ferramenta CASE" é pertencente à categoria dos metadados específicos de ferramentas CASE;
- Único: espécie de metadados, onde são requisitados e necessitados por somente um agente, por exemplo, gerentes de projeto;
- Comum: metadados que possuem algo em comum, sendo usados e necessitados por todas as categorias de "beneficiários".

Tabela 2.3: Categorias dos Metadados, traduzido de Tannenbaum (2002, pg. 131 e 132)

Desenvolvedores	**Gerente de Projeto TI**	**Usuários Finais**	**Catálogo SGBD**	**Fonte do Metadados**	**Categoria do Metadados**
Nome do Negócio dos Elementos de Dados	Nome do Negócio do Elemento de Dados	Nome do Negócio do Elemento de Dados	Nome do Negócio do Elemento de Dados	Metadados de Ferramenta CASE	Comum
Nome do Banco de Dados	Nome do Banco de Dados		Nome do Banco de Dados	SGBD	
Nome da Tabela		Localização do Elemento de Dados	Nome da Tabela	SGBD	
Nome da Ferramenta CASE				Metadados da Ferramenta CASE	Específico
Versão da Ferramenta CASE				Metadados da Ferramenta CASE	Específico
Nome API do SGBD			Nome API do SGBD	SGBD	
Tipo do SGBD			Tipo do SGBD	Catálogo SGBD	
Nome da Fonte dos Elementos de Dados	Nome da Fonte do Elemento de Dados	Nome da Fonte do Elemento de Dados	Nome da Fonte do Elemento de Dados	Dicionário de Dados	Comum
Nome da Aplicação Legada	Nome da Aplicação Legada			Dicionário de Dados Interno	
	Gerente de Projeto de Aplicação Legada			Dicionário de Dados Interno	Único
Derivação de Regras do Elemento de Dados	Derivação de Regras do Elemento de Dados	Derivação de Regras do Elemento de Dados		Dicionário de Dados Interno	
Última Atualização do Dicionário de Dados Interno	Última Atualização do Dicionário de Dados Interno			Dicionário de Dados Interno	

Analisando a Tabela 2.3, percebe-se que alguns requisitos de metadados não foram classificados em Específico, Único ou Comum. Segundo Tannenbaum (2002), esses requisitos de metadados constituem sua própria categoria, a categoria "sem classificação". Sendo que esses metadados estão em demanda (é requerido por, pelo menos uma categoria de beneficiário), mas não apresentando a qualidade de "comum" em relação a outras categorias

de beneficiários. Na seqüência são relatadas as condições necessárias para um metadado ser considerado “sem classificação” (TANNENBAUM, 2002):

- Variações de metadados: em quase todas as situações, os “beneficiários” usam terminologia própria, possibilitando que uma categoria de beneficiário “chama” um “Elemento de Dados”, enquanto que outra categoria, “chama” o mesmo requisito de “Campo”. Outro exemplo é onde “Elementos de Dados de Localização” é equivalente a “Nome do Banco de Dados”, sendo necessário remover “Elementos de Dados de Localização” da lista de “sem classificação”, enquanto que “Nome do Banco de Dados” deve integrar a lista de “Comum”. Cerca de 20 por cento dessas variações comumente são eliminadas pela análise de um analista experiente;
- Múltiplas fontes de metadados: quando um requisito de metadado apresenta fonte de mais de um lugar, sendo quase sempre variações de metadados (cada fonte pode chamar a mesma “parte” do metadado por um nome diferente), ficando obscura a escolha de qual usar. Um exemplo é quando a fonte de um metadado gravado também é fonte de outro requisito de metadado;
- Múltiplas categorias de beneficiários: se mais de um grupo quer uma “parte” do metadado, mas o próprio metadado não é requerido universalmente, isto é, não é Comum, então esse metadado é considerado “sem classificação”.

A Figura 2.13 mostra alguns requisitos de metadados sendo removidos, como (TANNENBAUM, 2002):

- “Nome da Aplicação Legada”, sendo determinado para ser uma variação de “Nome da Aplicação Fonte”;
- “Nome do Arquivo Fonte”, sendo determinado para ser uma variação de “Nome do Banco de Dados”;
- “Nome do Arquivo Fonte”, sendo determinado para ser uma variação de “Tipo SGBD”;
- “Nomes dos Atributos” e “Definições dos Atributos”, sendo determinados para serem uma variação de “Elemento dos Dados do Nome do Negócio” e “Definições dos Elementos dos Dados”, respectivamente;

- "Nomes das Entidades" e "Nomes dos Relacionamentos", embora armazenados em "Repositório de Metadados CASE", que a partir de agora não serão mais utilizados por nenhum metadado "beneficiário".

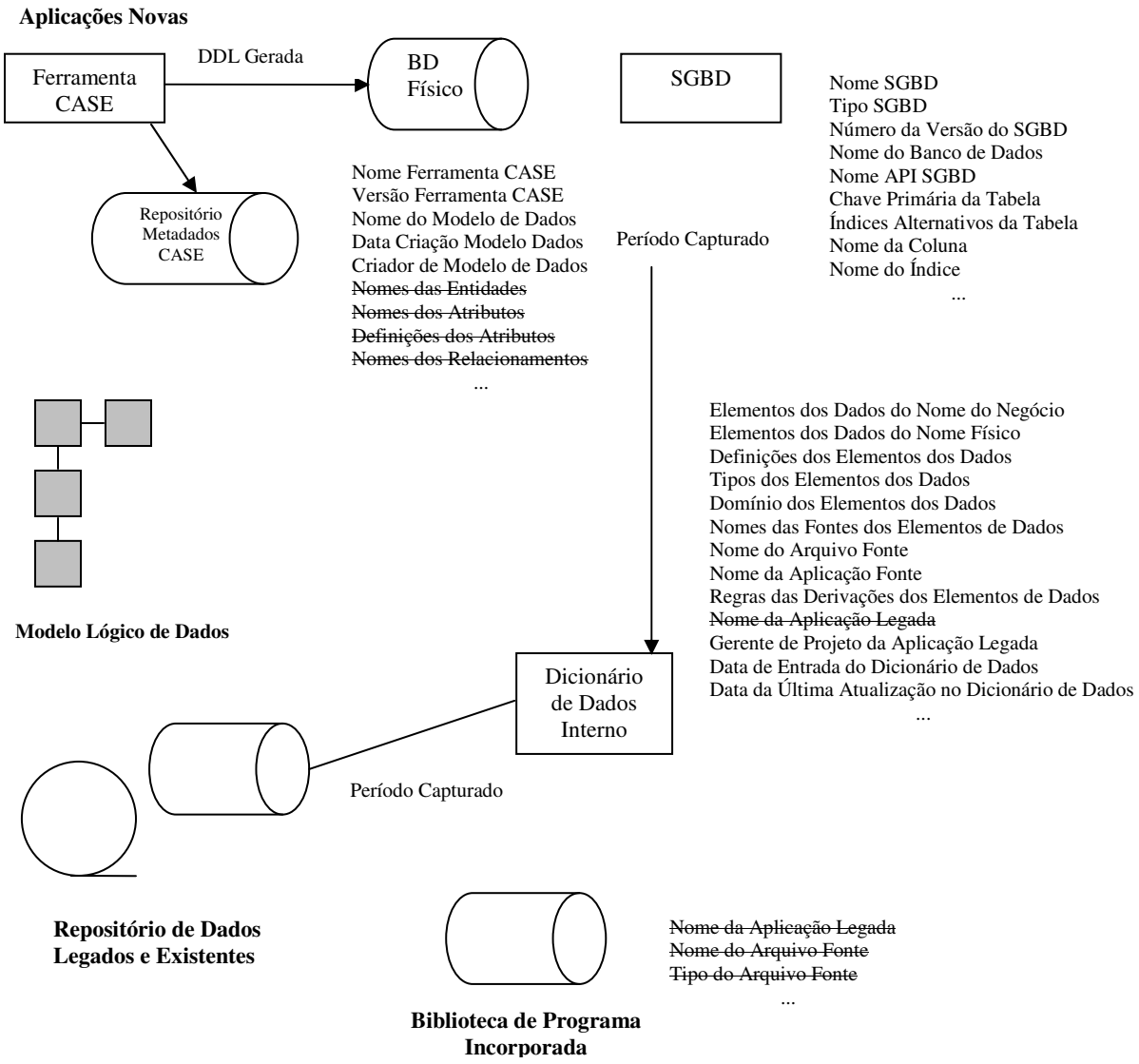

Figura 2.13: Fluxo de atualização com variações consideradas (TANNENBAUM, 2002, pg. 142)

2.7.2 Modelo de Metadados de Castoldi

Castoldi (1999) propôs um modelo de metadados que oferece suporte à fase de preparação de dados de sistemas de descoberta de conhecimento em banco de dados. Esse modelo de metadados cobre as necessidades básicas de uma ferramenta para preparação de dados, relacionadas na Tabela 2.4 e classificadas como:

- Definição de tabelas;
- Definição de regras de transformação e povoamento;
- Definição da freqüência com que cada regra deve ser executada;
- Documentação dos metadados.

Tabela 2.4: Descrição das tabelas que compõem o Modelo de Metadados (CASTOLDI, 1999, pg 16)

Tabela	**Definição**
Base	Conexão à base de dados. Tanto bases operacionais quanto bases históricas devem ser informadas aqui
Tabela	Contém informações a respeito das tabelas que estão nas bases de dados.
Coluna	Contém informações a respeito das colunas, como nome e tipo.
Índice	Chave pela qual tabela é classificada.
IndColuna	Coluna pertencente à chave.
Povoamento	Regras para o povoamento das tabelas. Devem ser permitidas apenas para bases históricas.
PovColuna	Expressão utilizada para povoar a coluna.
PcDireto	O dado é armazenado sem transformação.
PcDiscreto	Utilizado para discretizar valores contínuos em categorias discretas (como faixas salariais).
PcdItem	Item discreto. Utilizado em conjunto com PcDiscreto.
PcExpressao	Utilizado quando não for possível expressar a transformação dos dados através de atribuição direta ou categorização.
Expressão	Expressão utilizada por alguma entidade no modelo dos metadados.
ExprArgumento	Argumento da expressão utilizada pelos metadados.
Operador	Relação dos operadores pertencentes à gramática utilizada nas expressões.
Argumento	Relação dos argumentos necessários a um operador. Todos os argumentos devem ser preenchidos para que a expressão tenha um funcionamento correto.
TipoDado	Armazena os tipos de dados retornados pelas expressões e utilizados nas definições das colunas.
Comentário	Armazena a documentação dos metadados.
Gatilho	Permite automatizar o processo de povoamento. Cada gatilho possui uma regra de povoamento associada e uma condição de disparo.
GEvento	Faz referência a um trigger de banco de dados. Quando este for executado, o gatilho dispara a regra de povoamento.
GEvHistorico	Registra o número de vezes que o gatilho foi disparado. Também armazena o valor da chave que disparou o gatilho.
GExpressao	Expressão contendo a condição de disparo. O gatilho é disparado quando a Expressão atinge o valor alvo.
GExHistorico	Registra o número de vezes que o gatilho foi disparado. Também armazena o valor da expressão que disparou o gatilho.

2.8 CONSIDERAÇÕES FINAIS

Os conceitos apresentados neste capítulo têm como objetivo fornecer suporte para a concepção da proposta descrita no Capítulo 3. Há de se considerar que um DW é apenas um subconjunto de um contexto maior, denominado ambiente de descoberta de conhecimento. Sendo que nesse ambiente, a presença de metadados é imprescindível para o seu funcionamento adequado.

Os modelos e padrões relativos aos metadados apresentados foram de suma importância para a elaboração do modelo de metadados. No próximo capítulo é apresentado o modelo de metadados em XML proposto para sistemas KDD.

3 MODELO DE METADADOS PROPOSTO PARA SISTEMAS KDD

Os metadados são tão importantes que o desenvolvimento de um projeto de DW deve primeiro se iniciar por eles (HURWITZ, 1996). Assim, é necessário definir um modelo de metadados para representação das informações sobre como os dados são extraídos, transformados e armazenados no DW para serem utilizados em sistemas KDD.

Já em um contexto geral, Demarest (1997) cita alguns fatores de insucesso na concepção de um projeto de DW:

- Engenharia de dados problemática;
- Esquema de projeto não realístico;
- Falta de um método de projeto.

A dificuldade do projeto de DW somado com a falta de um padrão para os metadados são os motivos para esse empreendimento.

Neste capítulo é apresentado um modelo de metadados definido em XML proposto para sistemas KDD.

3.1 MODELO DE METADADOS

De acordo com as necessidades de um sistema KDD e com base no modelo de metadados genérico proposto por Tannenbaum (2002), foi proposto um modelo de metadados.

Neste modelo são disponibilizadas ao usuário de um sistema KDD todas as informações necessárias para o seu entendimento geral, fornecendo informações como, por exemplo, as fontes de dados que alimentam o DW, os dados contidos no DW, relacionamento entre os dados de origem e os do DW, informações de localização dos dados de origem, da área de estágio e do DW, etc. Assim, os metadados servirão como um guia na criação, povoamento e acesso ao DW.

Tendo como base as principais etapas no desenvolvimento de um Sistema KDD (Pré-processamento, Mineração e Pós-processamento), referenciadas na Seção 2.1, e a arquitetura de DW proposta por Menolli (2004), os itens de metadados identificados como sendo

fundamentais ao modelo proposto são descritos a seguir, sendo também representados no Apêndice A deste trabalho:

1) Pré-Processamento:

- Busca:
  - Endereços das Fontes de Origem: imprescindível saber onde cada banco de dados que concebeu o DW está localizado, identificando, por exemplo, qual computador da rede, qual diretório, ou seja, informações pertinentes à localização;
  - Tipo de SGBD de cada Banco de Dados de Origem: característica do SGBD (centralizado, distribuído, heterogêneo, etc.);
  - Versão do SGBD dos Bancos de Dados de Origem: diante da variedade de SGDB's e mesmo quanto a versões de um mesmo SGBD, é importante saber, qual a versão do SGBD utilizado, se é recente ou antiga;
  - Nomes dos Usuários e Senhas de Origem: fundamental, uma vez que para conectar a um banco de dados pode ser necessário informar qual usuário com sua respectiva senha está conectando a esse banco de dados;
  - Nomes dos Bancos de Dados de Origem: todos os nomes de Bancos de Dados que posteriormente terão seus dados sumarizados no DW, como exemplo, BDADMIN em *Oracle*, *MySQL* ou *SQLServer 2000*, entre outros;
  - Nomes das Tabelas, Atributos e Chaves de Origem: além do nome do banco de dados, é fundamental saber, quais tabelas o compõem, quais atributos compõem essas tabelas e quais desses atributos são chaves;
  - Nomes de *Aliases* de Origem: mesmo sabendo o nome de uma tabela, pode ser necessário ter um nome mais representativo, desmembrando abreviações utilizadas em nomes de tabelas, relacionamentos e atributos, tornando o entendimento do ambiente por parte do usuário final mais simples, ou seja, usar uma nomenclatura conhecida ao negócio em questão;
  - Nomes de *Drivers* e APIs de Origem: informações sobre todos os *drivers* e APIs utilizados para a concepção do DW, por exemplo, para conectar ao banco de dados *MySQL* é necessário possuir o *driver* para conexão via ODBC chamado "MySQL ODBC 3.51 Driver";
  - Tipos de Arquivos Fontes de Origem: qual a extensão do arquivo fonte de dados, por exemplo, sendo os dados provenientes do *Access*, o arquivo fonte de dados tem extensão "mdb", ou se é do *Excel* a extensão é "xls";

- Nomes das Aplicações Legadas: o nome de cada aplicação que alimenta os dados do DW. Por exemplo, entre os dados existe uma parte proveniente de um sistema em COBOL cujo nome da aplicação é "Sistema Administrativo-Financeiro 2.1".

- Transformação:
  - Itens de Dados Buscados nas Fontes: quais itens (atributos da tabela) foram extraídos das fontes de dados de origem;
  - Transformações Realizadas nos Dados/Regras de Transformações: devem ser registrados todos os tipos de transformações ocorridas nos dados, como por exemplo, retirada dos espaços em brancos, padronização para atributo tipo data. Sendo que essas transformações podem ser extraídas de uma regra pré-definida, por exemplo, todos os valores monetários devem conter duas casas decimais.

- Carga:
  - Estágio:
    - Itens de Dados Transformados: os itens (atributos da tabela) que sofreram transformações;
    - Endereços das Fontes de Dados de Estágio: onde os dados, após sofrerem as transformações necessárias, serão gravados, sendo responsável pela indicação do computador, o diretório de destino, ou seja, todas as informações pertinentes à localização;
    - Tipo de SGBD do Banco de Dados de Estágio: característica do SGBD (centralizado, distribuído, heterogêneo, etc.);
    - Versão do SGBD do Banco de Dados de Estágio: diante da variedade de SGDB's e mesmo quanto a versões de um mesmo SGBD, é importante saber qual a versão do SGBD utilizado, se é recente ou antiga;
    - Nomes dos Usuários e Senhas de Estágio: fundamental, uma vez que para conectar a um banco de dados pode ser necessário informar qual usuário com sua respectiva senha está conectando a esse banco de dados;
    - Nomes dos Bancos de Dados de Estágio: todos os nomes de Banco de Dados desta etapa, como exemplo, BDESTAGIO em *Oracle*, *MySQL* ou *SQLServer 2000*, entre outros;

- Nomes das Tabelas, Atributos e Chaves de Estágio: além do nome do banco de dados, é fundamental saber quais tabelas o compõem, quais atributos compõem essas tabelas e, também, entre os atributos quais são chaves;
- Nomes de *Aliases* de Estágio: mesmo sabendo o nome de uma tabela, pode ser necessário ter um nome mais representativo, desmembrando abreviações utilizadas em nomes de tabelas, relacionamentos e atributos, tornando o entendimento do ambiente por parte do usuário final mais simples, ou seja, usar uma nomenclatura conhecida ao negócio em questão;
- Nomes de *Drivers* e APIs de Estágio: informações sobre todos os *drivers* e APIs utilizados na etapa de "Estágio", por exemplo, para conectar ao banco de dados *MySQL* é necessário possuir o *driver* para conexão via ODBC chamado "MySQL ODBC 3.51 Driver";
- Tipos de Arquivos Fontes de Estágio: qual a extensão do arquivo fonte de estágio, por exemplo, sendo os dados armazenados no *Access*, o arquivo fonte de dados tem extensão "mdb", ou se é do *Excel* a extensão é "xls";
- Freqüência da Carga dos Dados de Estágio: com que freqüência os dados são carregados na área de estágio, se é diária, semanal, mensal, trimestral, semestral, anual ou esporádica, quando houver alterações nos dados das fontes de dados;
- Volume de Dados do Estágio: a quantidade de informação presente na área de Estágio, diante dessa informação, pode-se, por exemplo, concluir que é necessário sumarizar mais os dados do DW;
- Data do Último Estágio: quando ocorreu o último estágio, diante dessa data, dependo do negócio em questão, pode-se concluir ser necessário uma carga imediata dos dados da área de estágio;
- Log: Identificação do usuário responsável pela carga de dados na área de estágio.

o Povoamento:

- Itens de Dados do Estágio: itens (atributos da tabela) da área de estágio que conceberão o DW;

- Endereços das Fontes de Dados do DW: onde os dados do DW foram gravados, sendo responsável pela indicação do computador, o diretório de destino, ou seja, todas as informações pertinentes à localização;
- Tipo de SGBD do Banco de Dados do DW: característica do SGBD (centralizado, distribuído, heterogêneo, etc.);
- Versão do SGBD do Banco de Dados do DW: diante da variedade de SGDB's, é importante saber qual a versão do SGBD utilizado, se é recente ou antiga;
- Nomes dos Usuários e Senhas do DW: fundamental, uma vez que para conectar a um banco de dados, pode ser necessário informar qual usuário com sua respectiva senha está conectando a esse banco de dados;
- Nomes do Banco de Dados do DW: nome do Banco de Dados onde os dados sumarizados foram gravados, como exemplo, BDDW em *Oracle*, *MySQL* ou *SQLServer 2000*, entre outros;
- Nomes das Tabelas, Atributos e Chaves do DW: além do nome do banco de dados, é fundamental saber, quais tabelas o compõem, quais atributos compõem essas tabelas, e também entre os atributos quais são chaves;
- Nomes de *Aliases* do DW: mesmo sabendo o nome de uma tabela, pode ser necessário ter um nome mais representativo, desmembrando abreviações utilizadas em nomes de tabelas, relacionamentos e atributos, tornando o entendimento do ambiente por parte do usuário final mais simples, ou seja, usar uma nomenclatura conhecida ao negócio em questão;
- Nomes de *Drivers* e APIs de DW: informações sobre todos os *drivers* e APIs utilizados para conectar ao DW;
- Tipos de Arquivos Fontes do DW: de qual extensão é tal arquivo fonte de dados do DW;
- Operações de Povoamento: este item apresenta a mesma função do item de metadado "Transformações Realizadas nos Dados/Regras de Transformações". Um exemplo deste item poderia ser a necessidade de documentar uma fórmula usada para sumarizar um determinado

atributo do DW, informando que o atributo é proveniente de uma junção de atributos de tabelas distintas dos sistemas de origem;

- Freqüência de Povoamento: com que freqüência os dados serão carregados no DW, se é diária, semanal, mensal, trimestral, semestral ou anual;
- Volume de Dados do Povoamento: a quantidade de informação presente no "Povoamento" que conceberá o DW;
- Data do Último Povoamento: quando ocorreu o último povoamento do DW, diante dessa data, dependem do negócio em questão, pode-se concluir ser necessária a atualização imediata dos dados do DW;
- Log: Identificação do usuário responsável pelo povoamento de dados no DW.

2) Aplicação de Técnicas de Análise:

- Informações sobre Algoritmos de Mineração: informação sobre qual tipo de algoritmo é mais adequado para alcançar determinado objetivo, por exemplo, um algoritmo de agrupamento tem como finalidade agrupar os dados com características similares;
- Informações de Acesso aos Algoritmos de Mineração: informações referentes à localização do algoritmo, como proceder para ter acesso a ele;
- Informações sobre Ferramentas OLAP: caracterização das ferramentas OLAP envolvidas no ambiente de descoberta de conhecimento;
- Informações de Acesso às Ferramentas OLAP: localização das ferramentas OLAP disponíveis no ambiente de descoberta de conhecimento.

3) Pós-Processamento:

- Itens de Dados Povoados Utilizados na Aplicação da Técnica de Análise: quais itens (atributos da tabela) "sofreram" a aplicação de uma determinada técnica de análise;
- Técnica Aplicada na Geração do Resultado: qual técnica de análise foi utilizada para se chegar ao resultado exibido;
- Nome do Arquivo de Resultado: nome físico do arquivo que armazena o resultado alcançado;

- Endereço do Arquivo de Resultado: endereço de armazenamento do arquivo mantenedor do resultado alcançado;
- Nomes dos Negócios: item responsável por relatar o nome de negócio de cada aplicação envolvida no DW, por exemplo, vendas no varejo, vendas no atacado, locação de automóveis, etc.

Dessa forma, o XML dos metadados foi estruturado nomeando os três nodos principais como sendo: “Pré-processamento”, “Aplicação de Técnicas de Análise” e “Pós-processamento”.

Contudo, era necessário adequar o modelo proposto a um padrão comum de metadados para sistemas KDD. Dois padrões foram descritos no Capítulo 2, OIM e CWM, sendo escolhido o CWM devido ao fato de ser direcionado para sistemas de descoberta de conhecimento, enquanto que o padrão OIM é genérico para sistemas de informação.

Na Figura 3.1 é mostrado o XML do modelo de metadados proposto, contendo as *tags* principais do padrão CWM. O XML completo está representado no Apêndice B.

É válido observar que somente os pacotes principais (*Foundation, Resource, Analysis* e *Management*) da proposta CWM foram utilizados, devido principalmente ao fato do modelo genérico de metadados atender às principais necessidades de um ambiente de descoberta de conhecimento. Assim, não foi necessário aprofundar nas classes desses pacotes acima citados, uma vez que a utilização dessas classes deixaria o modelo de metadados com informação em excesso, sendo que o objetivo desse estudo foi elaborar um modelo da forma mais concisa possível para o funcionamento do ambiente em questão. O detalhamento dessas classes em um ambiente de descoberta de conhecimento poderia ser uma sugestão de um trabalho futuro.

Na Figura 3.1, as tags XML “Search”, “Stage_Data” e “Multidimensional” representam etapas e atividades do processo KDD, simbolizando informações sobre as fontes de origem, da área de estágio e do DW, respectivamente.

```
<?xml version="1.0" standalone="yes"?>
<Metadata_Model>
 <Foundation>
      ...
 </Foundation>
 <Resource>
   <Search>
       <Relational>
            <Address_Source_Origin>\\servidor2\BancosDados\</Address_Source_Origin>
            <Type_DBMS_DB_Origin>centralizado</Type_DBMS_DB_Origin>
            <Version_DBMS_DB_Origin>2.0</Version_DBMS_DB_Origin>
```

```
                <User_Password_Origin>josecarlos|123</User_Password_Origin>
                <DB_Origin>BDAdmFin|MySQL</DB_Origin>
                <Aliases_Origin>Sistema_Vendas</Aliases_Origin>
                <Drivers_API_Origin>MySQL ODBC 3.51 Driver</Drivers_API_Origin>
                <Type_File_Source_Origin>.opt|.frm|.myd|.myi</Type_File_Source_Origin>
                <Legacy_Application>Sistema Administrativo-Financeiro 2.1</Legacy_Application>
                <Tables_Attributes_Keys_Origin>
                  <Table>Produto</Table>
                  <Attributes>
                      <Name>Cod_Produto</Name>
                      <Type>Integer</Type>
                      <Name>Descrição</Name>
                        …
                  </Attributes>
                  <Keys>Cod_Produto</Keys>
                  <Table>Vendas</Table>
                  <Attributes>
                     <Name>Cod_Venda</Name>
                     <Type>Integer</Type>
                       …
                  </Attributes>
                  <Keys>Cod_Venda</Keys>
                </Tables_Attributes_Keys_Origin>
            </Relational>
        </Search>
        <Pre_Processing>
            <Load>
              <Stage_Data>
                  <Relational>
                    …
                  </Relational>
              </Stage_Data>
              <Fill_Data_DW>
                 <Items_Datas_Stage>
                     <Multidimensional>
                          …
                     </Multidimensional>
                 </Items_Datas_Stage>
              </Fill_Data_DW>
            </Load>
        </Pre_Processing>
    </Resource>
    </Analysis>
        ...
    </Analysis>
    <Management>
          …
    </Management>
</Metadata_Model>
```

Figura 3.1: Modelo de Metadados em XML Resumido

3.2 CONSIDERAÇÕES FINAIS

Como foi visto neste capítulo, o modelo de metadados proposto é amplamente baseado nas principais etapas do desenvolvimento de um Sistema KDD (Pré-processamento, Mineração e Pós-processamento), definidas por Feldens et al. (1998), nos itens de metadados relacionados por Tannenbaum (2002), além da arquitetura de DW definida por Menolli (2004).

Para refinar o modelo foi preciso padronizá-lo de acordo com um padrão aceito na comunidade computacional, sendo neste trabalho utilizado o padrão CWM para a organização do metadados. A escolha do padrão CWM se deve principalmente ao fato de ser um modelo de estrutura de metadados específico para sistemas de ambiente de descoberta de conhecimento. Enquanto que o padrão OIM é mais genérico, uma vez que é baseado no desenvolvimento de sistemas de informação em contexto geral, enquadrando várias espécies de sistemas computacionais.

O modelo de metadados apresentado procura registrar as informações fundamentais que um executivo ou analista de sistemas necessitam conhecer em um sistema KDD. Para um executivo pode ser útil a informação relativa ao resultado da aplicação de um determinado algoritmo de mineração de dados em determinada época. Já para um analista de sistemas, uma informação mais técnica como a localização das tabelas que contemplam a área de estágio.

A linguagem XML foi utilizada para proporcionar a interoperabilidade do modelo de metadados proposto.

4 GERENCIADOR DE METADADOS PROPOSTO

Para a utilização do modelo de metadados proposto, descrito no capítulo anterior, foi desenvolvido um componente de software. Este componente está baseado na arquitetura de referência proposta por Valentin (2006). Assim, inicialmente neste capítulo, esta arquitetura é descrita sucintamente e, na seqüência, é apresentada a especificação do componente que é responsável pelo gerenciamento de metadados.

4.1 ARQUITETURA DE REFERÊNCIA PROPOSTA POR VALENTIN

A arquitetura de referência proposta por Valentin (2006) apresenta como característica principal o fato de ser dividida em camadas, sendo que para efeito de comunicação entre essas camadas é utilizado o conceito de arquitetura orientada a serviço, denominada SOA (*Service Oriented Architecture*). Dessa forma, é definida uma camada para agregar os processos de negócio e outra para os serviços da aplicação, além de camadas que oferecem suporte para a realização dos serviços, entre eles, os serviços relativos ao gerenciamento dos metadados (VALENTIN, 2006).

A Figura 4.1 mostra as camadas que compõem a arquitetura proposta por Valentin (2006). A camada de visualização disponibiliza para o usuário a interface de comunicação. Para realizar um serviço, a camada de visualização "chama" a camada de processo, como por exemplo, o processo "executar a carga no DW". Este processo por sua vez, necessitará de serviços, recorrendo assim à camada de serviços que também necessitará de suporte da camada de apoio. A relação entre a camada de serviços e a camada de apoio se dá pelo motivo de alguns serviços necessitarem de um gerenciamento mais específico de determinadas tecnologias para desenvolver suas funções.

A camada de apoio é onde se encontra o gerenciador de metadados, que é o componente responsável pelo registro e recuperação dos dados necessários na construção e acesso ao DW.

Para facilitar o entendimento das ações a serem realizadas em um sistema KDD, são mostrados na Figura 4.2 os principais casos de uso. Há de se considerar que na realização de cada caso de uso é necessária a interação com o gerenciador de metadados. A Figura 4.3 mostra a interação do caso de uso "Definir Origem" com o Gerenciador de Metadados.

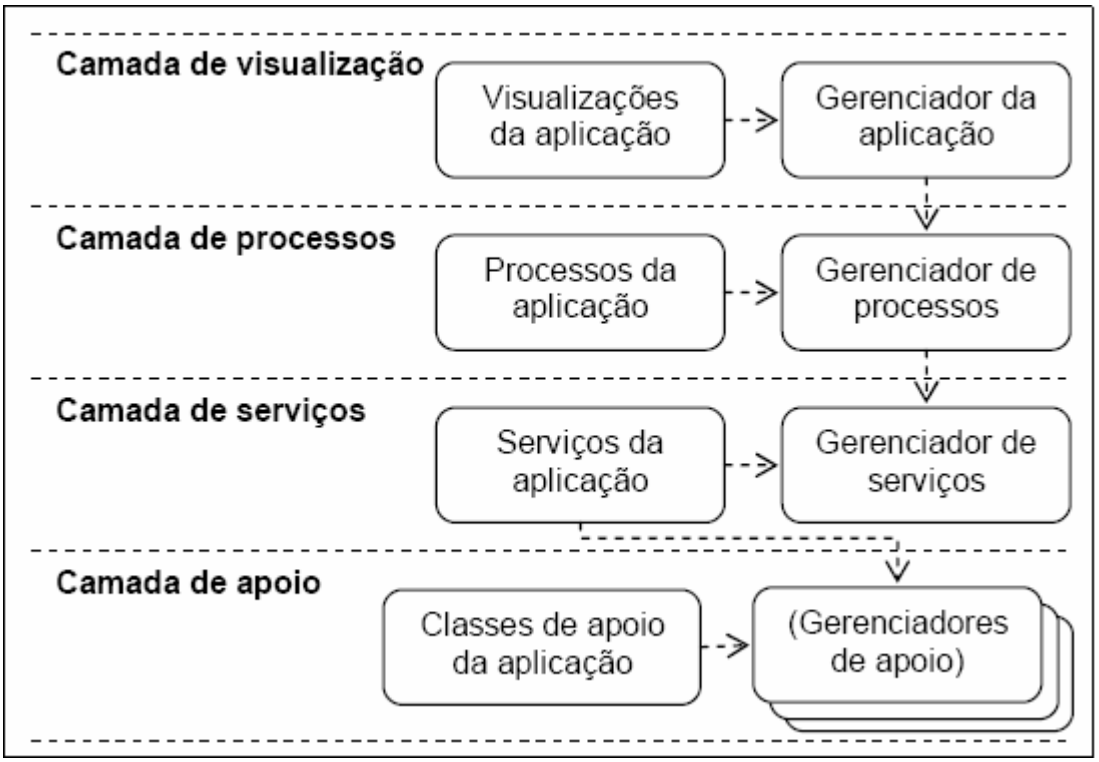

Figura 4.1: Camadas que Compõem a Arquitetura de Referência proposta por Valentin (2006)

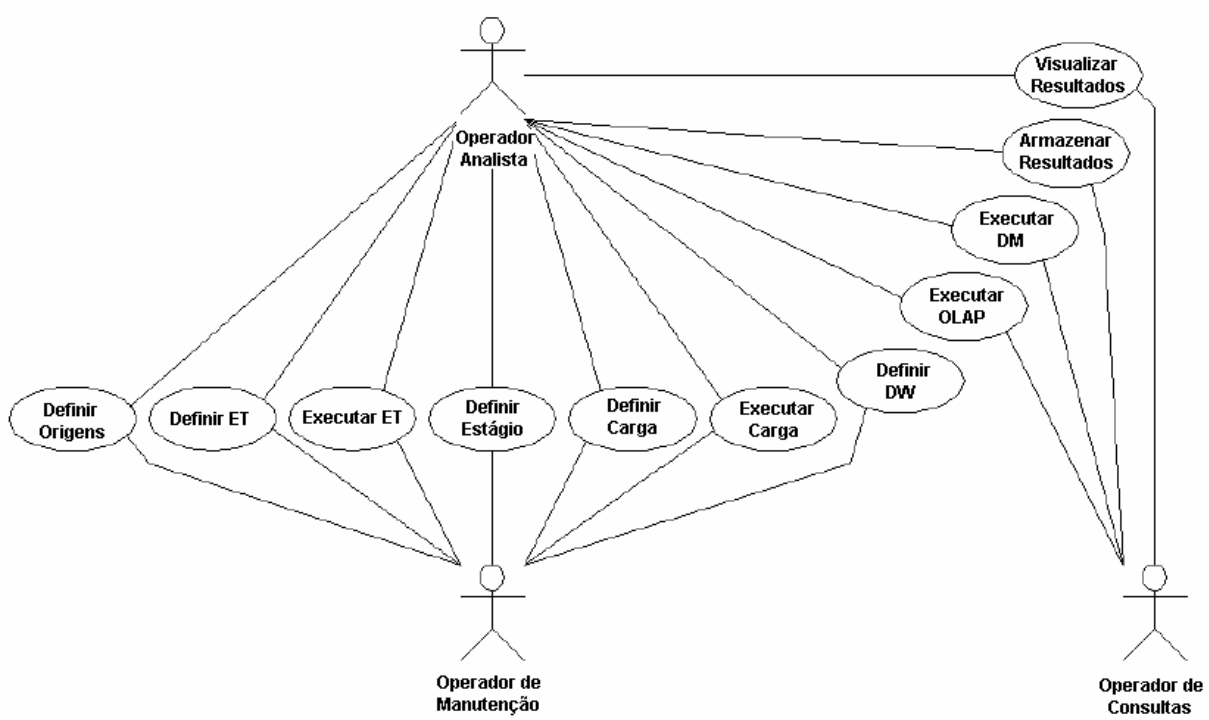

Figura 4.2: Casos de Uso do Sistema KDD Representado pelo Modelo de Referência de Sistemas KDD (VALENTIN, 2006)

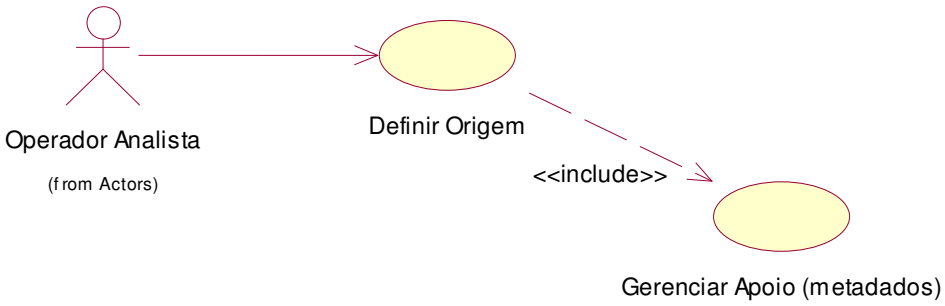

Figura 4.3: Interação do Caso de Uso “Definir Origem” com o Gerenciador de Metadados

Valentin (2006) mostra na Figura 4.4 as entidades (conexões, tabelas e atributos), serviços, processos e visualizações que foram obtidos do caso de uso “Definir Origem”. Sendo que as estruturas das bases de origem são definidas pelos sistemas legados, que por sua vez são os responsáveis por essas bases de dados.

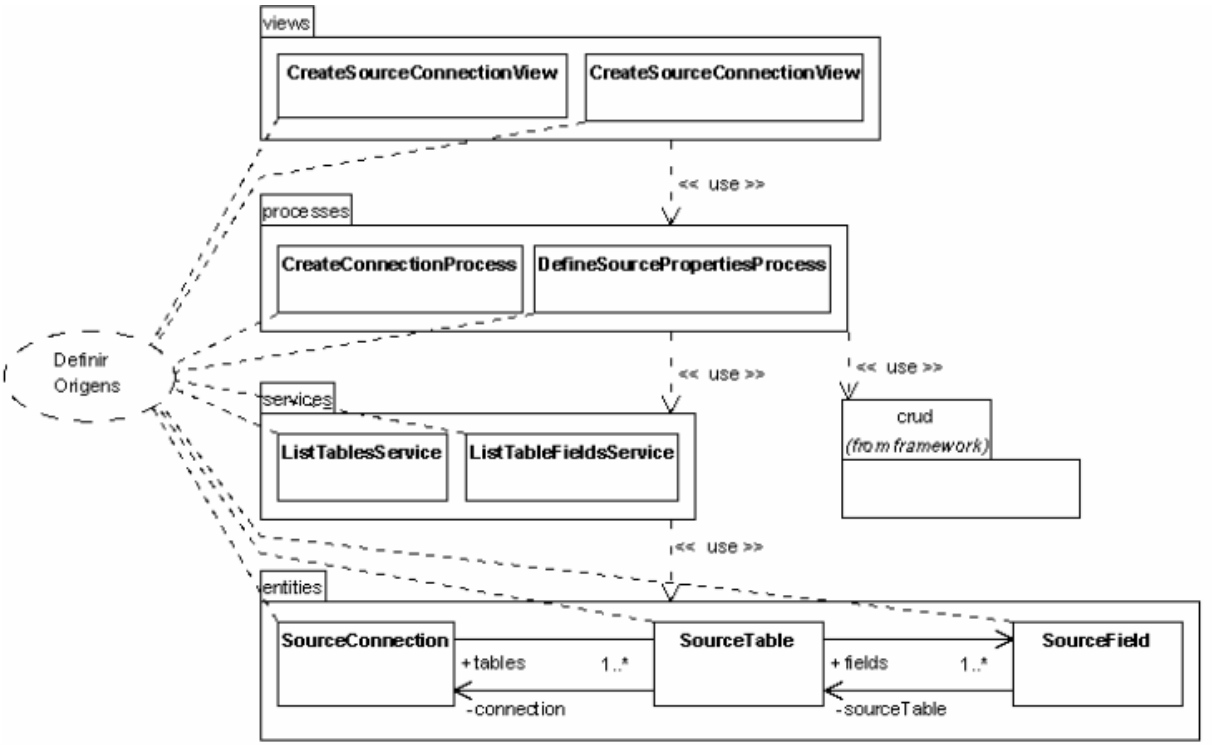

Figura 4.4: Entidades, serviços, processos e visualizações definidas pelo caso de uso Definir Origens (VALENTIN, 2006)

É de responsabilidade da entidade *SourceConnection* armazenar as informações de uma conexão com as bases de dados de origem. Sendo que cada conexão é relacionada com

inúmeras entidades do tipo *SourceTable* que descrevem a estrutura das tabelas de origens. Cada entidade *SourceTable* possui inúmeras entidades *SourceField* que descrevem os itens de dados das tabelas de origens (VALENTIN, 2006).

4.2 ESPECIFICAÇÃO DO GERENCIADOR DE METADADOS

A Figura 4.5 mostra as classes necessárias para a realização do caso de uso Definir Origens somente relativas ao gerenciamento de metadados.

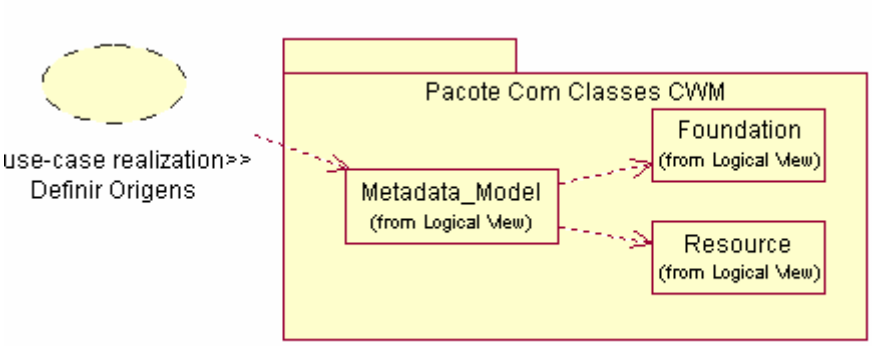

Figura 4.5: Classes necessárias para a realização do caso de uso Definir Origens

Na seqüência são relacionadas as classes de objetos definidas para o gerenciador de metadados, necessárias na realização de cada caso de uso proposto na arquitetura de Valentin (2006):

1) Definir Origens:
   - Metadata_Model;
   - Foundation;
   - Resource.
2) Definir ET:
   - Metadata_Model;
   - Foundation;
   - Resource.
3) Executar ET:
   - Metadata_Model;
   - Foundation;
   - Resource.
4) Definir Estágio:

- Metadata_Model;
- Resource.

5) Definir Carga:
   - Metadata_Model;
   - Resource.
6) Executar Carga:
   - Metadata_Model;
   - Resource;
   - Management.
7) Definir DW:
   - Metadata_Model;
   - Resource.
8) Executar OLAP:
   - Metadata_Model;
   - Analysis.
9) Executar DM:
   - Metadata_Model;
   - Analysis.
10) Armazenar Resultados:
   - Metadata_Model;
   - Analysis.
11) Visualizar Resultados:
   - Metadata_Model;
   - Analysis.

A Figura 4.6 mostra um diagrama de classes do modelo de metadados proposto, mesclando classes do padrão CWM com as etapas do processo de descoberta de conhecimento.

A seguir são descritos os métodos de cada classe de objetos do gerenciador de metadados:

a) Metadata_Model:

- Management_Metadata: responsável por verificar a existência do arquivo de metadados. Caso não exista, um arquivo XML será criado de acordo com a tarefa executada. Quando existe, o método cria ou recupera a(s)

classe(s) necessária(s) (Foundation, Resource, Analysis ou Management), de acordo com o contexto.

b) Foundation:

- Business_Information: obtém o nome do negócio;
- StageFoundation: responsável por gerar as *tags* XML necessárias.

c) Expressions:

- Transformation_Role: mantém regras de transformação utilizadas (sintaxe/semântica).

d) Parameters:

- Parameters: mantém os nomes dos parâmetros e seus respectivos tipos de uma determinada expressão (name = operando1 e type = integer);

e) Resource:

- Add_Resource: acrescenta recursos (endereço,tipo/versão SGBD) (origem/área de estágio/DW);
- Update_Resource: atualiza recursos (origem/área de estágio/DW);
- Delete_Resource: exclui recursos (origem/área de estágio/DW);
- Get_Resource: busca de recursos (origem/área de estágio/DW);
- StageResource: responsável por gerar as *tags* XML necessárias.

f) Multimensional:

- Type_Transformation: mantém o tipo expressão que gerou tal atributo da tabela multidimensional, ou seja, se o dado presente no modelo multidimensional, por exemplo, se trata de um resumo de vários atributos ou mesmo o próprio atributo. A ausência de informação significa que o atributo não possui procedência.

g) Tables:

- Add_TKA: adiciona nome da tabela (*Table*), chaves (*Keys*) e atributos (*Attributes*) (origem/área de estágio/DW);
- Update_TKA: atualiza nome de tabela, chaves e atributos (origem/área de estágio/DW);
- Delete_TKA: exclui tabela com seus repectivos atributos (origem/área de estágio/DW);
- Get_TKA: busca tabela e atributos para o ambiente (origem/área de estágio/DW).

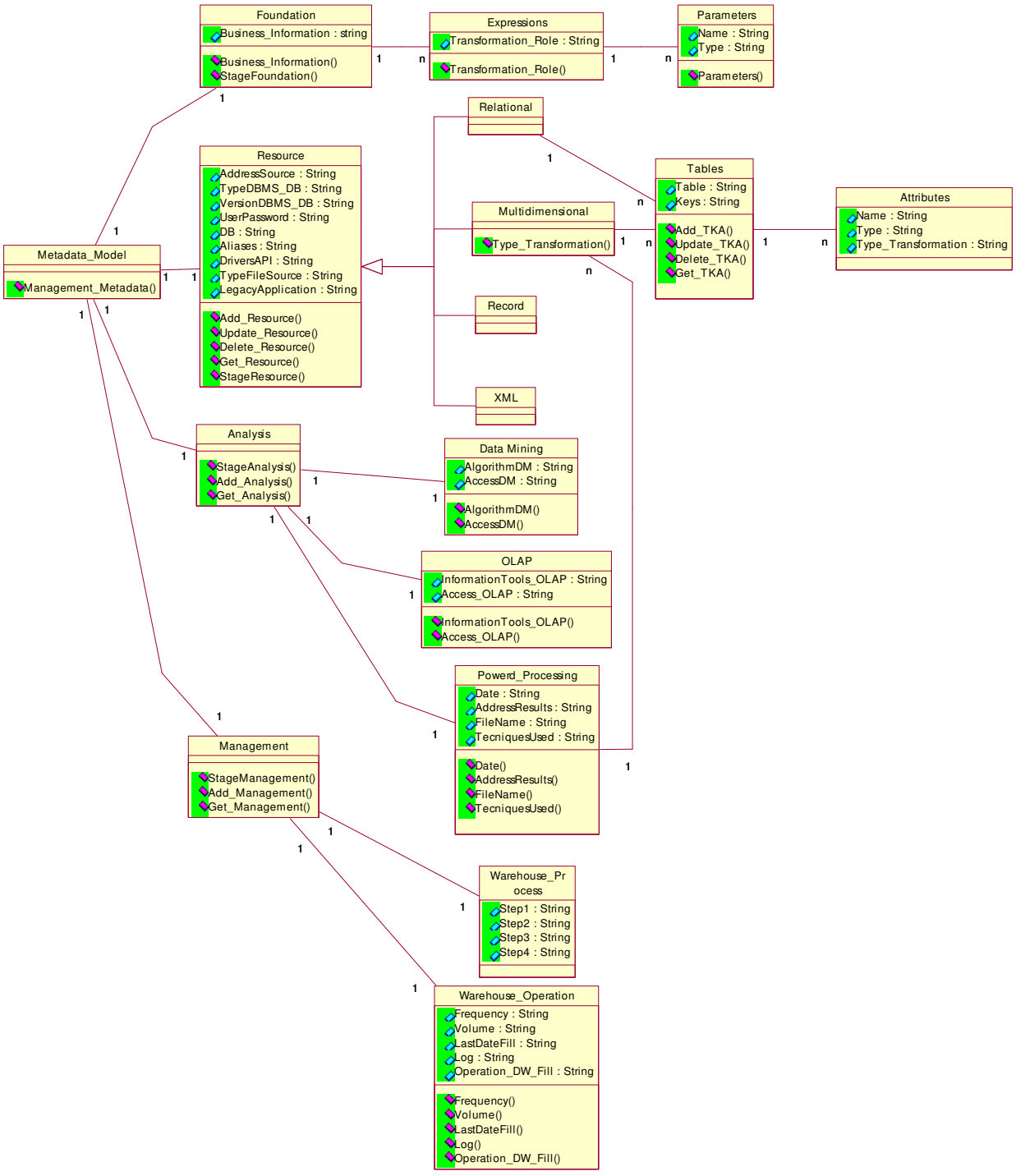

Figura 4.6: Diagrama de Classes do Modelo de Metadados

h) Analysis:

- StageAnalysis: responsável por gerar as *tags* XML necessárias;
- Add_Analysis: responsável por adicionar informações relativas à análise de dados em um ambiente de DW;

- Get_Analysis: responsável por recuperar informações relativas à análise de dados em um ambiente de DW.

i) Data Mining:

- AlgorithmDM: exibe informação sobre o algoritmo de mineração de dados utilizado;
- AccessDM: mostra a localização do algoritmo de mineração de dados utilizado.

j) OLAP:

- Information_Tools_OLAP: exibe informação da ferramenta OLAP utilizada;
- Access_OLAP: mostra a localização da ferramenta OLAP utilizada.

k) Powerd_Processing:

- Date: busca a data de povoamento do DW;
- AddressResults: mostra o endereço eletrônico do arquivo de resultado;
- FileName: mostra o nome do arquivo de resultado;
- TecniquesUsed: exibe a técnica usada para encontrar tal resultado.

l) Management:

- StageManagement: responsável por gerar as *tags* XML necessárias;
- Add_Management: responsável por adicionar informações relativas ao gerenciamento de dados em um ambiente de DW;
- Get_Management: responsável por recuperar informações relativas ao gerenciamento de dados em um ambiente de DW.

m) Warehouse_Operation:

- Frequency: define a freqüência de carga de dados na área de estágio/DW;
- Volume: define o volume dos dados da área de estágio/DW;
- LastDateFill: registra a última data de povoamento da área de estágio/DW;
- Log: busca o login do responsável pela operação de povoamento da área de estágio/DW;
- Operation_DW_Fill: responsável por armazenar qual operação foi realizada na carga de dados da tabela fato do DW, como por exemplo, a sumarização com base nas vendas por semana.

Os atributos persistidos nas classes concretas estão listados a seguir:

1) **Relational:**
   - AddressSource;
   - TypeDBMS_DB;
   - VersionDBMS_DB;
   - UserPassword;
   - DB;
   - Aliases;
   - DriversAPI;
   - TypeFileSource;
   - LegacyApplication.
2) **Multidimensional:**
   - AddressSource;
   - TypeDBMS_DB;
   - VersionDBMS_DB;
   - UserPassword;
   - DB;
   - Aliases;
   - DriversAPI;
   - TypeFileSource.
3) **Record/XML:**
   - AddressSource;
   - UserPassword;
   - Aliases;
   - DriversAPI;
   - TypeFileSource.

Nas Figuras 4.7 a 4.17 são mostrados os diagramas de seqüência de cada caso de uso, retratando a seqüência de execução das ações. A Figura 4.7 retrata a seqüência de execução dos métodos do caso de uso "Definir Origens". Além de operações básicas como: adição, atualização ou exclusão de recursos (métodos: Add_Resource, Add_TKA, Get_Resource, Get_TKA, Update_Resource, Update_TKA, etc...), são definidas, neste caso de uso, informações sobre a área de negócio representada pela base de dados de origem (método Business_Information).

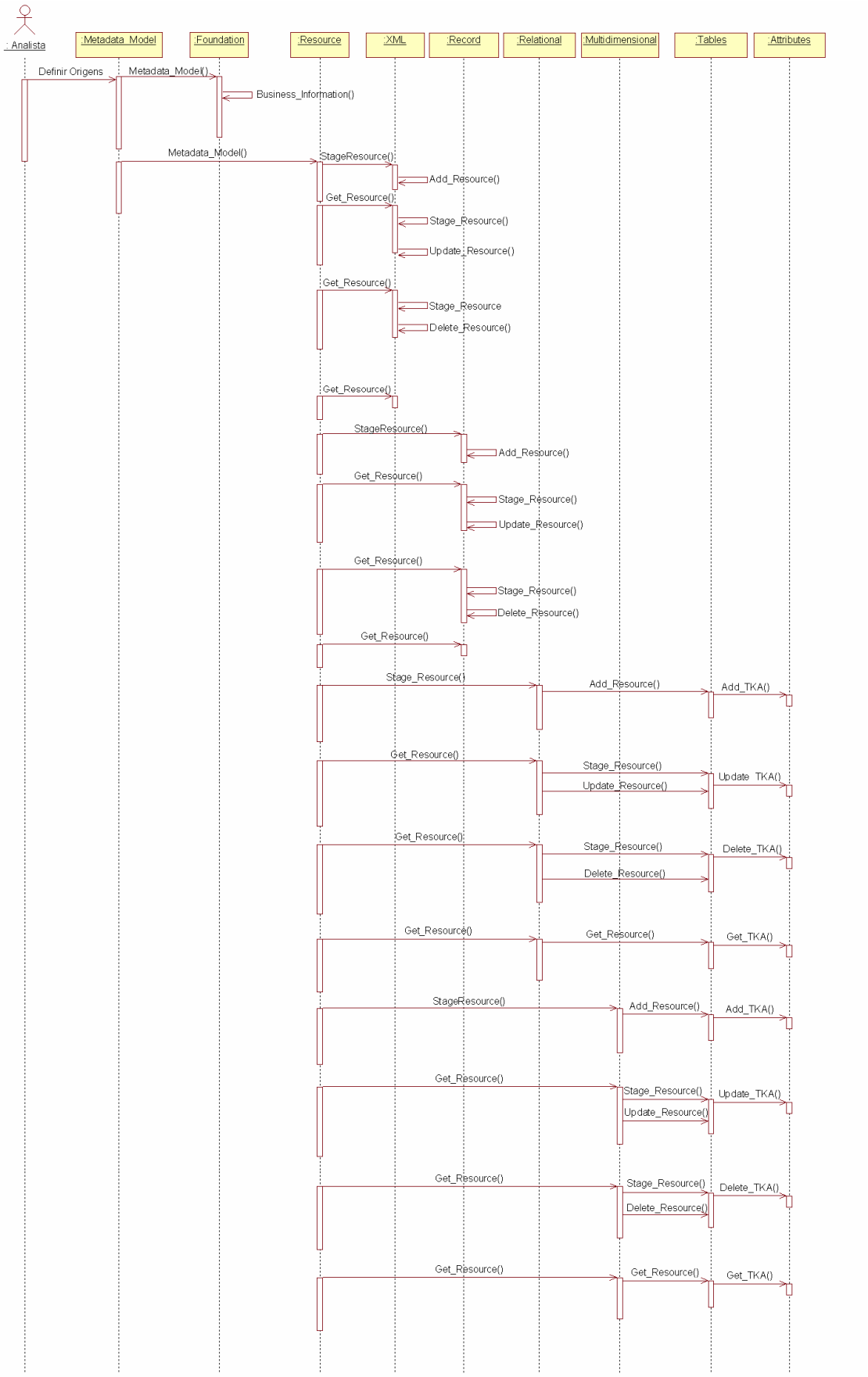

Figura 4.7: Diagrama de Seqüência do Caso de Uso Definir Origens

A Figura 4.8 relata a execução do caso de uso “Definir Extração e Transformação”. Para isto, é necessário disponibilizar as operações de transformação de dados e selecionar quais itens de dados sofrerão alterações. Após a execução destas operações, as transformações são registradas.

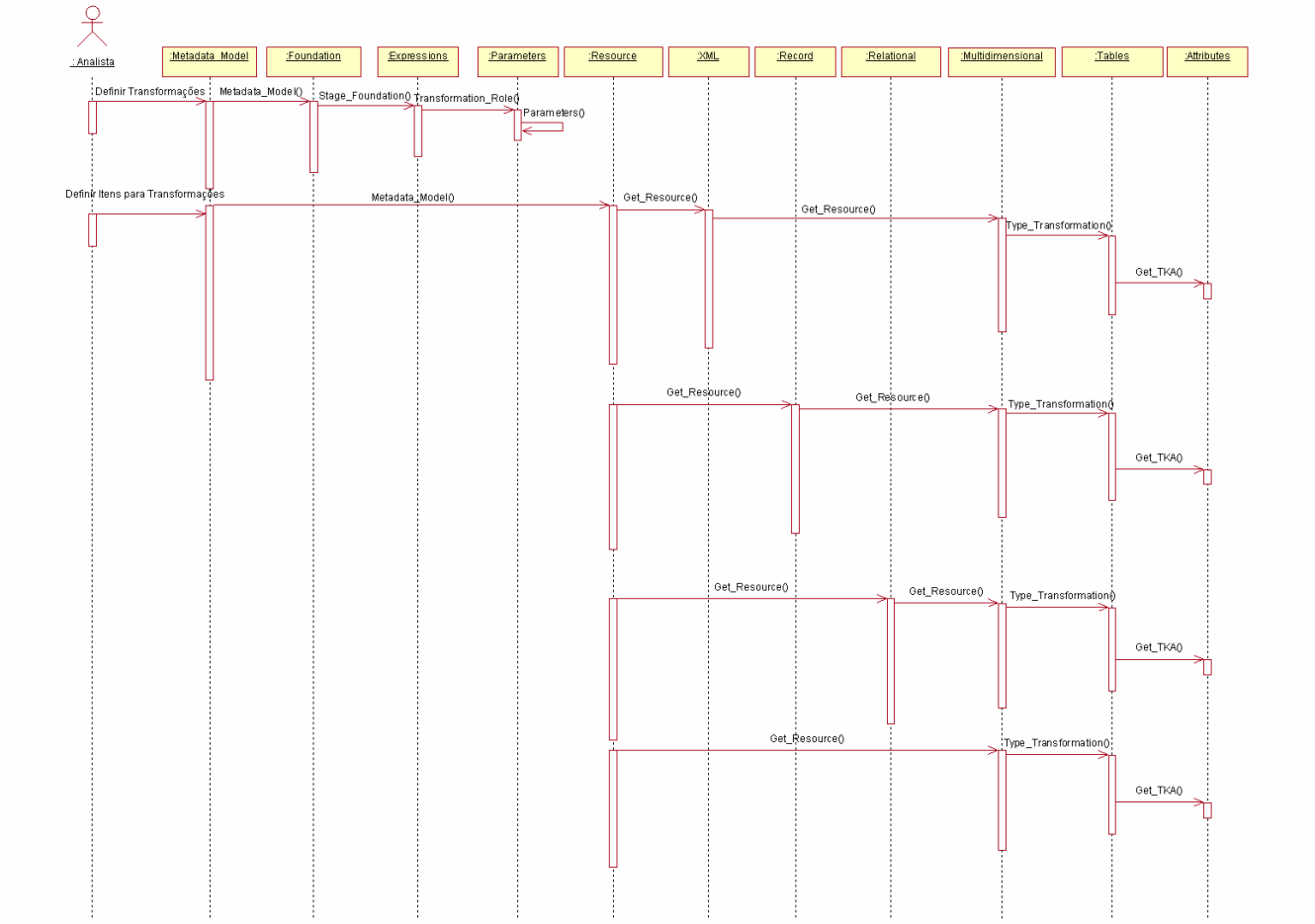

Figura 4.8: Diagrama de Seqüência do Caso de Uso Definir ET

A Figura 4.9 apresenta a execução do caso de uso “Executar Extração e Transformação”. Neste caso de uso, as regras de transformação selecionadas são executadas de acordo com os parâmetros definidos anteriormente.

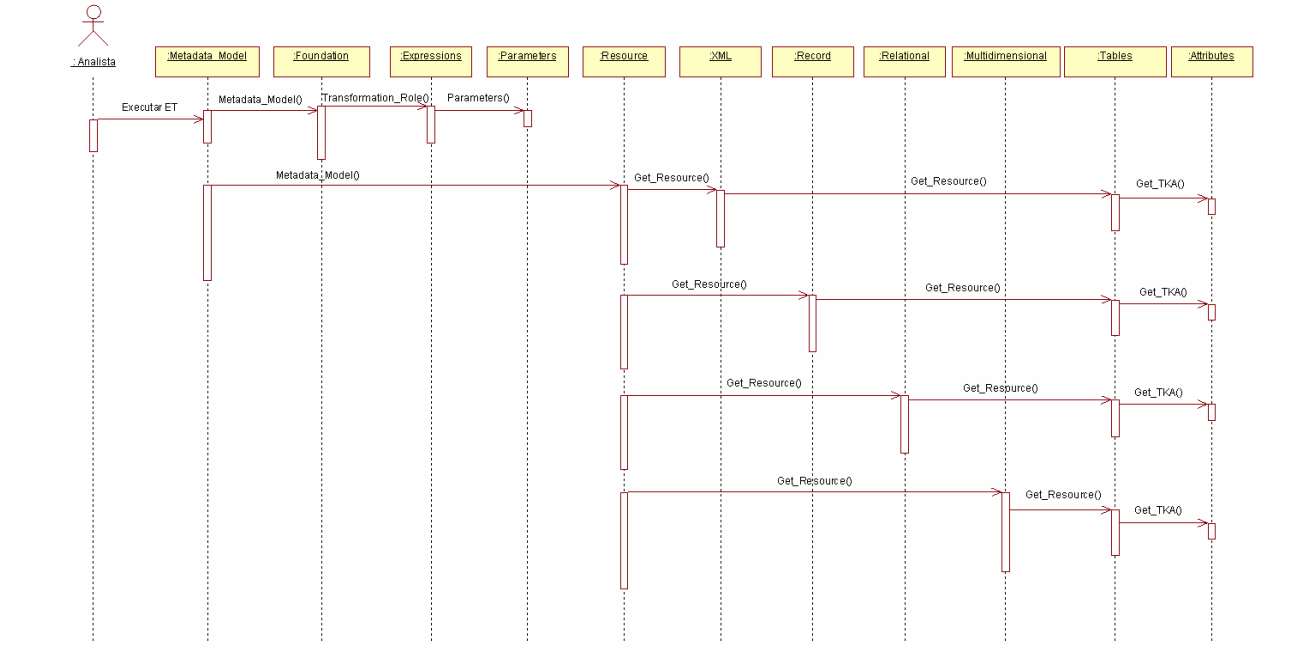

Figura 4.9: Diagrama de Seqüência do Caso de Uso Executar ET

A Figura 4.10 mostra a seqüência de execução de métodos do caso de uso “Definir Estágio”. Após extração e eventuais transformações dos dados de origem, são solicitadas informações sobre a localização desses dados da área de estágio.

A Figura 4.11 se refere à execução do caso de uso “Definir Carga”, onde são solicitadas informações como, por exemplo, agenda da carga (freqüência da realização da carga de dados no DW). Já a Figura 4.12 mostra a execução da carga.

A Figura 4.13 exibe a execução do caso de uso “Definir DW”. Nessa etapa, são solicitadas informações sobre a localização dos dados que conceberão o DW.

O diagrama de seqüência mostrado na Figura 4.14 se refere ao estudo de caso “Executar OLAP”. Além dos eventuais dados utilizados na execução da ferramenta OLAP, é necessário conhecer informações sobre a ferramenta.

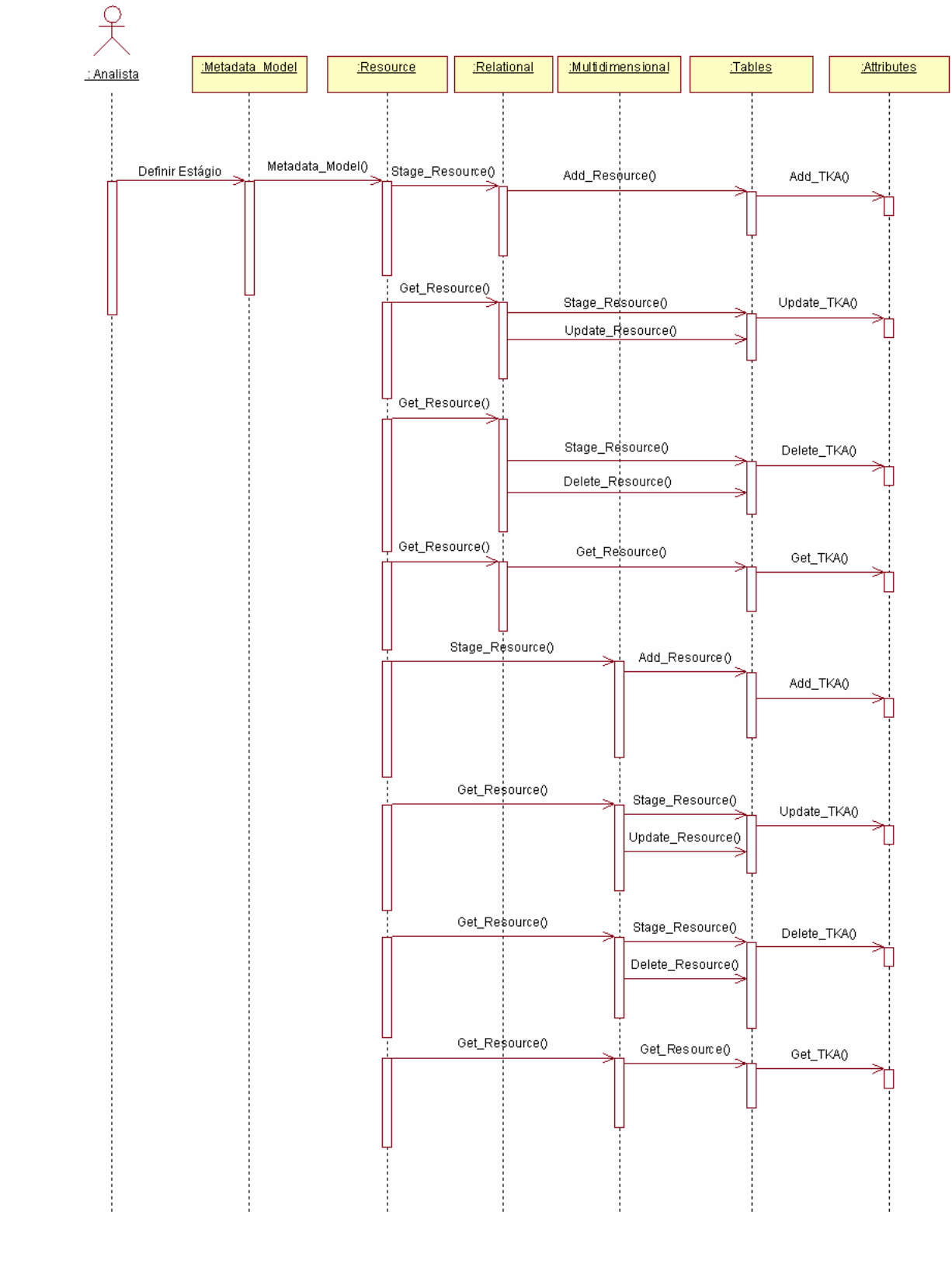

Figura 4.10: Diagrama de Seqüência do Caso de Uso Definir Estágio

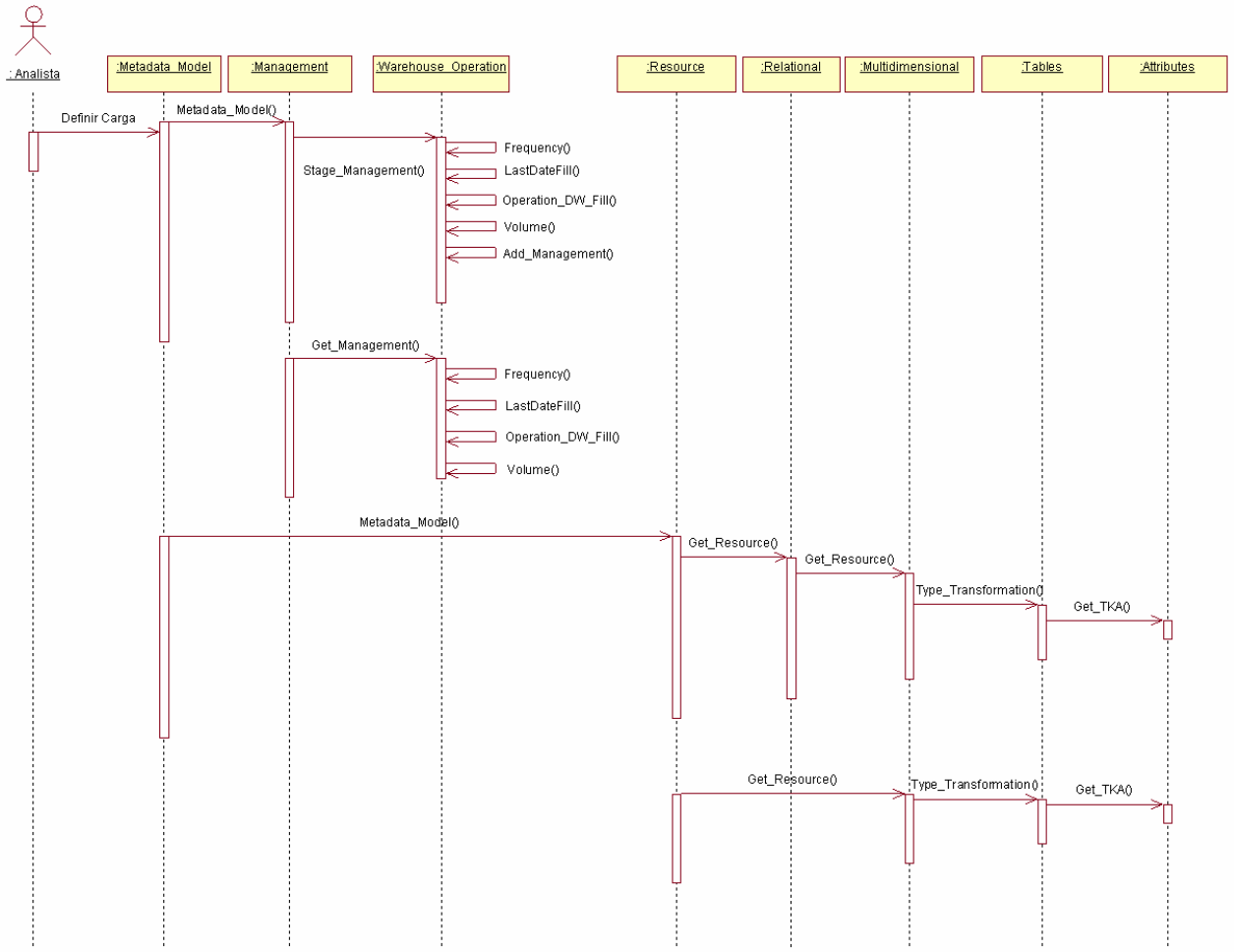

Figura 4.11: Diagrama de Seqüência do Caso de Definir Carga

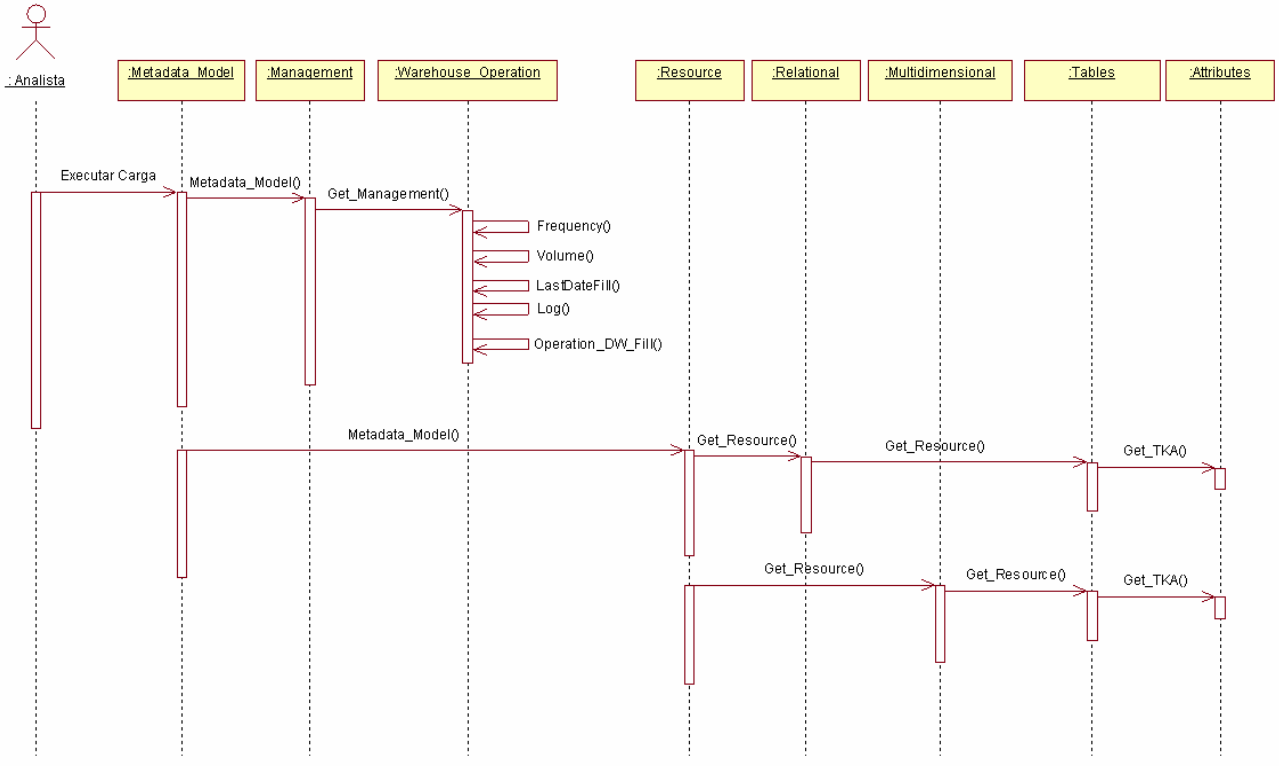

Figura 4.12: Diagrama de Seqüência do Caso de Uso Executar Carga

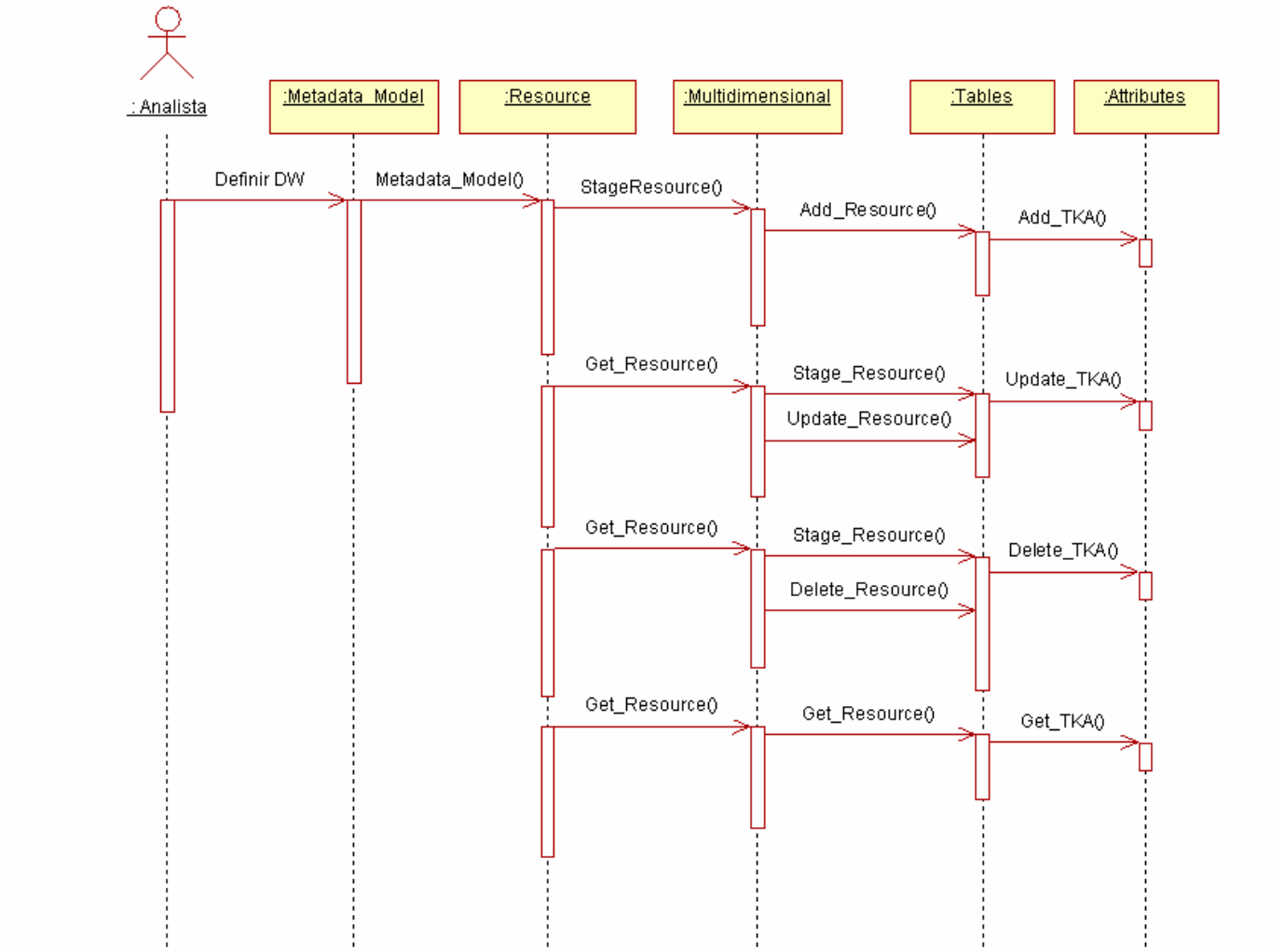

Figura 4.13: Diagrama de Seqüência do Caso de Uso Definir DW

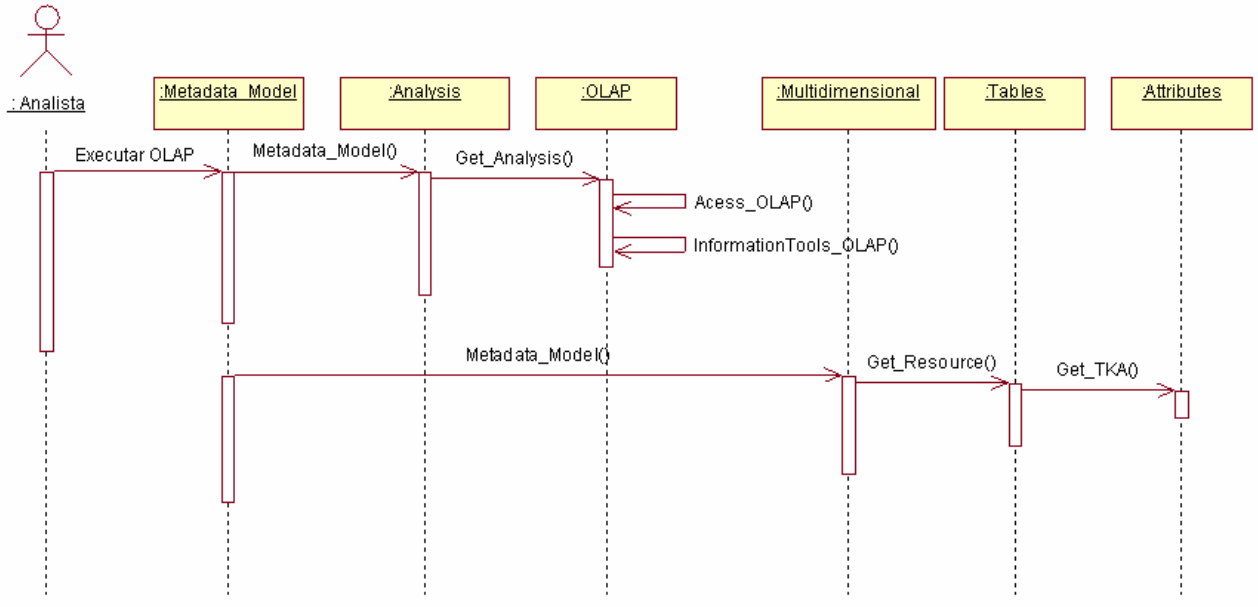

Figura 4.14: Diagrama de Seqüência do Caso de Uso Executar OLAP

A Figura 4.15 mostra a execução do estudo de caso “Executar *Data Mining*”. Além dos eventuais dados utilizados na execução de um algoritmo de mineração de dados, é necessário conhecer informações sobre esses algoritmos.

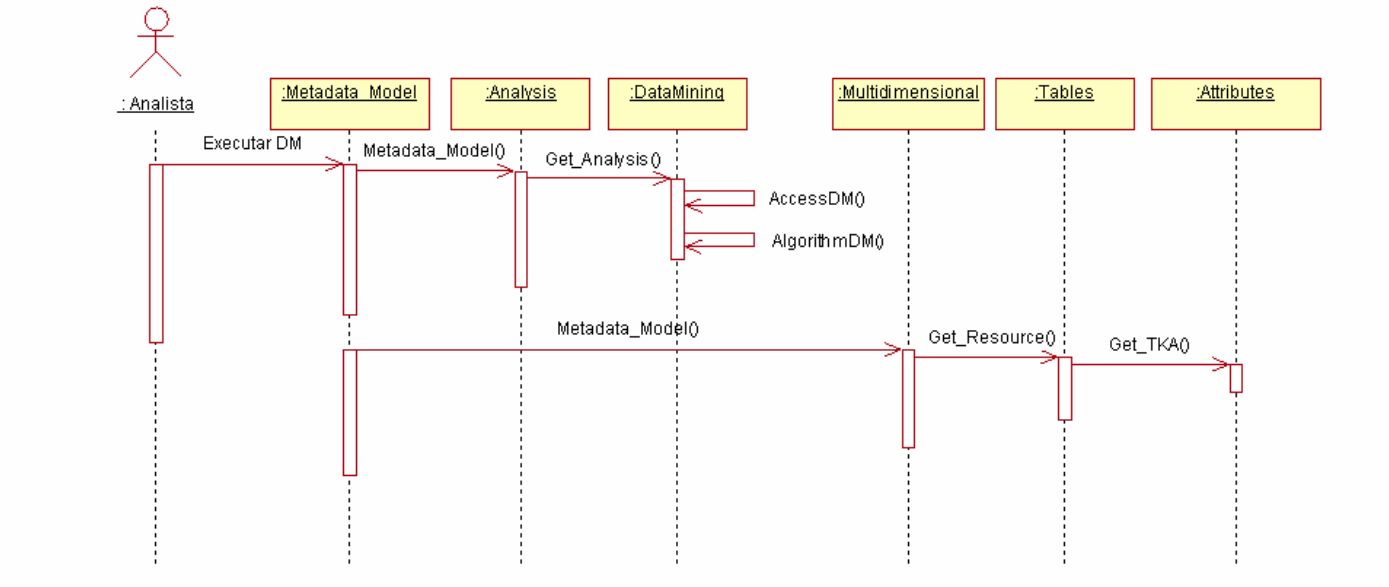

Figura 4.15: Diagrama de Seqüência do Caso de Uso Executar DM

Após a aplicação de alguma técnica (OLAP ou algoritmo de DM) sobre os dados, os resultados obtidos devem ser armazenados Essa operação é visualizada na Figura 4.16. Já na Figura 4.17 é mostrada a execução do estudo de caso “Visualizar Resultados”. Os resultados armazenados anteriormente são disponibilizados para consulta.

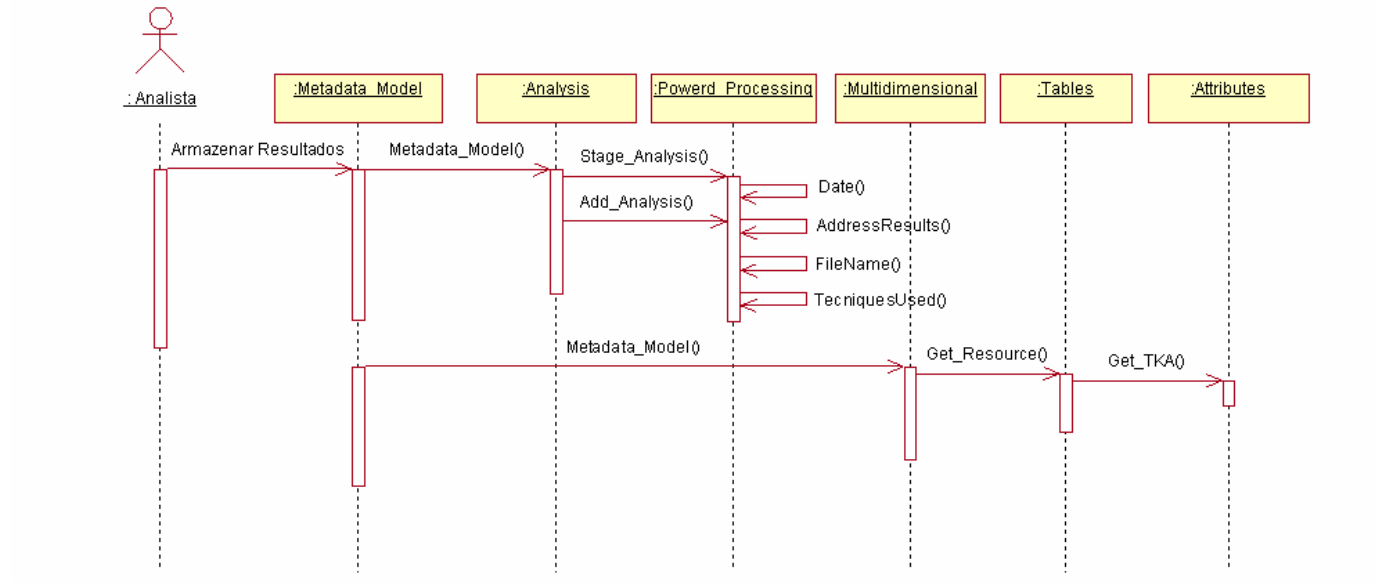

Figura 4.16: Diagrama de Seqüência do Caso de Uso Armazenar Resultados

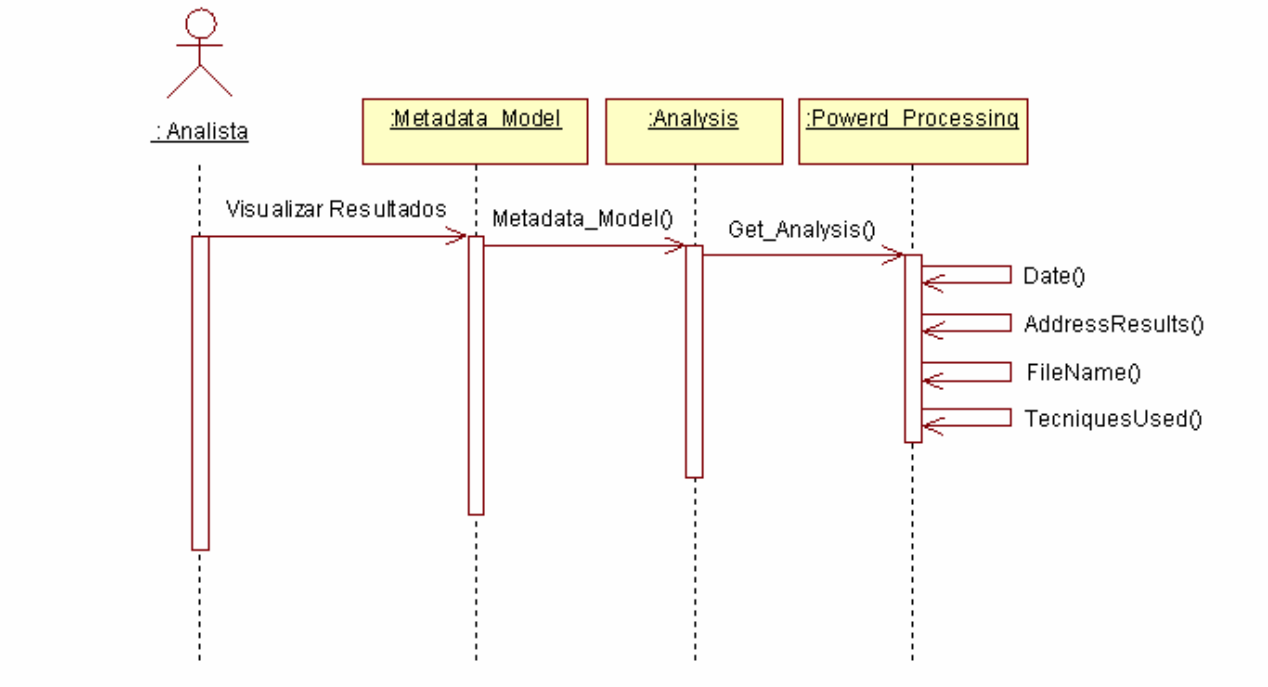

Figura 4.17: Diagrama de Seqüência do Caso de Uso Visualizar Resultados

Na Figura 4.18 são representadas as dependências entre os principais pacotes da arquitetura de referência de sistemas KDD de Valentin (2006). É visível que o pacote "Gerenciador de Visualização" depende do pacote "Gerenciador de Processos" que depende do pacote "Gerenciador de Serviços" que, por sua vez, também necessita de auxílio do pacote de menor nível (Gerenciador de Apoio (metadados)).

Na Figura 4.19, é mostrado o diagrama de pacotes CWM com suas respectivas classes. Percebe-se uma forte dependência dos pacotes Resource, Analysis e Management em relação ao pacote Foundation. Isto se deve ao fato do pacote Foundation ser base para os demais pacotes em um ambiente de descoberta de conhecimento. Uma particularidade é a dependência direta do pacote Analysis para o Resource, uma vez que os dados analisados são fortemente dependentes das fontes de dados existentes em um ambiente de descoberta de conhecimento. Segue relação das classes de cada pacote:

a) Foundation:

- Expressions;
- Parameters.

b) Resource:

- Relational;
- Multimensional;
- Record;

- XML;
- Tables:
- Attributes.

c) Analysis:

- Data Mining;
- OLAP;
- Powerd_Processing.

d) Management:

- Warehouse_Process;
- Warehouse_Operation.

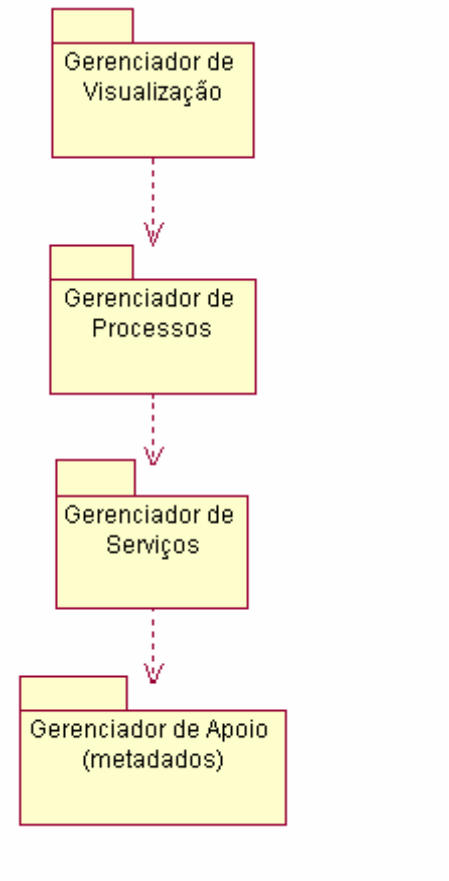

Figura 4.18: Dependência entre os Principais Pacotes Adaptado da Arquitetura de Referência de Valentin (2006)

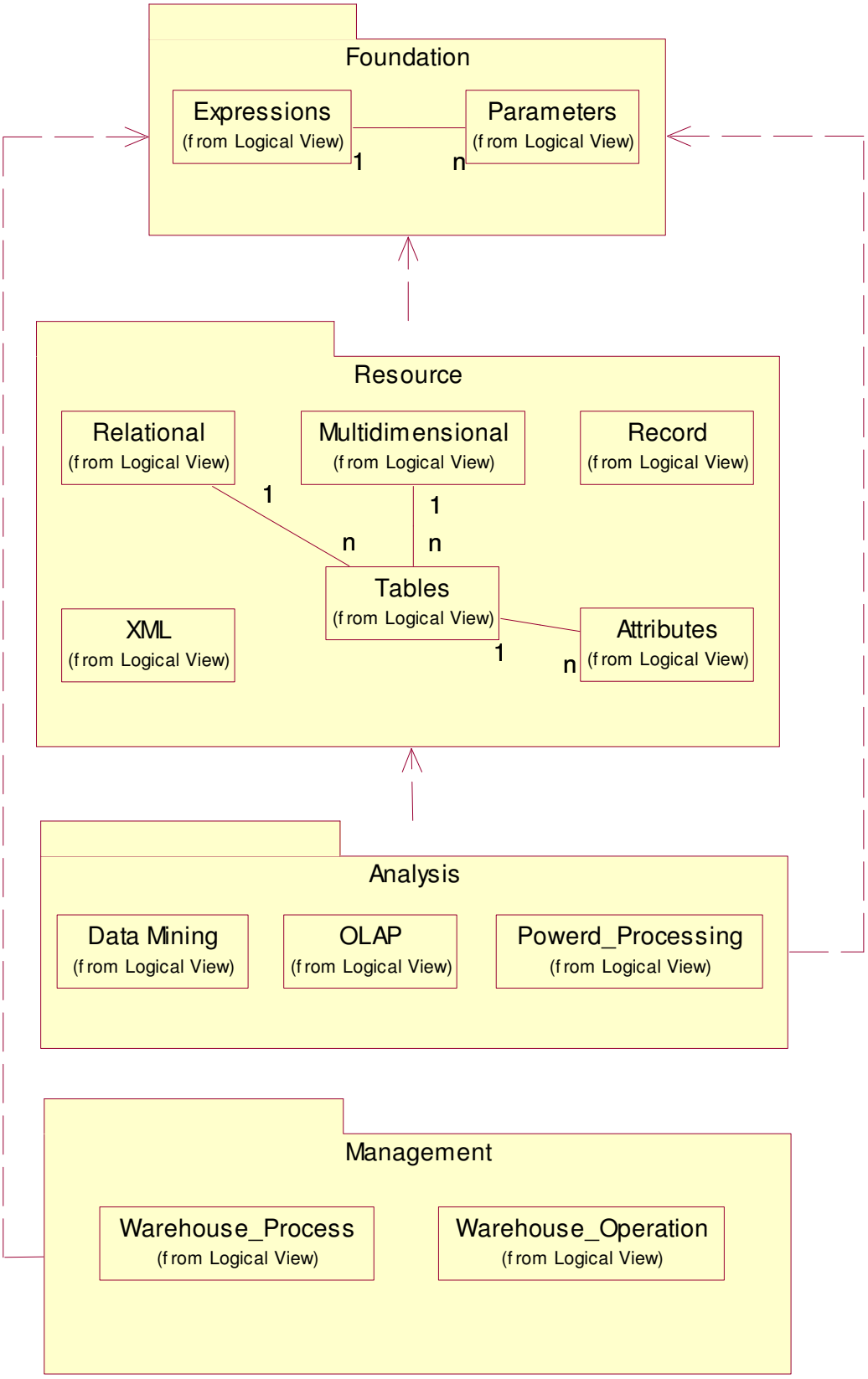

Figura 4.19: Diagrama dos Pacotes CWM com Suas Respectivas Classes Associadas

O gerenciador de metadados é representado pelo componente “Metadata_Model”. Os serviços relativos a metadados são solicitados pelo “ServiceManager” diretamente ao “Metadata_Model”. Portanto, o componente “ServiceManager” é o responsável pela comunicação do sistema KDD com o componente gerenciador de metadados.

A Figura 4.20 ilustra um diagrama de implementação com os componentes e classes criados e também os próprios da linguagem Java que foram utilizados no desenvolvimento do protótipo, descrito no próximo capítulo, e do componente gerenciador de metadados.

4.3 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a especificação do componente de gerenciamento de metadados, tendo como base a arquitetura de referência proposta por Valentin (2006). Essa arquitetura é composta por quatro camadas, sendo uma delas a camada de apoio, onde se encontra o gerenciador de metadados. Na especificação apresentada, são mostrados os diagramas em UML construídos e descritas as classes de objetos e seus métodos.

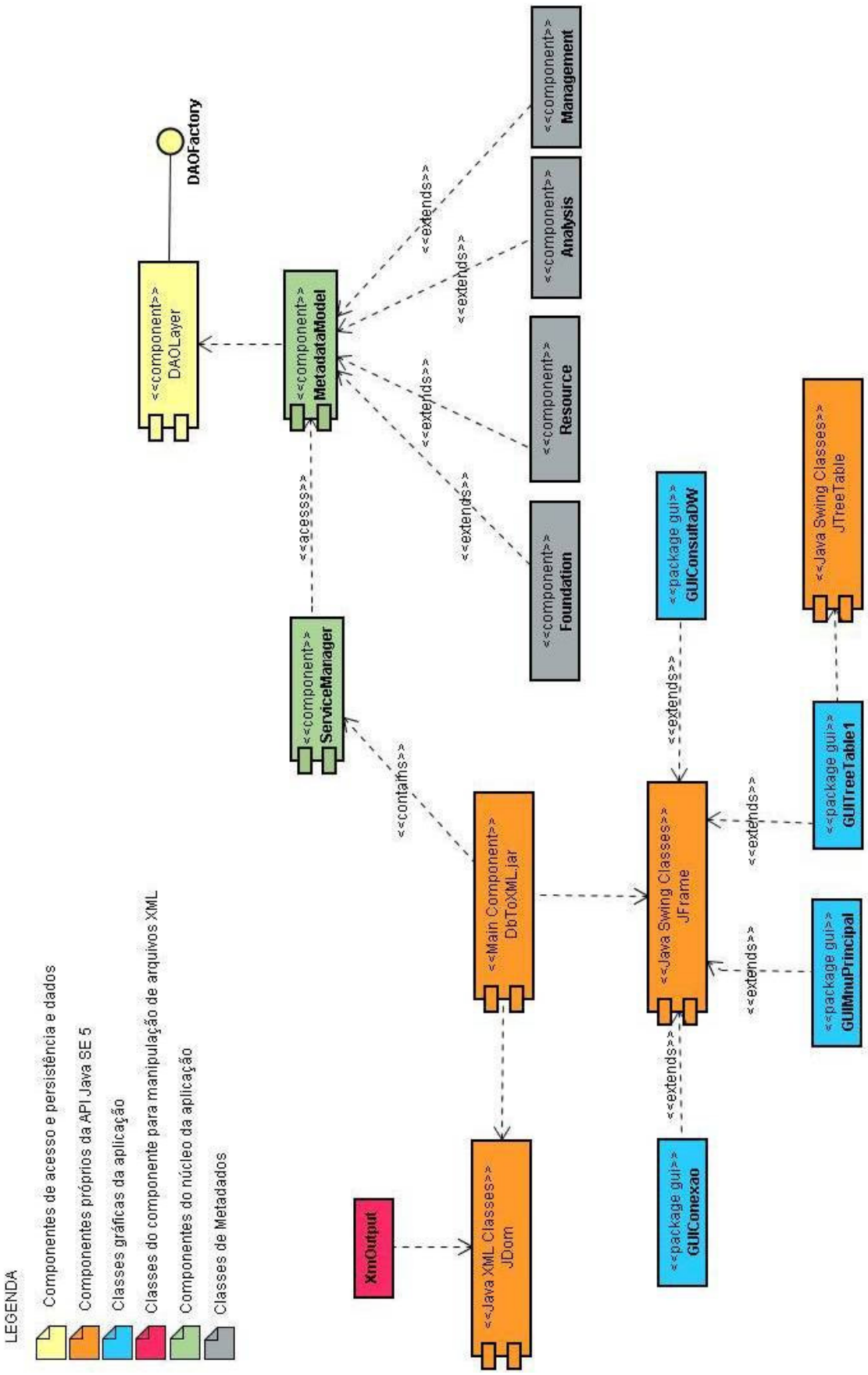

Figura 4.20: Diagrama de Implementação dos Componentes/Classes Utilizados na Concepção do Protótipo e Componente Gerenciador de Metadados

5 ESTUDO DE CASO

Neste capítulo é demonstrado o estudo de caso realizado utilizando o protótipo desenvolvido em Java para gerenciamento de metadados. Neste protótipo foi implementada uma interface simplificada considerando que em um sistema real de KDD, o gerenciador de metadados será um componente de software a ser utilizado como suporte ao processo KDD, principalmente para a construção e o acesso a um DW. Portanto, o projeto e a implementação da interface do sistema KDD não fazem parte do escopo deste trabalho.

5.1 FONTES DE DADOS DE ORIGEM

Neste estudo de caso, foi utilizado um banco de dados com quatro tabelas, simulando os dados de origem de um sistema legado. A Tabela 5.1 mostra as tabelas de origem com seus respectivos atributos.

Tabela 5.1: Relação de Tabelas de Sistema Legado (Origem)

Tabelas	Atributos
Produto	Cod_produto Descricao Vl_unitario
Cliente	Cod_cliente Nome
Vendas	Data_venda Cod_venda Cod_produto Cod_cliente Quantidade
Fornecedor	Cod_fornecedor Nome

5.2 DEMONSTRAÇÃO DO USO DO PROTÓTIPO

Nesta seção é demonstrado o uso do componente gerenciador de metadados por meio do protótipo implementado. Neste componente, os metadados são representados em XML.

A Figura 5.1 ilustra a tela principal do protótipo, sendo um menu que contempla as etapas básicas de um sistema de descoberta de conhecimento.

Como o objetivo é demonstrar como o os metadados são gerenciados pelo componente, a seguir são apresentados as etapas realizadas neste gerenciamento:

1) Definir origens:

A Figura 5.2 mostra a interface "DEFINIR ORIGENS" onde o usuário poderá se conectar a um banco de dados de origem. Após a conexão ser estabelecida, as tabelas com os respectivos atributos são listadas no vídeo. Nesse momento é possível escolher quais tabelas e quais atributos serão definidos como dados de origem do sistema KDD, essa ação pode ser visualizada na Figura 5.3.

Ao clicar em "DEFINIR ORIGEM", são gerados dados para o arquivo XML (metadados). Esses metadados gerados informarão posteriormente ao protótipo os dados que o usuário poderá extrair para a área de estágio.

Figura 5.1: Tela do Menu Principal do Protótipo

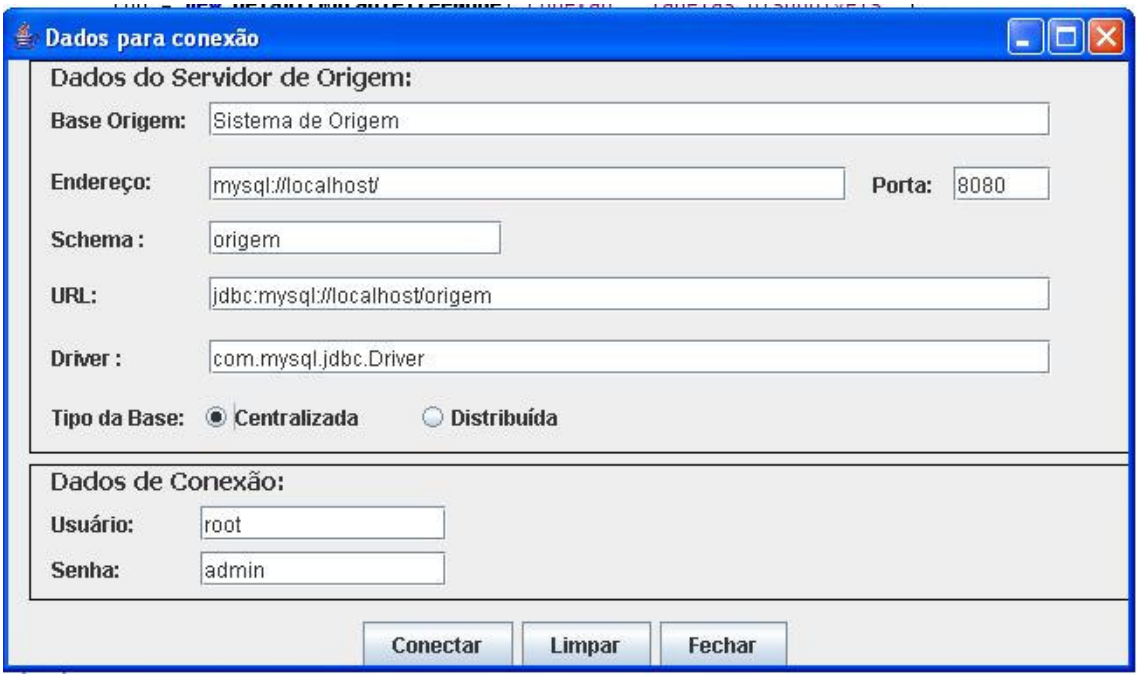

Figura 5.2: Tela de Definições de Origens

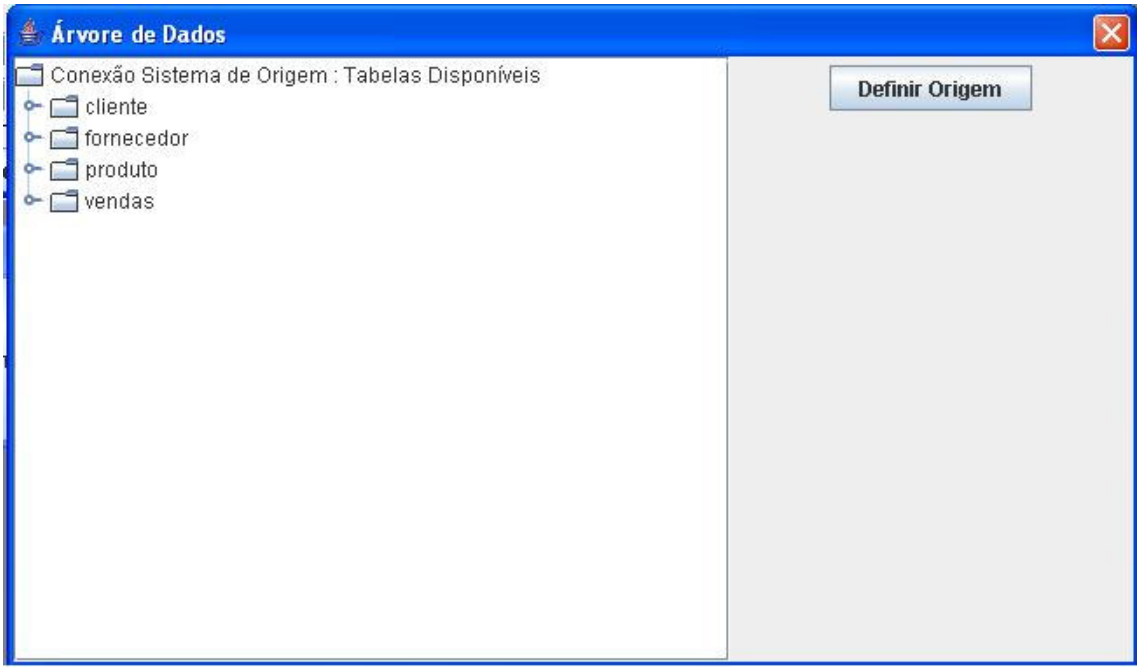

Figura 5.3: Tela de Definições de Tabelas e Atributos de Origem

Após a entrada dos dados solicitados, conforme ilustrado nas Figuras 5.2 e 5.3, o código XML a seguir é gerado, para representar as informações sobre as tabelas e atributos escolhidos que, neste caso, foram as tabelas Cliente, Produto e Vendas e todos os seus atributos.

```
. . .
<Resource>
 <search>
  <relational_Origin>
   <Address_Source_Origin>jdbc:mysql://localhost/origem</Address_Source_Origin>
        <Type_DBMS_DB_Origin>Centralizado</Type_DBMS_DB_Origin>
        <Version_DBMS_DB_Origin></Version_DBMS_DB_Origin>
        <User_Password_Origin>rootladmin</User_Password_Origin>
        <DB_Origin>origem</DB_Origin>
        <Aliases_Origin></Aliases_Origin>
        <Driver_API_Origin>com.mysql.jdbc.Driver</Driver_API_Origin>
        <Type_File_Source_Origin></Type_File_Source_Origin>
        <Legacy_Application>Sistema Financeiro</Legacy_Application>
        <Tables_Attributes_Keys_Origin>
         <Table>
           <Table_Name>cliente</Table_Name>
           <Attributes>
             <Name>cod_cliente</Name>
             <Type>int</Type>
             <Name>nome</Name>
             <Type>varchar</Type>
           </Attributes>
           <Keys>
             <Name>cod_cliente</Name>
           </Keys>
         </Table>
         <Table>
           <Table_Name>produto</Table_Name>
           <Attributes>
                <Name>cod_produto</Name>
                <Type>int</Type>
                <Name>descricao</Name>
                <Type>varchar</Type>
                <Name>vl_unitario</Name>
                <Type>decimal</Type>
          </Attributes>
          <Keys>
            <Name>cod_produto</Name>
          </Keys>
        </Table>
        <Table>
          <Table_Name>vendas</Table_Name>
          <Attributes>
                <Name>cod_vendas</Name>
                <Type>int</Type>
                <Name>cod_produto</Name>
                <Type>int</Type>
                <Name>cod_cliente</Name>
                <Type>int</Type>
                <Name>quantidade</Name>
                <Type>int</Type>
                <Name>data_venda</Name>
                <Type>datetime</Type>
          </Attributes>
          <Keys>
            <Name>cod_vendas</Name>
          </Keys>
        </Table>
   </Tables_Attributes_Keys_Origin>
  </relational_Origin>
```

```
  </search>
</Resource>
. . .
```

2) Definir Extração e Transformação:

A Figura 5.4 “DEFINIR EXTRAÇÃO E TRANSFORMAÇÃO” ilustra os dados de origem definidos. A recuperação desses dados se dá via XML (arquivo de metadados). Nesta figura estão representadas duas opções que possibilitam transformações nos itens de dados das tabelas de origem.

Antes de criar um banco de dados na área de estágio para armazenar os dados de origem extraídos e transformados, o usuário deverá informar dados sobre a sua localização, esta tarefa é visível na Figura 5.5.

Ao clicar no botão “DEFINIR ÁREA DE ESTÁGIO”, são gerados os metadados dos dados povoados na área de estágio, sendo informações relativas à localização do banco, tabelas, atributos e eventuais transformações solicitadas nos dados. Nesse momento é gerado um banco de dados de estágio, onde os dados são registrados.

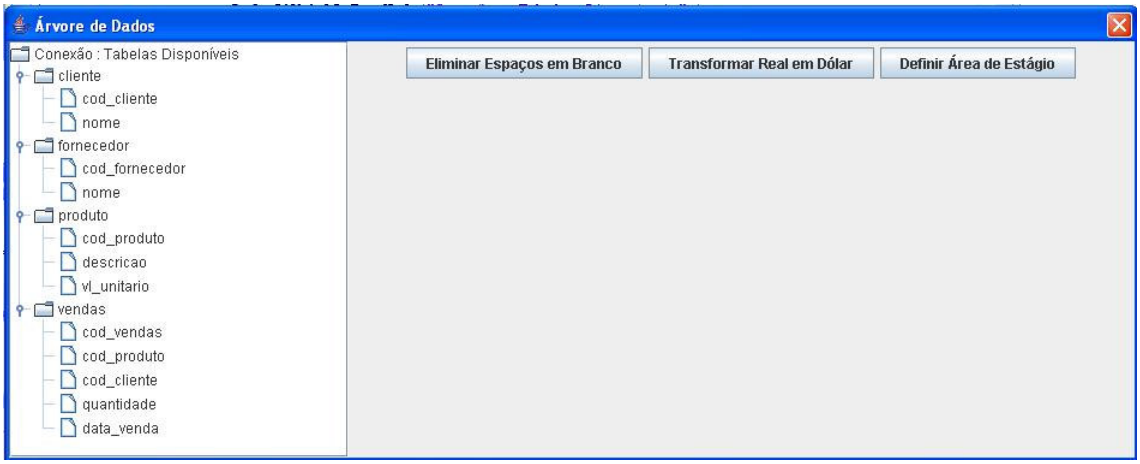

Figura 5.4: Tela de Definições de Transformações e Extração de Dados

Após a definição das transformações e do local físico do banco de dados da Área de Estágio, ilustrados nas Figuras 5.4 e 5.5, o arquivo XML de metadados é complementado com esses novos dados, conforme trecho deste arquivo apresentado na seqüência. Observe que o atributo “nome” da tabela “cliente” do banco de dados “origem” sofreu uma transformação de exclusão de espaços em branco.

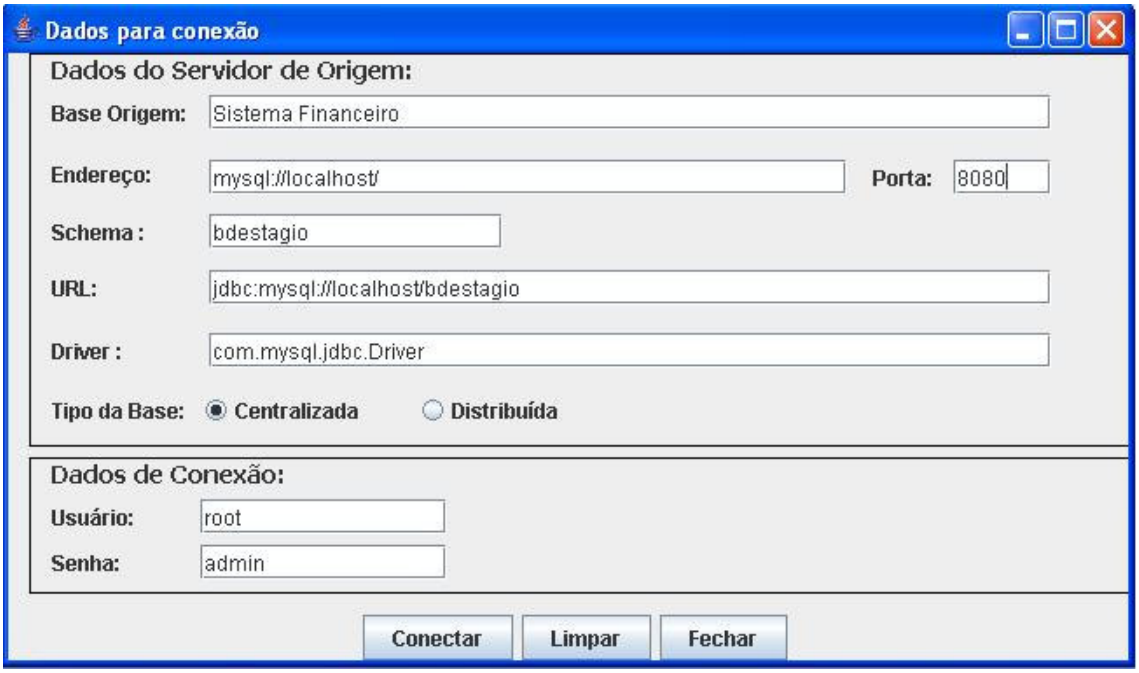

Figura 5.5: Tela de Definições de Localização do Banco de Dados de Estágio

```
. . .
<Pre_Processing>
 <load>
  <Stage_Data>
   <relational_Stage>
    <Address_Source_Stage>mysql://localhost/:</Address_Source_Stage>
    <Type_DBMS_DB_Stage>Centralizado</Type_DBMS_DB_Stage>
    <Version_DBMS_DB_Stage></Version_DBMS_DB_Stage>
    <User_Password_Stage>rootladmin</User_Password_Stage>
    <DB_Stage>bdestagio</DB_Stage>
    <Aliases_Stage></Aliases_Stage>
    <Driver_API_Stage>com.mysql.jdbc.Driver</Driver_API_Stage>
    <Type_File_Source_Stage></Type_File_Source_Stage>
    <Tables_Attributes_Keys_Stage>
     <Table_Stage>
      <Table_Name_Stage>stage_cliente</Table_Name_Stage>
      <Attributes_Stage>
       <Name>cod_cliente</Name>
       <Type>int</Type>
       <Type_Transformation></Type_Transformation>
      <Name>nome</Name>
      <Type>varchar</Type>
      <Type_Transformation>trim(origem.cliente.nome)</Type_Transformation>
     </Attributes_Stage>
     <Keys_Stage>
      <Name>cod_cliente</Name>
     </Keys_Stage>
    </Table_Stage>
    <Table_Stage>
     <Table_Name_Stage>stage_produto</Table_Name_Stage>
     <Attributes_Stage>
      <Name>cod_produto</Name>
      <Type>int</Type>
      <Type_Transformation></Type_Transformation>
```

```
            <Name>descricao</Name>
            <Type>varchar</Type>
            <Type_Transformation></Type_Transformation>
            <Name>vl_unitario</Name>
            <Type>decimal</Type>
            <Type_Transformation></Type_Transformation>
          </Attributes_Stage>
          <Keys_Stage>
            <Name>cod_produto</Name>
          </Keys_Stage>
        </Table_Stage>
        <Table_Stage>
          <Table_Name_Stage>stage_vendas</Table_Name_Stage>
          <Attributes_Stage>
            <Name>cod_vendas</Name>
            <Type>int</Type>
            <Type_Transformation></Type_Transformation>
            <Name>cod_produto</Name>
            <Type>int</Type>
            <Type_Transformation></Type_Transformation>
            <Name>cod_cliente</Name>
            <Type>int</Type>
            <Type_Transformation></Type_Transformation>
            <Name>quantidade</Name>
            <Type>int</Type>
            <Type_Transformation></Type_Transformation>
            <Name>data_venda</Name>
            <Type>datetime</Type>
            <Type_Transformation></Type_Transformation>
          </Attributes_Stage>
          <Keys_Stage>
            <Name>cod_vendas</Name>
          </Keys_Stage>
        </Table_Stage>
      </Tables_Attributes_Keys_Stage>
    </relational_Stage>
  </Stage_Data>
 </load>
</Pre_Processing>
. . .
```

O trecho XML abaixo se refere à informação do negócio e as possíveis transformações de dados oferecidas pelo protótipo.

```
. . .
<Foundation>
 <Business_Information>Sistema Financeiro</Business_Information>
 <Expressions>
  <Expression1>
   <Transformation_Role>trim(campo)</Transformation_Role>
   <Parameters>
    <Name>nome</Name>
    <Type>String</Type>
   </Parameters>
  </Expression1>
  <Expression2>
   <Transformation_Role>converteRealParaDolar(valor)</Transformation_Role>
```

```
    <Parameters>
      <Name>valor</Name>
      <Type>Numeric</Type>
    </Parameters>
   </Expression2>
  </Expressions>
</Foundation>
. . .
```

3) Definir Dimensões:

A Figura 5.6 “DEFINIR DW - DIMENSÕES” mostra as tabelas da área de estágio recuperadas do arquivo XML criado até o momento. Cabe ao usuário definir quais das tabelas exibidas serão as fontes de dados das tabelas dimensão na modelagem multidimensional, essas tabelas serão desconsideradas na tela de definição da tabela fato.

Ao selecionar uma tabela qualquer, seus atributos serão visualizados, permitindo a sua inclusão no DW.

Antes de criar um banco de dados do DW, o usuário deverá informar dados sobre a sua localização, conforme mostra a Figura 5.7. Ao clicar o botão “DEFINIR DW - DIMENSÕES” as tabelas de dimensão são criadas, sendo as chaves primárias geradas automaticamente pelo protótipo. Na literatura a respeito de DW é evidenciado que a chave da tabela de dimensão deve ser simples e não deve estar relacionada com a chave primária da tabela do sistema legado. Essa técnica é usada, por exemplo, para garantir a existência de um determinado produto ao longo do tempo, independente de eventuais alterações ocorridas nos sistemas legados.

Após criação da tabela de dimensão no DW, os dados presentes na tabela relacionada à área de estágio são carregados. Uma dimensão chamada “DIM_DATA” é criada para representar os dados detalhados referentes à data.

4) Definir Fato:

A tabela fato normalmente representa o assunto no contexto de um DW. Neste caso, a tabela “vendas” é considerada como forte candidata a tornar-se tabela fato.

A Figura 5.8 ilustra a tela “DEFINIR FATO” que é mostrada automaticamente após o usuário escolher as dimensões que irão compor o DW. Nesta tela já aparece a tabela candidata a ser a tabela fato, juntamente com os seus atributos que irão compor as métricas. Esses dados são extraídos do arquivo XML.

Após selecionar os atributos que serão as métricas e clicar no botão “DEFINIR FATO” é exibida a estrutura da tabela fato, sendo que a chave primária é composta pela

combinação de todas as chaves das dimensões definidas. Nesse momento é realizada a carga de dados da tabela de estágio para a tabela fato, sendo os dados sumarizados por dia.

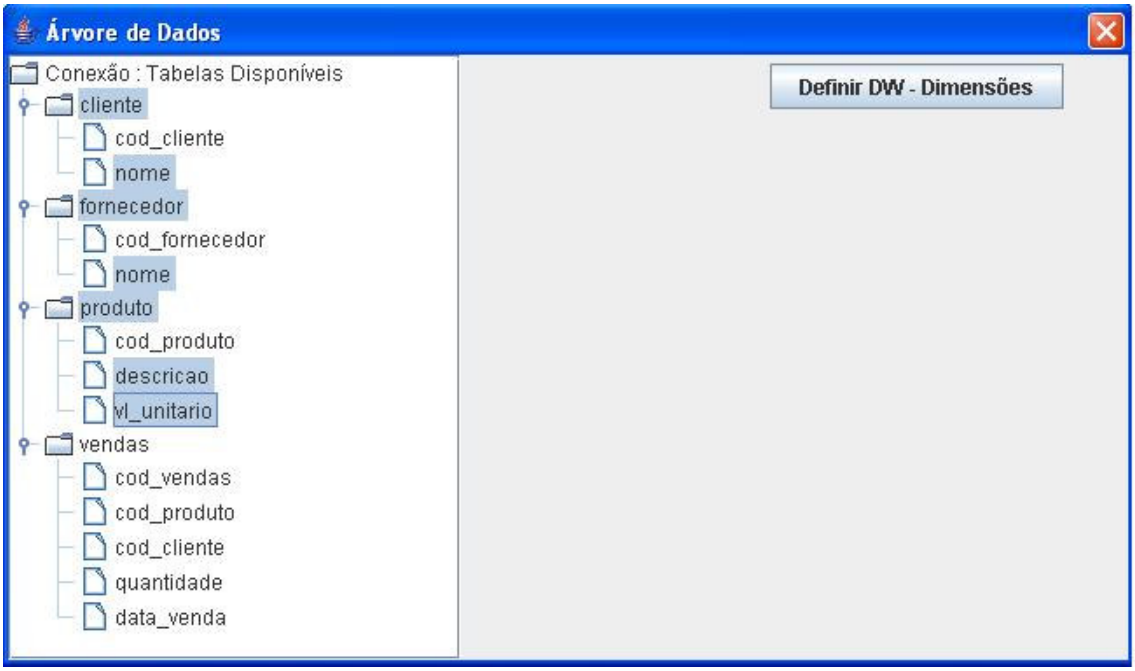

Figura 5.6: Tela de Definições de DW - Dimensões

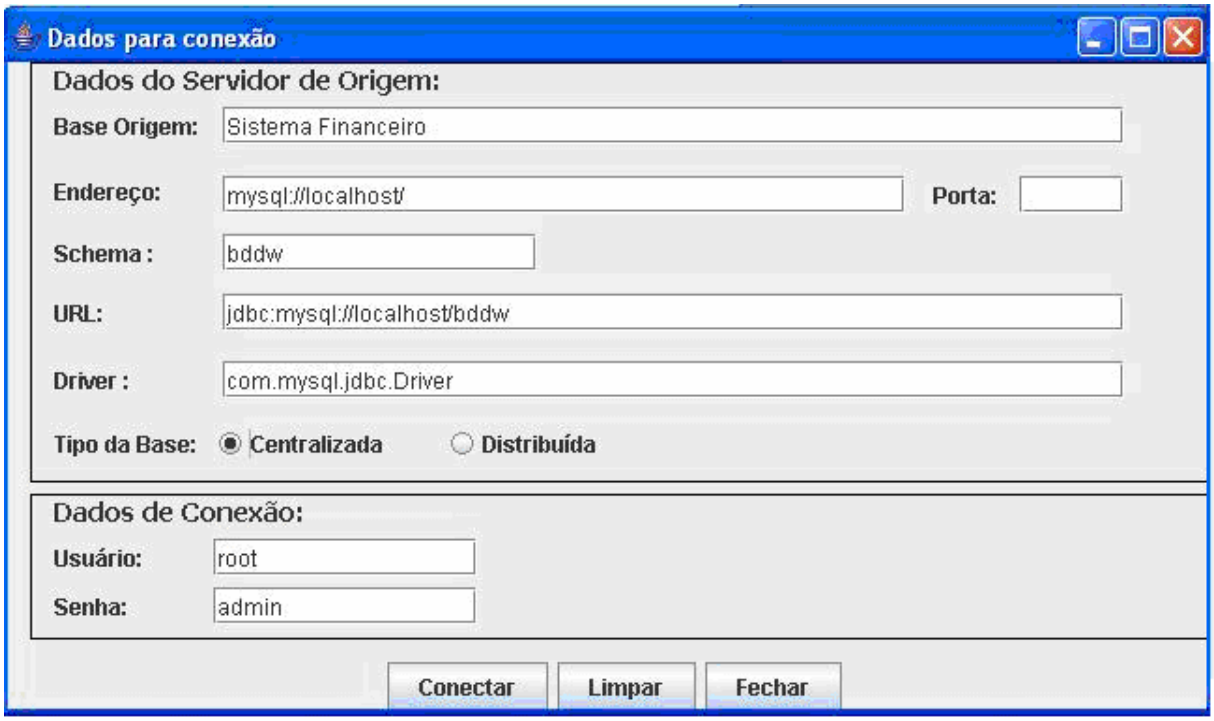

Figura 5.7: Tela de Definições de Localização do Banco de Dados do DW

Figura 5.8: Tela de Definições de DW - Fato

Após a definição das tabelas de dimensão e fato, o arquivo de metadados em XML, listado na seqüência, é complementado. Na tabela "fato_vendas", a métrica avaliada é a quantidade de produtos, que foi calculada por meio da soma da quantidade de um determinado produto de um mesmo cliente e em uma mesma data. Esse cálculo foi realizado utilizando comando SQL sobre os dados contidos na área de estágio.

```
. . .
<Fill_Data_DW>
   <Items_Datas_Stage>
    <Multidimensional>
      <Address_DW>mysql://localhost/:</Address_DW>
      <Type_DBMS_DB_DW>centralizado</Type_DBMS_DB_DW>
      <Version_DBMS_DB_DW></Version_DBMS_DB_DW>
      <User_Password_DW>root|admin</User_Password_DW>
      <DB_DW>bddw</DB_DW>
      <Aliases_DW></Aliases_DW>
      <Drivers_API_DW>com.mysql.jdbc.Driver</Drivers_API_DW>
      <Type_File_Source_DW></Type_File_Source_DW>
      <Tables_Attributes_Keys_DW>
       <Table_DW>fato_vendas</Table_DW>
       <Attributes_DW>
          <Name>dim_codigocliente</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
          <Name>dim_codigoproduto</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
          <Name>dim_codigodata</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
          <Name>quantidade</Name>
          <Type>int</Type>
          <Type_Transformation>Sum(bdestagio.vendas.quantidade) GROUP BY
bdestagio.stage_produto.cod_produto, bdestagio.stage_cliente.cod_cliente</Type_Transformation>
```

```
        </Attributes_DW>
        <Keys_DW>
          <Name>dim_codigocliente</Name>
          <Name>dim_codigoproduto</Name>
          <Name>dim_codigodata</Name>
        </Keys_DW>
        <Table_DW>dim_produto</Table_DW>
        <Attributes_DW>
          <Name>dim_codigoproduto</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
          <Name>cod_produto</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
          <Name>dim_descricao</Name>
          <Type>varchar</Type>
          <Type_Transformation></Type_Transformation>
          <Name>dim_vl_unitario</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
        </Attributes_DW>
        <Keys_DW>
          <Name>dim_codigoproduto</Name>
        </Keys_DW>
        <Table_DW>dim_cliente</Table_DW>
        <Attributes_DW>
          <Name>dim_codigocliente</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
          <Name>cod_cliente</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
          <Name>dim_nome</Name>
          <Type>varchar</Type>
          <Type_Transformation></Type_Transformation>
        </Attributes_DW>
        <Keys_DW>
          <Name>dim_codigocliente</Name>
        </Keys_DW>
         <Table_DW>dim_data</Table_DW>
        <Attributes_DW>
          <Name>dim_codigodata</Name>
          <Type>int</Type>
          <Type_Transformation></Type_Transformation>
          <Name>dim_data</Name>
          <Type>datetime</Type>
          <Type_Transformation></Type_Transformation>
        </Attributes_DW>
        <Keys_DW>
          <Name>dim_codigodata</Name>
        </Keys_DW>
      </Tables_Attributes_Keys_DW>
    </Multidimensional>
  </Items_Datas_Stage>
</Fill_Data_DW>

. . .
```

5) Consulta no DW:

A Figura 5.9 mostra uma consulta simples no DW, onde o usuário deverá apenas escolher a data da consulta e clicar o botão “Pesquisar”. Após essa escolha, é realizada uma busca na tabela fato e feito um somatório da quantidade de produtos vendidos na data selecionada e, em seguida, é apresentado ao usuário o resultado da pesquisa, sendo gerados os dados sobre a pesquisa realizada no arquivo de metadados, conforme apresentado no trecho XML a seguir.

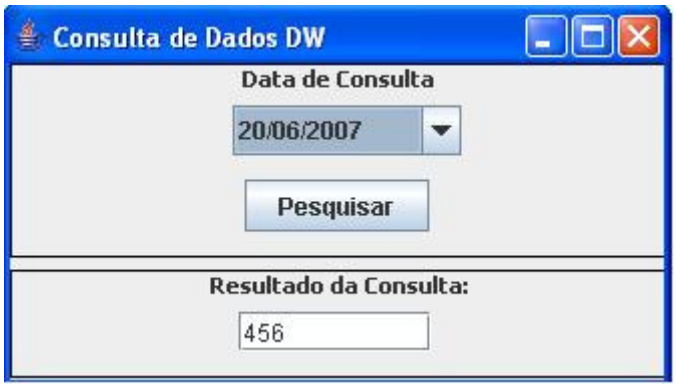

Figura 5.9: Tela de Consulta no DW

```
. . .
<Information_Visualization>
  <Powerd_Processing>
    <Items_Used_Analysis_Techniques>
      <Multimensional>
        <Table>fato_vendas</Table>
          <Attributes>
            <Name>dim_codigodata</Name>
            <Type>int</Type>
            <Type_Transformation></Type_Transformation>
            <Name>quantidade</Name>
            <Type>int</Type>
            <Type_Transformation>Sum(bddw.fato_vendas.quantidade)</Type_Transformation>
          </Attributes>
        <Table>dim_data</Table>
          <Attributes>
            <Name>dim_codigodata</Name>
            <Type>int</Type>
            <Type_Transformation></Type_Transformation>
            <Name>dim_data</Name>
            <Type>datetime</Type>
            <Type_Transformation></Type_Transformation>
          </Attributes>
      </Multimensional>
      <Tecni_Used_Result></Tecni_Used_Result>
      <Date>20/06/2007</Date>
      <Address_Result></Address_Result>
      <File_Name></File_Name>
    </Items_Used_Analysis_Techniques>
```

```
  </Powerd_Processing>
</Information_Visualization>
. . .
```

5.3 CONSIDERAÇÕES FINAIS

Para a realização do estudo de caso foi desenvolvido um protótipo que implementa a interface básica das principais etapas de um sistema KDD, voltada principalmente ao uso do componente de software.

Neste capítulo foi mostrada a execução do protótipo e como o usuário interage com o sistema para a construção e o acesso de um DW. Além das telas construídas, foram listados os trechos criados em XML a cada etapa realizada.

O arquivo XML, que representa o modelo de metadados proposto, é construído e recuperado pelo componente de gerenciamento de metadados.

CONCLUSÃO E TRABALHOS FUTUROS

Atualmente existe uma grande necessidade de competitividade em qualquer ramo da sociedade, isso devido as mais diversas concorrências enfrentadas. Um fator determinante para o sucesso de qualquer organização é conhecer informações de seus concorrentes e principalmente saber as principais características que envolvem o próprio negócio e, ainda, informações que permitam competitividade no mercado. Essas informações, geralmente, não são encontradas nos sistemas convencionais (transacionais). O DW propicia condições para a obtenção dessas informações estratégicas, uma vez que nele estão armazenados dados sobre os sistemas fundamentais de uma organização.

Para descobrir um conhecimento "escondido" em uma grande massa de dados são executadas consultas OLAP e mineração de dados nas tabelas multidimensionais. O processo que inclui a construção de um DW e a extração de conhecimento é denominado de Processo de Descoberta de Conhecimento em Banco de Dados, onde é fundamental a presença de metadados. Os metadados são responsáveis por armazenar informações tanto técnicas quanto de negócio do ambiente de descoberta de conhecimento. É fundamental saber a procedência de dados sumarizados no DW e se eles sofreram alguma alteração, além de informações sobre a localização dos dados de origem, da área de estágio e do DW, etc.

As ferramentas que prestam suporte para extração de conhecimento em grande massa de dados, tais como, o Oracle Warehouse Builder (OWB, 2006), Oracle9i OLAP (ORACLE9i, 2006) e MineSet (MINESET, 2006), apresentam modelos proprietários de metadados, assim sentiu-se a necessidade de definir um modelo de metadados para sistemas de descoberta de conhecimento (sistemas KDD).

Na definição do modelo de metadados, foi imprescindível a contribuição de Tannenbaum (2006), além de outros autores. Apesar dos modelos propostos na literatura apresentarem os elementos essenciais para documentar um sistema KDD, eles não se baseiam em qualquer tipo de padrão de metadados. A escolha de um padrão facilitaria o intercâmbio entre diversas ferramentas do gênero. Como padrão foi adotado o CWM, sendo a sua especialidade em DW a grande vantagem desse padrão em relação aos outros.

A importância do modelo de metadados definido e apresentado neste trabalho é o fato de ser persistido em linguagem XML, o que significa independência de plataforma computacional, um fator ponderável diante da diversidade computacional nos dias atuais. O fato dos metadados serem persistidos em XML diminui o *overhead*, justificando assim o seu

uso. Além disso, o modelo proposto abrange por completo a representação das informações necessárias de serem registradas em metadados para sistemas KDD.

Além do modelo de metadados, que mostrou os itens mínimos necessários para a contemplação do mesmo em um sistema KDD, neste trabalho foi realizada a especificação de um componente de software para gerenciamento deste modelo e demonstrada a execução passo-a-passo de um protótipo que possibilita a construção e o acesso a um DW utilizando este componente. Portanto, a principal contribuição deste trabalho é a identificação dos itens de metadados fundamentais a um sistema KDD.

Como trabalhos futuros podem ser citados:

- Estudo mais detalhado e uso do padrão CWM de forma mais completa;
- Projeto de uma interface para sistemas KDD;
- Desenvolvimento de um sistema KDD que inclui o componente de gerenciamento de metadados implementado.

REFERÊNCIAS BIBLIOGRÁFICAS

AMARAL, G. C. M. **XML Metadata Interchange (XMI).** Tópicos Especiais em Banco de Dados II. Núcleo de Computação Eletrônica. Mestrado em Informática, Universidade Federal do Rio de Janeiro, 2002.

BALLARD, C.; HERREMAN, D.; SCHAU, D.; BELL, R.; KIM, E.; VALENCIC, A. **Data Modeling Techniques for Data Warehousing.** IBM Corporation, 1998.

BARQUINI, R. **Planning and designing the Warehouse**. New Jersey: Prentice-Hall, 1996.

BARRETO, C. M. **Modelo de Metadados para a Descrição de Documentos Eletrônicos na WEB.** Programa de Pós-Graduação em Ciências e Computação. Dissertação de Mestrado. IME - Instituto Militar de Engenharia, 1999.

BOHM, K.; RAKOW, T.C. - "Metadata for Multimedia Documents". *Sigmod Record* 23, 4, 21-26, 1994.

BUSS, S. KAWASSAKI, V. **Desenvolvimento WEB- Estudos de Tecnologia**. Programa de Pós-Graduação. Pontifica Universidade Católica – Paraná. Disponível em <http://espec.ppgia.pucpr.br/~sbuss/tecno/default.htm>, 2000. Acesso em 20 dez 2005.

CARVALHO, J. R. ***Web services*** **como Integradores de Banco de Dados Distribuídos.** Monografia de Especialização em Engenharia de Software e Banco de Dados. Departamento de Computação. Universidade Estadual de Londrina, 2005.

CASTOLDI, A. V. **Um Modelo de Meta Dados para Suporte a Sistemas de Descoberta de Conhecimento em Banco de Dados.** Trabalho de Conclusão de Curso (Graduação) – Ciência da Computação. Departamento de Informática. Universidade Estadual de Maringá, 1999.

CHANG, D. T. **Common Warehouse Metamodel (CWM), UML and XML.** IBM Database Technology Institute. OMG CWM Working Group, Meta Data Conference, March 19-23, 2000.

CHAUDHURI, S.; DAYAL, U. **An Overview of Data Warehousing and OLAP Technology.** ACM SIGMOD *Record*. New York, Vol. 26., N° 1, 1997.

CIFERRI, C. D. A. **Distribuição dos Dados em Ambiente de Data Warehousing: O sistema WebD^{2}W e Algoritmos Voltados à Fragmentação Horizontal dos Dados.** Programa de Pós-Graduação em Ciência da Computação. Tese de doutorado. Universidade Federal de Pernambuco, Recife, Pernambuco, 2002.

COME, G. **Os metadados no ambiente de Data Warehouse.** *In* IV SEMEAD – Seminários de Administração. Programa de Pós-Graduação de Administração. FEA. Universidade de São Paulo, Outubro de 1999, 1999.

CWM. **Common Warehouse Metamodel Specification.** OMG (Object Management Group) Versão 1.0. Fevereiro de 2001, 2001;

DEMAREST, M. **The politics of data warehousing**. Disponível em <http://www.dmreview.com/whitepaper/wid293.pdf>, 1997. Acesso em: 20 dez 2005.

DE LUCCA, J. E. **Novas Tecnologias de Rede.** Apostila de Curso de Pós-Graduação em Computação. Departamento de Informática e Estatística. Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2003.

DIAS, M. M. **Um modelo de formalização do processo de desenvolvimento de sistemas de descoberta de conhecimento em banco de dados**. Programa de Pós-Graduação em Engenharia de Produção. Tese de Doutorado. Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2001.

ELMASRI, R; NAVATHE, S. B. **Sistemas de Banco de Dados.** 4a ed. São Paulo: Addison Wesley, 2005.

FELDENS, M. A.; MORAES, R. L.; PAVAN, A.; CASTILHO, J. M. V. **Towards a methodology for the discovery of useful knowledge combining data mining, data warehousing and visualization.** *In Proceedins of the* XXIV CLEI (Conferência Latino-

Americana de Informática). Quito. Equador. Disponível em <http://jacui.inf.ufrgs.br/~feldens/clei98.html>, 1998.

GROTH, R. **Data mining: A Hands-on Approach for Business Professionals.** USA: Prentice Hall PTR., 1998.

GUPTA, S. K; BHATNAGAR, V.; WASAN, S.K. **Architecture for Knowledge Discovery and Knowledge Management - Knowledge and Information Systems**, 2004.

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

HURWITZ, J. **Preparing for the Warehouse – DBMS Maganize.** Disponível em <http://www.dbmsmag.com/9604d04.html>, 1996. Acesso em 20 dez 2005.

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

INMON, W. H. **Metadata in the Data Warehouse**. Disponível em <www.billinmon.com/library/whiteprs>, 2000. Acesso em 20 dez 2005.

MATOS JR., N. O. **Middlewares XML para Banco de Dados Relacional.** Programa de Pós-Graduação em Informática (Mestrado). Centro de Ciências e Tecnologia. Universidade Federal da Paraíba, 2001.

KASHYAP, V.; SHAH, K.; SHETH, A. **Metadata for Bulding the Multimedia Patch Quilt**. Multimedia Database System: Issues and Research Directions, Springer-Verlag, 1995.

KIMBALL, R. **Data Warehouse Toolkit: Pratical Techniques for Building Dimensional Data Warehouses.** New York: John Wiley & Sons, 1996.

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

LANGIANO, B. C.; FÁVERO, D. F.; REBELLO, D. R. **A aplicação de XML no Desenvolvimento de Aplicações Distribuídas.** Projeto de Iniciação Científica (PIC)-UEM. Universidade Estadual de Maringá, 2002.

LANS, R. V. **O que é data mining e uma análise do mercado de produtos.** *In Proceedings of the* 8° Congresso Nacional de Novas Tecnologias e Aplicações em Bance de Dados, 1997.

LEAL, L. N. **Gerador de aplicações para repositórios de metadados baseados em XSL e XML Schema.** Trabalho de Conclusão de Curso (Graduação) – Informática. Departamento de Ciência da Computação do Instituto de Matemática. Universidade Federal do Rio de Janeiro, 2003.

LÖFFLER, J.; BIATOV, K.; ECKES, C.; KÖHLER, J. **IFINDER: An MPEG-7-Based Retrieval System for Distributed Multimedia Content.** *In Proceedings of the 10Th ACM International Conference on Multimedia*, Pg. 431-435, Juan-les-Pins, France, 2002.

MACHADO, F. N. R. **Projeto de Data Warehouse – Uma visão Multidimensional**. Érica, 2000.

MANILLA, H. **Data mining: machine learning, statistics and databases.** *In Proceedings of the Eight International Conference on Scientific and Database Management*, Pg. 1-8, 1996.

MARINO, M. T. **Integração de Informações em Ambientes Científicos na WEB: Uma Abordagem Baseada na Arquitetura RDF.** Dissertação de Mestrado. IM/NCE – Instituto de Matemática e Núcleo de Computação Eletrônica. Universidade Federal do Rio de Janeiro, 2001.

MENOLLI, A. L. A. **Definição de uma Arquitetura de Data Warehouse para Gestão em Ciência e Tecnologia no Brasil.** Programa de Pós-Graduação em Ciência da Computação. Dissertação de Mestrado. Departamento de Informática. Universidade Estadual de Maringá, 2004.

MINESET. **Silicon Graphics Computer Systems – MineSet.** Disponível em <http://www.sgi.com/chembio/apps/datamine.html>. Acesso em 20 jan 2006.

MOURA, D. F. C. **XML – Extensible Markup Language.** Disciplina de Redes de Computadores – COPPE/UFRJ. Universidade Federal do Rio de Janeiro. Disponível em <http://www.gta.ufrj.br/~mdavid/xml.htm>. Acesso em 20 dez 2005.

OMG XMI. **XML Metadata Interchange (XMI).** Disponível em: <ftp://ftp.omg.org/pub/docs/ad/98-10-95.pdf>, 2001. Acesso em 20 dez 2005.

ORACLE9i. **Oracle9*i* Database.** Disponível em <http://otn.oracle.com/software/products/oracle9i/>. Acesso em 20 jan 2006.

OWB. **Oracle Warehouse Builder Documentation.** Disponível em <http://www.oracle.com/technology/documentation/warehouse.html>. Acesso em 20 jan 2006.

OMG MOF. **Meta Object Facility (MOF) Specification – Version 1.4.** Disponível em: <www.omg.org>, 2002. Acesso em 20 dez 2005.

ORENSTEIN, O. A. **XMI Metadata Interchange.** Tópicos Especiais em Banco de Dados II. Núcleo de Computação Eletrônica. Mestrado em Informática. Universidade Federal do Rio de Janeiro, 2001.

PITTS-MOULTIS, N.;KIRK, C. **XML Black Book.** Editora Makron Books, São Paulo, 2000.

PEREIRA, D. M. **Uso do Padrão OIM de Metadados no Suporte às Transformações de Dados em Ambiente de Data Warehouse. .** Dissertação de Mestrado. IM/NCE – Instituto de Matemática e Núcleo de Computação Eletrônica. Universidade Federal do Rio de Janeiro, 2000.

SANTOS, A. Cerqueira. **Introdução a Metadados.** In CienteFico, Ano III, Vol. II, Salvador, 2003.

SANTOS, M. S. **Uma proposta para a integração de Modelos de Padrões de Software com Ferramentas de Apoio ao Desenvolvimento de Sistemas.** Dissertação de Mestrado. Departamento de Computação. Universidade Federal do Ceará, 2004.

SILVA, E. L.; MENEZES, E. M. **Metodologia de pesquisa e elaboração de dissertação.** Laboratório de Ensino à Distância da UFSC. Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2000.

SINGH, H. **Data Warehousing: Concepts, Technologies, Implementations and Management – Upper Saddle River.** NJ: Prentice Hall, 1997.

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

SONG, Y.; ROWEN, W.; MEDSKER, C.; EWEN, E. **An Analysis of Many-to-Many Relatioships Between Fact and Dimension Tabels in Dimensional Modeling.** *In Proceedings of the International Workshop on Design and Management of Data Warehouse* (DMDW 2001), Interlaken, Suiça, 2001.

TANNENBAUM, A. **Metadata Solutions: Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand.** New York: Addison Wesley, United States of America, 2002.

VALENTIN, L. G. **Uma Arquitetura para um Sistema de Descoberta de Conhecimento.** Programa de Pós-Graduação em Ciência da Computação. Dissertação de Mestrado. Departamento de Informática. Universidade Estadual de Maringá, 2006.

VETTERLI, T.; VADUVA, A.; STAUDT, M. **Metadata Standards for Data Warehousing: Open Information vs. Commom Warehouse Metamodel.** ACM SIGMOD, Vol. 29, nº 3, p. 68-75, Setembro de 2000, 2000.

APÊNDICE A

Na Tabela A.1 estão relacionados os itens do modelo de metadados proposto. Esta tabela é dividida em três colunas principais que são nomeadas como: Beneficiário, Nome do Item e Procedência do Metadados.

As categorias beneficiárias são definidas como sendo as principais atividades realizadas em cada etapa do processo KDD. Para Tannenbaum (2002), a categoria beneficiária é aquela que se beneficiará com a informação referente ao item de metadados ("Nome do Item" na Tabela A.1).

A procedência dos metadados representa a fonte da informação que irá compor o metadados. Por exemplo, a informação do item de metadados "Nomes dos Negócios" pode ter como procedência uma ferramenta CASE (*Computer Aided Software Engineering*), usada para modelar os dados desse sistema, ou um dicionário de dados interno do próprio metadados. Outro exemplo, o item de metadados "Nomes dos Bancos de Dados de Origem" é extraído do próprio SGBD (Sistema Gerenciador de Banco de Dados).

Tabela A.1: Dados do Modelo de Metadados para um Sistema KDD

Beneficiário		**Nome do Item**	**Procedência do Metadados**
Pré-Processamento	**Busca**	Endereços das Fontes de Dados de Origem	Dicionário de Dados Interno
		Tipo de SGBD de cada Banco de Dados de Origem	Dicionário de Dados Interno
		Versão do SGBD dos Bancos de Dados de Origem	SGBD de Origem
		Nomes dos Usuários e Senhas de Origem	SGBD de Origem
		Nomes dos Bancos de Dados de Origem	SGBD de Origem
		Nomes das Tabelas, Atributos e Chaves de Origem	SGBD de Origem
		Nomes de *Aliases* de Origem	SGBD de Origem
		Nomes dos D*rivers e* API's de Origem	Dicionário de Dados Interno
		Tipos de Arquivos Fontes de Origem	Dicionário de Dados Interno
		Nomes das Aplicações Legadas	Dicionário de Dados Interno
	Transformação	Item de Dados Buscados nas Fontes	Dicionário de Dados Interno
		Transformações Realizadas nos Dados/Regras das Transformações	Dicionário de Dados Interno

Beneficiário			Nome do Item	Procedência do Metadados
Pré-Processamento	**Carga**	**Estágio**	Itens de Dados Transformados	Dicionário de Dados Interno
			Endereços das Fontes de Dados de Destino	Dicionário de Dados Interno
			Tipo de SGBD do Banco de Dados de Estágio	Dicionário de Dados Interno
			Versão do SGBD do Banco de Dados de Estágio	SGBD do Estágio
			Nomes dos Usuários e Senhas de Estágio	SGBD do Estágio
			Nomes dos Bancos de Dados de Estágio	SGBD do Estágio
			Nomes das Tabelas, Atributos e Chaves de Estágio	SGBD do Estágio
			Nomes de *Aliases* de Estágio	SGBD do Estágio
			Nomes dos D*rivers e* APIs de Estágio	Dicionário de Dados Interno
			Tipos dos Arquivos Fontes de Estágio	Dicionário de Dados Interno
			Freqüência da Carga dos Dados de Estágio	Dicionário de Dados Interno
			Volume de Dados do Estágio	SGBD do Estágio
			Data do Último Estágio	Dicionário de Dados Interno
			Log	Dicionário de Dados Interno
		Povoamento	Item de Dados do Estágio	Dicionário de Dados Interno
			Endereço da Fonte de Dados do DW	Dicionário de Dados Interno
			Tipo de SGBD do DW	Dicionário de Dados Interno
			Versão do SGBD do DW	SGBD do DW
			Nomes dos Usuários e Senhas do DW	SGBD do DW
			Nomes do Banco de Dados do DW	SGBD do DW
			Nomes das Tabelas, Atributos e Chaves do DW	SGBD do DW
			Nomes de *Aliases* do DW	SGBD do DW
			Nomes dos D*rivers e* APIs do DW	Dicionário de Dados Interno
			Tipo de Arquivos Fontes do DW	Dicionário de Dados Interno
			Operações de Povoamento	Dicionário de Dados Interno
			Freqüência de Povoamento	Dicionário de Dados Interno
			Volume de Dados do Povoamento	SGBD do Estágio
			Data do Último Povoamento	Dicionário de Dados Interno
			Log	Dicionário de Dados Interno

Beneficiário	Nome do Item	Procedência do Metadados
Aplicação de técnicas análise	Informações sobre Algoritmos de Mineração	Dicionário de Dados Interno
	Informações de Acesso aos Algoritmos de Mineração	Dicionário de Dados Interno
	Informações sobre Ferramentas OLAP	Dicionário de Dados Interno
	Informações de Acesso às Ferramentas OLAP	Dicionário de Dados Interno
Pós-Processamento	Itens de Dados Povoados Utilizados na Aplicação da Técnica de Análise	Dicionário de Dados Interno
	Técnica Aplicada na Geração do Resultado	Dicionário de Dados Interno
	Nome Arquivo de Resultado	Dicionário de Dados Interno
	Endereço do Arquivo de Resultado	Dicionário de Dados Interno
	Nomes dos Negócios	Dicionário de Dados Interno/Ferramenta CASE

APÊNDICE B

```
<?xml version="1.0" standalone="yes"?>
<Metadata_Model>
<Foundation>
  <Business_Information>Vendas a Varejo</Business_Information>
  <Expressions>
    <Expressions1>
            <Transformation_Role>sum(a,b)</Transformation_Role>
              <Parameters>
                    <Name>a</Name>
                    <Type>Integer</Type>
                    <Name>b</Name>
                    <Type>Integer</Type>
              </Parameters>
   </Expressions1>
  </Expressions>
</Foundation>
<Resource>
     <Search>
          <Relational>
                <Address_Source_Origin>\\servidor2\BancosDados\</Address_Source_Origin>
                <Type_DBMS_DB_Origin>centralizado</Type_DBMS_DB_Origin>
                <Version_DBMS_DB_Origin>2.0</Version_DBMS_DB_Origin>
                <User_Password_Origin>josecarlos|123</User_Password_Origin>
                <DB_Origin>BDAdmFin|MySQL</DB_Origin>
                <Tables_Attributes_Keys_Origin>
                   <Table>Cliente</Table>
                   <Attributes>
                      <Name>Cod_Cliente</Name>
                      <Type>Integer</Type>
                      <Name>Nome</Name>
                     <Type>String(25)</Type>
                     <Name>Endereço</Name>
                     <Type>String(30)</Type>
                     <Name>Telefone</Name>
                     <Type>Integer</Type>
                   </Attributes>
                   <Keys>Cod_Cliente</Keys>
                   <Table>Produto</Table>
                   <Attributes>
                      <Name>Cod_Produto</Name>
                      <Type>Integer</Type>
                      <Name>Descrição</Name>
                      <Type>String(20)</Type>
                      <Name>Vl_Unitário</Name>
                      <Type>Integer</Type>
                   </Attributes>
                   <Key>Cod_Produto</Key>
                   <Table>Vendas</Table>
                   <Attributes>
                      <Name>Cod_Venda</Name>
                      <Type>Integer</Type>
                      <Name>Cod_Cliente</Name>
                      <Type>Integer</Type>
                      <Name>Cod_Produto</Name>
                      <Type>Integer</Type>
                      <Name>Quantidade</Name>
                      <Type>Integer</Type>
                      <Name>Data</Name>
                      <Type>DateTime</Type>
                   </Attributes>
                   <Keys>Cod_Venda</Keys>
                </Tables_Attributes_Keys_Origin>
                <Aliases_Origin>Sistema_Vendas</Aliases_Origin>
                <Drivers_API_Origin>MySQL ODBC 3.51 Driver</Drivers_API_Origin>
```

```
                <Type_File_Source_Origin>.optl.frml.mydl.myi</Type_File_Source_Origin>
                <Legacy_Application>Sistema Administrativo-Financeiro 2.1</Legacy_Application>
            </Relational>
        </Search>
        <Pre_Processing>
            <Load>
                <Stage_Data>
                    <Relational>
                        <Address_Source_Destiny>\\servidor3\AreáEstágio\</Address_Source_Destiny>
                        <Type_DBMS_DB_Destiny>centralizado</Type_DBMS_DB_Destiny>
                        <Version_DBMS_DB_Destiny>9i</Version_DBMS_DB_Destiny>
                        <User_Password_Destiny>martins|123456</User_Password_Destiny>
                        <DB_Destiny>BDEstágio|Oracle</DB_Destiny>
                        <Tables_Attributes_Keys_Destiny>
                            <Table>Cliente</Table>
                            <Attributes>
                                <Name>Cod_Cliente</Name>
                                <Type>Integer</Type>
                                <Type_Transformation>BDAdmFin.Cliente.Cod_Cliente</Type_Transformation>
                                <Name>Nome</Name>
                                <Type>String(20)</Type>
                                <Type_Transformation>Substr(BDAdmFin.Cliente.Nome,1,20)</Type_Transformation>
                                <Name>Endereço</Name>
                                <Type>String(30)</Type>
                                <Type_Transformation>RTrim(BDAdmFin.Cliente.Endereço)</Type_Transformation>
                                <Name>Telefone</Name>
                                <Type>String</Type>
                                <Type_Transformation>IntToStr(BDAdmFin.Cliente.Telefone)</Type_Transformation>
                            </Attributes>
                            <Keys>Cod_Cliente</Keys>
                            <Table>Produto</Table>
                            <Attributes>
                                <Name>Cod_Produto</Name>
                                <Type>Integer</Type>
                                <Type_Transformation>BDAdmFin.Produto.Cod_Produto</Type_Transformation>
                                <Name>Descrição</Name>
                                <Type>String(15)</Type>
                                <Type_Transformation>Substr(BDAdmFin.Produto.Descrição,1,10)</Type_Transformation>
                                <Name>Vl_Unitário</Name>
                                <Type>Integer</Type>
                                <Type_Transformation>BDAdmFin.Produto.Vl_Unitário</Type_Transformation>
                                <Name>Marca</Name>
                                <Type>String(5)</Type>
                                <Type_Transformation>Substr(BDAdmFin.Produto.Descrição,11,15)</Type_Transformation>
                                <Name>Tipo_Pacote</Name>
                                <Type>String(5)</Type>
                                <Type_Transformation>Substr(BDAdmFin.Produto.Descrição,16,20)</Type_Transformation>
                            </Attributes>
                            <Keys>Cod_Produto</Keys>
                            <Table>Vendas</Table>
                            <Attributes>
                                <Name>Cod_Venda</Name>
                                <Type>Integer</Type>
                                <Type_Transformation>BDAdmFin.Vendas.Cod_Venda</Type_Transformation>
                                <Name>Cod_Cliente</Name>
                                <Type>Integer</Type>
                                <Type_Transformation>BDAdmFin.Vendas.Cod_Cliente</Type_Transformation>
                                <Name>Cod_Produto</Name>
                                <Type>Integer</Type>
                                <Type_Transformation>BDAdmFin.Vendas.Cod_Produto</Type_Transformation>
                                <Name>Quantidade</Name>
                                <Type>Integer</Type>
                                <Type_Transformation>BDAdmFin.Vendas.Quantidade</Type_Transformation>
                                <Name>Data</Name>
                                <Type>DateTime</Type>
                                <Type_Transformation>DateFormat(BDAdmFin.Vendas.Data,'dd/mm/aaaa')</Type_Transformation>
                            </Attributes>
```

```
            <Keys>Cod_Venda</Keys>
          </Tables_Attributes_Keys_Destiny>
          <Aliases_Destiny>DW_Vendas</Aliases_Destiny>
          <Drivers_API_Destiny>Oracle ODBC Driver</Drivers_API_Destiny>
          <Type_File_Source_Destiny>.ora</Type_File_Source_Destiny>
        </Relational>
  </Stage_Data>
  <Fill_Data_DW>
      <Items_Datas_Stage>
        <Multidimensional>
          <Address_DW>\\servidor4\DW</Address_DW>
          <Type_DBMS_DB_DW>centralizado</Type_DBMS_DB_DW>
          <Version_DBMS_DB_DW>10g</Version_DBMS_DB_DW>
          <User_Password_DW>martins|123456</User_Password_DW>
          <DB_DW>BDDW|Oracle</DB_DW>
          <Tables_Attributes_Keys_DW>
            <Table>Vendas_Fato</Table>
            <Attributes>
              <Name>Chave_Produto</Name>
              <Type>Integer</Type>
              <Type_Transformation></Type_Transformation>
              <Name>Chave_Data</Name>
              <Type>Integer</Type>
              <Type_Transformation></Type_Transformation>
              <Name>Quant_Vendas</Name>
              <Type>Integer</Type>
              <Type_Transformation>Sum(BDAdmFin.Vendas.Quantidade)</Type_Transformation>
              <Name>Tot_Vendas</Name>
              <Type>Real</Type>
              <Type_Transformation>Sum(BDAdmFin.Vendas.Quantidade*BDAdmFim.Produto.Vl_Unitário)</Type_Transformation>
            </Attributes>
            <Keys>Chave_Produto|Chave_Data</Keys>
            <Table>Produto_Dimensão</Table>
            <Attributes>
              <Name>Chave_Produto</Name>
              <Type>Integer</Type>
              <Type_Transformation></Type_Transformation>
              <Name>Descrição</Name>
              <Type>String(15)</Type>
              <Type_Transformation>Substr(BDAdmFin.Produto.Descrição,1,10)</Type_Transformation>
              <Name>Marca</Name>
              <Type>String(5)</Type>
              <Type_Transformation>Substr(BDAdmFin.Produto.Descrição,11,15)</Type_Transformation>
              <Name>Tipo_Pacote</Name>
              <Type>String(5)</Type>
              <Type_Transformation>Substr(BDAdmFin.Produto.Descrição,16,20)</Type_Transformation>
            </Attributes>
            <Keys>Chave_Produto</Keys>
            <Table>Data_Dimensão</Table>
            <Attributes>
              <Name>Chave_Data</Name>
              <Type>Integer</Type>
              <Type_Transformation></Type_Transformation>
              <Name>Data</Name>
              <Type>DateTime</Type>
              <Type_Transformation>DateFormat(BDAdmFin.Vendas.Data,'dd/mm/aaaa')</Type_Transformation>
              <Name>Feriado</Name>
              <Type>Boolean</Type>
              <Type_Transformation>Holyday(BDAdmFin.Vendas.Data)</Type_Transformation>
              <Name>Dia_Semana</Name>
              <Type>Boolean</Type>
              <Type_Transformation>Week_Day(BDAdmFin.Vendas.Data)</Type_Transformation>
            </Attributes>
            <Keys>Chave_Data</Keys>
          </Tables_Attributes_Keys_DW>
                <Aliases_DW>DW_Project</Aliases_DW>
    <Drivers_API_DW>Oracle ODBC Driver 10g</Drivers_API_DW>
```

```
                    <Type_File_Source_DW>.ora</Type_File_Source_DW>
                  </Multidimensional>
                 </Items_Datas_Stage>
                </Fill_Data_DW>
    </Load>
   </Pre_Processing>
</Resource>
</Analysis>
     <Tecniques_Analysis_Application>
           <Data_Mining>
              <Algorithm_Mining>Cluster|Árvore Decisão</Algorithm_Mining >
              <Access>\\servidor3\Algoritmos\</Access>
           </Data_Mining>
           <OLAP>
              <Information_Tools>Oracle9i OLAP</ Information_Tools>
              <Access>\\servidor3\Ferramentas_OLAP\</Access>
           </OLAP>
     </Tecniques_Analysis_Application>
      <Information_Visualization>
         <Powerd_Processing>
            <Items_Used_Analysis_Techniques>
              <Multimensional>
                 <Table>Vendas_Fato</Table>
                 <Attributes>
                    <Name>Chave_Produto</Name>
                    <Type>Integer</Type>
                    <Type_Transformation></Type_Transformation>
                    <Name>Chave_Data</Name>
                    <Type>Integer</Type>
                    <Type_Transformation></Type_Transformation>
                    <Name>Quant_Vendas</Name>
                    <Type>Integer</Type>
                    <Type_Transformation>Sum(BDDW.Vendas_Fato.Quantidade)</Type_Transformation>
                 </Attributes>
                 <Table>Produto_Dimensão</Table>
                 <Attributes>
                    <Name>Chave_Produto</Name>
                    <Type>Integer</Type>
                    <Type_Transformation></Type_Transformation>
                    <Name>Descrição</Name>
                    <Type>String(15)</Type>
                    <Type_Transformation>Substr(BDAdmFin.Produto.Descrição,1,10)</Type_Transformation>
                    <Name>Marca</Name>
                    <Type>String(5)</Type>
                    <Type_Transformation>Substr(BDAdmFin.Produto.Descrição,11,15)</Type_Transformation>
                 </Attributes>
              </Multimensional>
              <Tecni_Used_Result>Classificação Qt. Vendas por Marca</Tecni_Used_Result>
              <Date>22/11/2006</Date>
              <Address_Result>\\servidor4\Resultados\</Address_Result>
              <File_Name>Resultado1.doc</File_Name>
            </Items_Used_Analysis_Techniques>
         </Powerd_Processing>
     </Information_Visualization>
</Analysis>
<Management>
     <Warehouse_Process>
        <Pass1>Search</Pass1>
        <Pass2>Expressions</Pass2>
        <Pass3>Stage_Data</Pass3>
        <Pass4>Fill_Data_DW</Pass4>
     </Warehouse_Process>
     <Warehouse_Operation>
         <Stage>
            <Frequen_Load_Datas>diárialmensal|trimestrallesporádica</Frequen_Load_Datas>
            <Volume_Datas_Stage>10 GB</Volume_Datas_Stage>
            <Date_Last_Load_Stage>20/11/2006</Date_Last_Load_Stage>
```

```
            <Log>mcarlos</Log>
          </Stage>
          <Fill>
            <Operation_Fill>Fato Venda: Sumário de vendas p/ semana</Operation_Fill>
            <Frequency_Fill>diárialmensalltrimestrallesporádica</Frequency_Fill>
            <Volume_Datas_Fill>80 GB</Volume_Datas_Fill>
            <Date_Last_Fill>21/11/2006</Date_Last_Fill>
            <Log>mhelena</Log>
          </Fill>
      </Warehouse_Operation>
  </Management>
</Metadata_Model>
```

Figura B.1: Modelo de Metadados Proposto

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo