GILMAR KAVALCO

AUTOMATIZAÇÃO DA EVOLUÇÃO DE ESQUEMAS EM FERRAMENTAS DE TRANSPORTE DE DADOS PARA DATA WAREHOUSE

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre, Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Marcos Sfair Sunye

CURITIBA
2001

UFPR
Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática do aluno ***Gilmar Kavalco***, avaliamos o trabalho intitulado ***"Automatização da Evolução de Esquemas em Ferramentas de Transporte de Dados para Data Warehouse"***, cuja defesa foi realizada no dia 08 de junho de 2001. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 08 de junho de 2001.

Prof. Dr. Marcos Sfair Sunye
Presidente - Orientador

Prof. Dr. Carlos Alberto Maziero
Membro Externo - PUC/PR

Profª. Dra. Laura Sanchez Garcia
DINF/UFPR

Dedico este trabalho à minha esposa Elvira e a meu filho Matheus que sempre me apoiaram durante todo o tempo em que estive desenvolvendo este estudo.

AGRADECIMENTOS

Inicialmente gostaria de agradecer aos amigos Aroldo Pereira Vieira, Cícero Giacomitti, Attílio Zanellato Neto e Mário Jorge Schneider, que direta ou indiretamente participaram deste estudo e dos resultados aqui contidos.

Aos gerentes Amaury Lemberg Filho e Izabel Beluso Iwamura e a coordenadora Cláudia Amorim do Banco Banestado, que me apoiaram durante o período de estudos, através de ajuda e compreensão nas atividades do cotidiano na empresa.

A Universidade Federal do Paraná e a todo o Departamento de Informática, professores e funcionários.

Ao meu orientador Marcos Sfair Sunye por sua competência e atenção no desenvolvimento deste estudo.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

B2B	-	BUSINESS TO BUSINESS
CSV	-	COMMA SEPARATED VALUES
DBA	-	DATABASE ADMINISTRATOR
DOM	-	DOCUMENT OBJECT MODEL
DTD	-	DOCUMENT TYPE DEFINITION
DW	-	DATA WAREHOUSE
HTML	-	HYPERTEXT MARKUP LANGUAGE
OLAP	-	ON-LINE ANALYTICAL PROCESSING
OLTP	-	ON-LINE TRANSACTION PROCESSING
SGBD	-	SISTEMA GERENCIADOR DE BASE DE DADOS
SGML	-	STANDARD GENERALIZED MARKUP LANGUAGE
SQL	-	STRUCTURED QUERY LANGUAGE
TSV	-	TAB SEPARATED VALUES
XML	-	EXTENSIBLE MARKUP LANGUAGE
W3C	-	WORLD WIDE WEB CONSORTIUM

RESUMO

A diversidade de bases de dados encontradas nas empresas vem criando um dos desafios mais importantes na área de Bancos de Dados. Este desafio é o de tornar interoperáveis as diversas bases de dados existentes nestas empresas. Uma forma de conseguir a interoperabilidade dos dados é através da criação de um esquema integrado a partir dos sistemas aplicativos operacionais, tais como Recursos Humanos, Contabilidade, Contas a Pagar, entre outros. O conjunto de todos estes sistemas permite criar um ambiente integrado e consolidado de informações da empresa. Este tipo de ambiente têm como característica a contínua evolução dos diversos modelos de dados dos sistemas que participam da consolidação como fornecedores de informações. Este trabalho estuda o processo de administração do transporte contínuo de dados para um ambiente consolidado e o controle automático da evolução dos esquemas que participam da integração, visando a criação de um ambiente de *data warehouse*. Buscou-se uma solução através do desenvolvimento de um modelo de dados, o qual chamamos de "integrador". Com base neste modelo foi construída uma ferramenta para administrar e automatizar todo o movimento de informações sobre mudanças nos modelos, informações estas que caracterizam a evolução dos modelos. Para dar maior independência na solução e padronizar a comunicação entre modelos, foi utilizado XML na formatação dos arquivos e nas mensagens entre esquemas. Para testar a ferramenta e o modelo "integrador" proposto, desenvolveu-se um projeto piloto de *data warehouse* simulando um ambiente do setor bancário.

Palavras-chave: Integração de esquemas; XML; *Data Warehouse*.

ABSTRACT

The database's diversity in corporations generates one of the most important challenges in the area of databases. This challenge is to reach the interoperability among the several existent databases in these companies. A form of getting the communication of the data is through the creation of an integrated schema starting from the systems operational applications, such as Human Resources, Accounting, Accounting Payable, among others. The set of all these systems allow to create an environment integrated and consolidated of the company information. This type of environment has as characteristic the continuous evolution of the diverse system data models that participate in the consolidation as supplying of information. This work studies the process management of the continuous carrier of data for a consolidated environment and the automatic control of the evolution of the projects that participate in the integration, aiming at the creation a data warehouse environment. A solution was searched through the development of a data model that we call "integrator". Based in this model, a tool was built to accomodate the legacies databases schema evolution. To give greater independence in the solution and to standardize the communication among models, XML was used in the formatting of the archives and the messages among schemas. To test the tool and the model "integrator", we developed a prototype of data warehouse simulating an environment in a Business Bank.

Key words: Schema integration; XML; Data Warehouse.

1 INTRODUÇÃO

A informatização nas empresas tem criado a necessidade dos mais diversos sistemas de informações, os quais suportam a expansão de seus negócios.

A riqueza de ferramentas, metodologias e linguagens criadas para atender a esta demanda por novos sistemas é responsável pela heterogeneidade das soluções existentes nas empresas. Os bancos de dados não são uma exceção a esta realidade.

Em função disto, surge o problema de fazer com que as diversas bases existentes se integrem possibilitando que novos sistemas aplicativos sejam implantados.

Em KIM (1995) encontramos um possível caminho para a comunicação entre esquemas, permitindo interoperabilidade de dados, através da utilização de *gateways*. Serviços de *gateways* permitem que pares de bancos de dados comuniquem-se entre si, simplificando o acesso remoto de dados.

Outra forma de integrar esquemas permitindo a interoperabilidade entre os dados é consolidá-los através da criação de um ambiente de *data warehouse*.

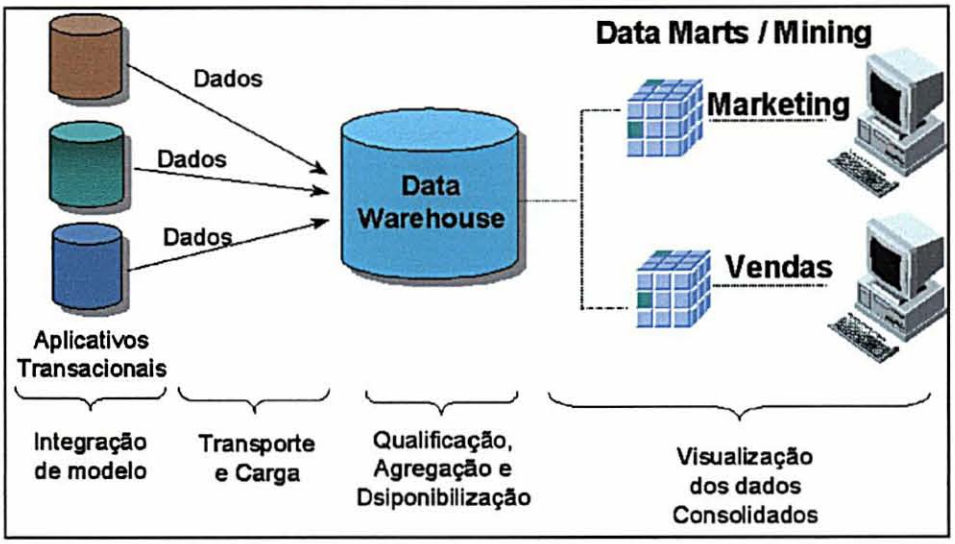
Segundo INMON (1997, p33), *data warehouse* é uma coleção de dados orientados por assunto, integrados, variando no tempo e não volátil, direcionados ao suporte na tomada de decisões.

Ralph KIMBALL (1996) define *data warehouse* como uma cópia de dados transacionais especificamente estruturados para consultas e análises.

A partir destes autores conclui-se que um *Data warehouse* é um banco de dados de informações integradas, extraídas de aplicativos transacionais das empresas, disponíveis para consultas e análises. Consultas sobre as bases de dados com estas informações tornam-se facilitadas pelo fato de estarem consolidadas em um único ambiente.

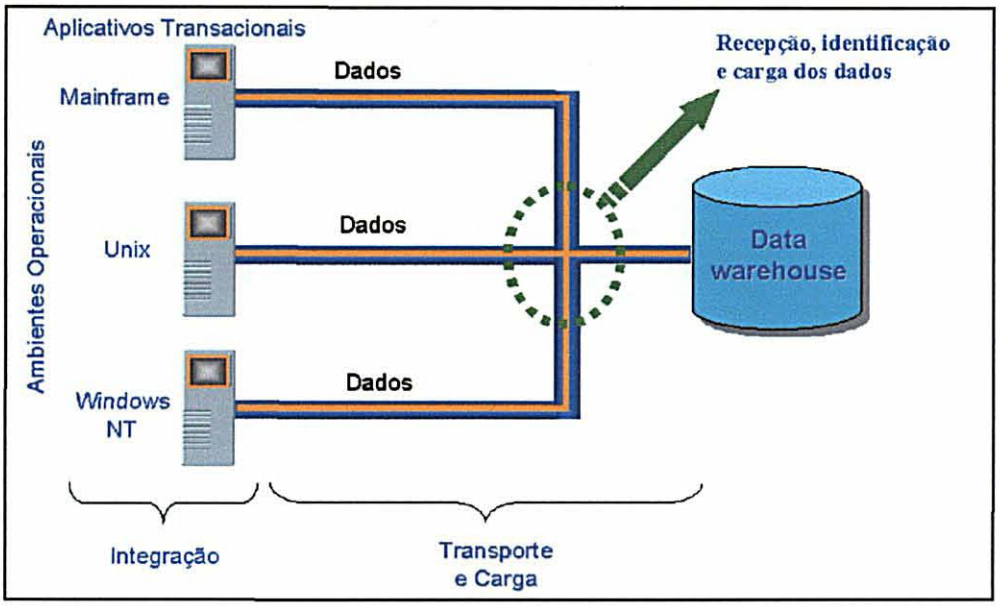
O processo de criação e manutenção deste tipo de ambiente reúne uma diversidade de passos, como mostrado na FIGURA 1. Podemos destacar a integração dos modelos origem em um modelo consolidado, a transferência dos dados dos aplicativos transacionais de origem para o ambiente integrado, a qualificação, agregação e disponibilização no ambiente de *data warehouse* e a visualização dos dados disponibilizados.

FIGURA 1 – GERAÇÃO DE UM *DATA WAREHOUSE*



Este estudo tem sua ênfase no passo do transporte de dados dos aplicativos origem para o ambiente de *data warehouse,* preocupando-se em como facilitar e controlar os processos de recepção e carga dos dados vindos dos aplicativos transacionais, como mostrado na FIGURA 2.

FIGURA 2 – TRANSPORTE E CARGA DE DADOS



Dentro do contexto de transporte e carga dos dados, estudamos o problema de automatizar o sincronismo de alterações das bases de dados dos aplicativos transacionais e da base do *data warehouse.*

A esse sincronismo chamou-se de "evolução dos modelos envolvidos na integração".

Por "evolução de modelos" entende-se qualquer alteração que seja realizada no modelo origem ou destino.

Com o objetivo de propor uma alternativa para controlar esta evolução de modelos, foi desenvolvido um protótipo de um ambiente de *data warehouse* e criada uma ferramenta para auxiliar na administração e controle da evolução dos modelos de dados e no recebimento dos dados para serem carregados no modelo consolidado.

A evolução de um modelo de dados pode ser exemplificada através da inclusão de um novo atributo em um aplicativo de Controle de Clientes, onde é inserido o endereço do correio eletrônico. Um outro exemplo seria a inclusão no modelo integrado de um atributo indicando a quantidade de saques feitos em caixa automático pelo cliente. Ambos

alteram o modelo original e ambos exigirão revisão dos processos de integração. Podemos considerar estas mudanças nos aplicativos como evoluções de seus modelos.

Para criar o ambiente de *data warehouse* do protótipo foram integrados aplicativos baseados em sistemas operacionais diversos dentro de um ambiente bancário, tais como *IBM/VM, Unix e Windows/NT*.

O processo de integração levou em consideração a necessidade de dar uma visão única dos negócios de investimento, empréstimo e conta-corrente. O processo de geração dos dados pelos aplicativos transacionais respeita os tempos de fechamento das informações para um determinado dia, obedecendo as regras de negócio e regulamentações do Banco Central do Brasil.

A partir da definição da forma de integração dos aplicativos transacionais, foram padronizados os leiautes dos arquivos a serem transferidos, iniciando assim os procedimentos para efetivar a integração.

Com o objetivo de ganhar em clareza e documentação, a padronização dos arquivos recebidos foi realizada utilizando a formatação no padrão *Extensible Markup Language* (XML), que é publicado pelo *Word Wide Web Consortium* (W3C).

A especificação XML muitas vezes é confundida com uma linguagem de programação, outras vezes com um tipo de arquivo. XML é na verdade uma linguagem de especificação para servir, transportar e armazenar dados. Foi projetada para ser fácil de implementar e interoperar tanto com *Standard Generalized Markup Language* (SGML) como com *Hypertext Markup Language* (HTML) (BRAY; PAOLI; SPERBERG-MKQUEEN, 1998).

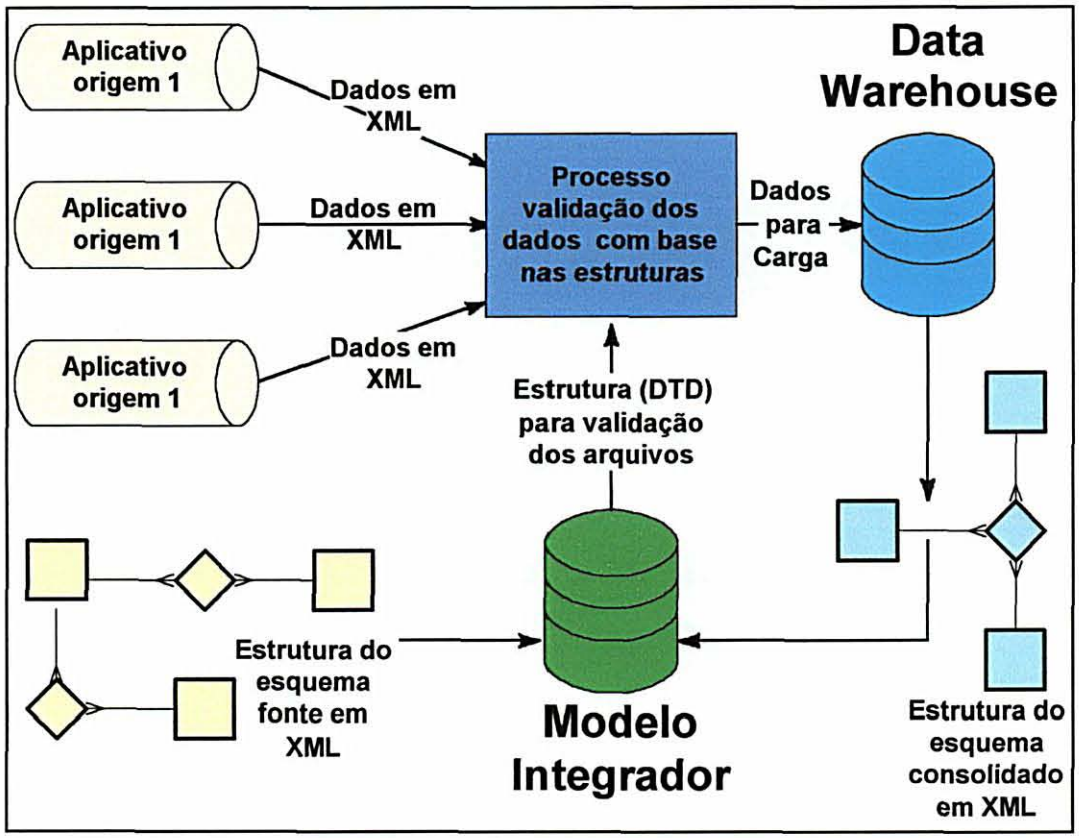
Para qualquer metodologia de integração de esquemas que seja utilizada, como as citadas em BATINI, LENZERINI e NAVATHE (1986) e SPACCAPIETRA e PARENT (1998), faz-se uma integração inicial automática atingindo um percentual maior ou menor no que tange à totalidade dos atributos.

Normalmente um processo de integração não consegue atingir a integração total dos atributos em função das diferentes formas de definir uma informação. Por exemplo, em um modelo é usado localidade e em outro cidade para representar a naturalidade do cliente.

Existem técnicas associadas às metodologias que conseguem identificar a maioria dos sinônimos para uma determinada integração, mas no final acaba sendo necessário a intervenção manual de um "integrador de esquemas", que pode ser o administrador de banco de dados, administrador de dados ou um usuário experiente. Ele deverá decidir como integrar os campos que restaram ou interferir numa determinada integração proposta.

Com as informações de como foram integrados os esquemas, foi criado um esquema de dados chamado "modelo integrador". Este modelo objetivou absorver o conhecimento do resultado final da integração e utilizá-lo como auxílio na validação das estruturas dos arquivos que estavam sendo recepcionados no *data warehouse*, como mostrado na FIGURA 3.

FIGURA 3 – INSERÇÃO DO ESQUEMA INTEGRADOR



Com o mapeamento tanto dos esquemas origem quanto do esquema conceitual integrado e armazenando todas estas informações no modelo integrador, foi possível a geração automática do *Document Type Definition* (DTD) dos arquivos de transferência de dados, dentro da estrutura XML, que padroniza e documenta a integração. A partir da DTD é feita a validação do arquivo que está sendo recebido.

Uma DTD indica as regras que o documento XML está seguindo (LIGHT,1999). Segundo ANDERSON, et al (2000), a DTD usa uma gramática formal para especificar as estruturas e valores permitidos para um documento XML.

Para os procedimentos de atualização da estrutura dos dados no modelo integrador, como mostrado na FIGURA 3, foi utilizado um protocolo em que a informação do tipo de operação que deve ser

executada, como inclusão, alteração ou exclusão de atributo, está no próprio registro. Um processo recebe este arquivo e faz a leitura do seu conteúdo, gerando um comando SQL equivalente para ser executado no banco de dados. Este protocolo também foi definido utilizando XML.

O protótipo fez uso deste modelo integrador e ampliou o seu escopo para atender às necessidades de administração do fluxo de carga dos arquivos recepcionados.

Para apresentar o resultado dos estudos realizados, este trabalho está organizado da seguinte maneira: o capítulo 2 aborda os conceitos de interoperabilidade de bases de dados, suas formas e a relevância para este estudo, o ambiente integrado de *data warehouse* e o padrão XML; o capítulo 3 apresenta a forma de automatizar as alterações que são efetuadas nos modelos, mostrando também como o integrador faz as intervenções para permitir manter a integridade do modelo; o capítulo 4 apresenta o estudo de caso com a estratégia de integração utilizada na implementação; as conclusões do trabalho estão no capítulo 5, mostrando os resultados obtidos com o estudo; o capítulo 6 aborda as melhorias possíveis e trabalhos futuros a serem realizados.

2 INTEGRAÇÃO E INTEROPERABILIDADE

As integrações de bancos de dados, segundo KIM (1995), trabalham com o objetivo de agrupar bases heterogêneas, através de um multibanco ou pela conversão e migração de todos os dados de um gerenciador de banco de dados para outro.

Outro tipo de integração que permite interoperabilidade de dados que estão dispersos em diversos bancos de dados é através da geração de uma base de dados consolidada, com o objetivo de manter histórico e não de alterar os dados já cadastrados, como por exemplo um ambiente de *data warehouse*.

Segundo SPACCAPIETRA e PARENT (1998), interoperabilidade total pode ser alcançada por bancos de dados distribuídos ou federados, que suportem integrações em um banco de dados virtual (isto é, banco de dados que são logicamente definidos, mas não fisicamente materializados). Bancos de dados federados permitem a existência de bases de dados que ficam sob o controle de seus respectivos donos, portanto suportando uma harmoniosa coexistência e uma escalabilidade na integração dos dados, mantendo a autonomia das bases locais.

A integração de esquemas tem atraído muitos pesquisadores, gerando diversas contribuições tais como SPACCAPIETRA e PARENT (1998), MURPHY, KULKARNI e ROTHENBERGER (2000), CONRAD et al (1999), DUSCHKA, GENESERETH e LEVY (2000). Este estudo não aborda as metodologias de integração, mas sim a forma de manter uma integração atualizada com as evoluções existentes nos modelos originais.

2.1 INTEGRAÇÃO E TRANSFERÊNCIA DE DADOS

Quando abordamos o tema de integração de dados com o controle da evolução dos esquemas, e não a integração de modelos de dados, a preocupação está no passo seguinte ao da integração de

esquemas, ou seja, no processo que transfere dados de uma origem para um destino.

Esta transferência de dados dos aplicativos origem para o ambiente integrado pode ocorrer de forma síncrona ou assíncrona, dependendo da necessidade da atualização da informação.

O objetivo da transferência síncrona é refletir as alterações feitas em um banco de dados em outro de forma *on-line*, ou seja imediatamente. Esta pode ser através de replicação de dados ou através da utilização de monitor de transações com confirmação da transação em duas fases.

Para permitir a comunicação imediata entre bancos de dados, podemos utilizar serviços de *gateways* para pares específicos de gerenciadores de bancos de dados (KIM, 1995), como *Oracle* para DB2/IBM ou Sybase para Informix.

Segundo KIM (1995) os *gateways* provêm facilidades para definição de uma camada de persistência para diversos bancos de dados, simplificando o acesso para bases distantes, mas não suportam a aplicação de consistência automáticas ou regras de validação entre diferentes bases de dados.

SPACCAPIETRA e PARENT (1998) definem *gateway* como softwares que provêem facilidades para definir visões persistentes sobre diferentes bases de dados, simplificam o acesso a dados distantes, mas não suportam aplicação automática de regras de consistências e de restrições entre bases de dados diferentes.

O problema da utilização de *gateways* é a forte dependência entre os bancos e suas versões comerciais, o que acaba gerando uma série de transtornos tanto para o banco integrado quanto para os bancos origem, havendo uma dependência de atualização por parte dos fornecedores da ferramenta que está sendo utilizada, o que em alguns casos exige até a reescrita de alguns programas do aplicativo.

Outro problema é que em qualquer alteração nas bases de origem ou destino, existe a possibilidade de serem necessárias alterações nos procedimentos de integração. Por exemplo, a alteração de campos em

uma tabela com a exclusão de um, dois ou qualquer número de atributos, pode gerar manutenção em programas de integração ou de mapeamento de atributos.

Nas aplicações onde as atualizações dos dados não são requeridas de forma *on-line*, pode-se trabalhar com movimentos periódicos, de forma assíncrona, variando entre intervalos de minutos, horas, dias ou qualquer outro intervalo de tempo. Neste tipo de transferência, podemos ter o movimento de um único ou de *n* registros, formando arquivos de tamanhos e tipos variados [INMON, 1997).

As formas de comunicações, utilizando a transferência assíncrona, podem ser através de ferramentas de entrega de dados ou de transferência de arquivos. O que importa é que o arquivo gerado em um determinado lugar seja transferido para o outro, dentro do tempo estipulado e que o processo que originou a transferência possa identificar que a entrega foi concretizada.

Nas transferências assíncronas, pode ou não existir uma ligação direta entre as bases, para as quais estão sendo transferidos dados, havendo então a necessidade de que a estrutura do dado no modelo origem e destino, seja conhecida tanto por quem gera quanto por quem recebe.

Objetivando a facilidade de obter e disponibilizar dados em um modelo consolidado, poderíamos optar, dependendo da problemática proposta, pela utilização de *gateways*, pela criação de um banco de dados federado ou de um banco consolidado, utilizando comunicação assíncrona.

Para a construção do protótipo de *data warehouse* deste estudo, definiu-se que a forma de comunicação deveria ser assíncrona através de arquivos de dados, para buscar o máximo de independência possível entre os esquemas.

2.2 MODELO BÁSICO DE UM AMBIENTE *DATA WAREHOUSE*

O conceito de *data warehouse* vem evoluindo, deixando de ser um sistema de apoio a decisão, ou sistema de informações para executivos para se tornar efetivamente um sistema de apoio à corporação. Isto quer dizer que neste tipo de ambiente serão encontrados dados integrados, consistentes e de fácil acesso.

Segundo INMON (1996), um dos primeiros a estudar o tema, um *data warehouse* é uma coleção de dados orientada por assuntos, integrada, variante no tempo, não volátil, que tem por objetivo dar suporte aos processos de tomada de decisão.

De uma maneira geral, um *data warehouse* é um banco de dados que contém dados dos sistemas operacionais, extraídos para serem consolidados, integrados e disponibilizados às pessoas que dele necessitam.

INMON (1996) descreve também que um *data warehouse* não é apenas um conjunto de dados mas também ferramentas para consultar, analisar e apresentar as informações. Este ambiente é o lugar onde os dados são publicados, podendo ser analisados das mais diversas formas e combinações.

Diferentes abordagens são encontradas no mercado, sendo que poderíamos distingui-las em:

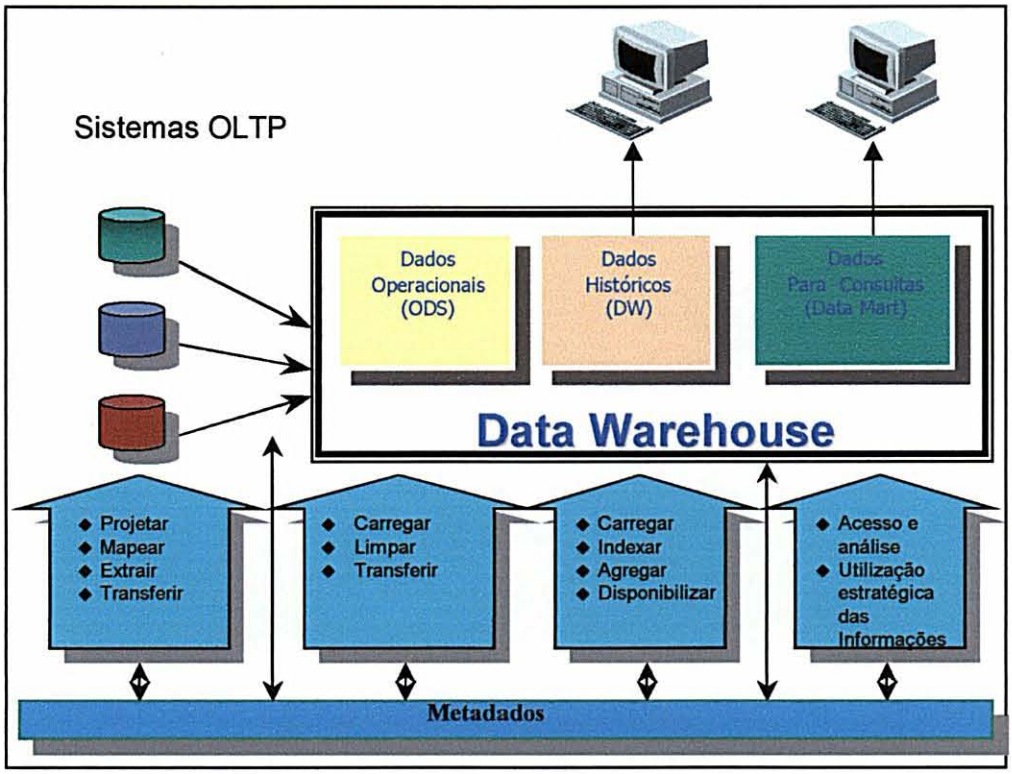
a) criar um ambiente de *data warehouse* integrado em um único banco de dados;
b) distribuir os dados do *data warehouse* com base no tipo de informação, com dados financeiros em um servidor, dados para marketing em outro e dados de empréstimos em outro;
c) distribuir os dados do *data warehouse* por nível de agregação, fazendo com que dados extremamente resumidos estejam em um servidor, dados no nível de detalhe intermediário ficam em outro servidor e dados detalhados num terceiro servidor.

Na primeira abordagem há uma tentativa de maximizar o uso dos computadores pela concentração de todas as informações. Na segunda a distribuição passa a ser feita por assunto, distribuindo o peso do processamento através da utilização de diversos servidores. A terceira abordagem é similar a segunda, sendo que aqui são feitas agregações nas tabelas, diminuindo o grau de detalhe extraído.

Outro ponto que provoca muita discussão é a composição da arquitetura de um ambiente de *data warehouse*.

Na FIGURA 4 é apresentada uma arquitetura genérica, mostrando uma das inúmeras alternativas possíveis.

FIGURA 4 - ARQUITETURA GERAL DE DW



A base de todo o processo é o estudo dos dados dos sistemas transacionais, também chamados de *On-line Transaction Processing* (OLTP), baseados nos aplicativos da empresa, onde é necessário que sejam feitos os projetos de integração, mapeamentos e as definições de

como extrair e transferir dados do ambiente produtivo para o *data warehouse*.

O *data warehouse* é dividido em três partes, sendo que a primeira tem a característica de não ser visualizada pelos usuários.

Na primeira parte, ambiente do *Operational Data Storage* (ODS), são carregados os dados, feitas as validações com o objetivo de qualificá-los e limpá-los, deixando-os íntegros e integrados. Em seguida são transferidos para o ambiente de *Data Warehouse* (DW).

No DW, que é o armazém dos dados históricos, guardam-se os dados pelo tempo que estiver definido no projeto, como cinco anos, dez anos ou variável conforme a informação e o seu nível de detalhe.

Na terceira parte está a camada de informações com a qual a grande maioria dos usuários irá interagir. Este ambiente é derivado do DW com o objetivo de montar cenários onde o acesso aos dados, com rapidez e segurança são os pontos principais. Nesta camada utilizam-se as mais diversas ferramentas para acesso aos dados, tais como planilhas, pacotes estatísticos, geradores de relatórios e ferramentas de análise do tipo *On-line Analytical Processing* (OLAP).

Um ponto de extrema importância dentro de todo o ciclo do *data warehouse* é o controle dos metadados.

Metadados são as informações sobre os dados mantidos na empresa, e neste caso em especial de todos os componentes envolvidos no *data warehouse*, como a descrição de cada campo integrado, a conceituação de cada fonte de informação, descrição de todos os programas envolvidos desde a primeira extração de dados até a disponbilização final para os usuários.

Informações de desempenho, cargas e monitoramento do sistema também são metadados que devem ser analisados para avaliar o comportamento de todo o ambiente.

O mercado de *data warehouse* está crescendo rapidamente em função de projetos que têm alcançado sucesso e das empresas desenvolvedoras de Sistemas Gerenciadores de Banco de Dados (SGBD)

que vem desenvolvendo produtos com maior capacidade de processamento, com tempo de resposta cada vez melhores e pela necessidade de atendimento à demanda de informações aos executivos das corporações de forma íntegra, consolidada e rápida.

Kimball (1996, p 279) comenta que "embora existam grandes avanços sendo feitos em hardware, principalmente na área de processamento paralelo, o futuro do *data warehouse* pertence ao software". Ele lista alguns dos pontos que deverão melhorar significativamente nos próximos anos:

a) otimização das estratégias de execução para consultas de junção em esquema estrela;
b) indexação de tabelas de dimensão, em especial tabelas de muitos milhões de linhas;
c) acesso e indexação da chave composta de grandes tabelas de fatos;
d) extensão do SQL para processar consultas do estilo OLAP;
e) suporte a processamento paralelo;
f) ferramentas para projeto de bancos de dados multidimensionais;
g) ferramentas para extração e administração;
h) ferramentas de consulta para o usuário final;

Isto nos mostra que a área de software tem um grande avanço pela frente e que em parceria com os novos *hardwares*, os ambientes de *data warehouse* irão ganhar maior robustez, confiabilidade e agilidade.

2.3 UMA VISÃO DA XML – EXTEND MARKUP LANGUAGE

XML é uma linguagem de descrição de dados estruturados e um subconjunto da SGML - Standard Generalized Markup Language (Linguagem Padrão de Marcação Generalizada), padrão internacional

(ISO 8879) para definição de estruturas e conteúdo de documentos eletrônicos. A idéia básica da XML é a de ser uma linguagem para definição de documentos que fundamentalmente os separa em duas partes distintas: o conteúdo e a forma.

Qualquer documento XML conterá dados e os metadados que os definem e lhes permite a materialização na forma desejada. Um metadado pode ser definido como informações auxiliares sobre os dados. Por exemplo, quando temos dentro de um arquivo a seguinte composição:

<data_movto>
<ano>2000</ano>
<mês>03</mês>
<dia>27</dia>
</data_movto>

Os dados propriamente ditos são "2000", "03" e "27" e os metadados que descrevem a informação são "data_movto", "ano", "mês" e "dia".

Ao invés de um conjunto fixo de marcações dedicados à apresentação, como no HTML, a XML permite que marcadores sejam definidos dinamicamente, o que traz uma grande flexibilidade para descrição de documentos com características auto-interpretáveis.

Sendo assim, a primeira vantagem óbvia é que num mesmo documento poderá ser produzido na sua forma final em várias mídias/formatos diferentes, ou simplesmente em uma delas, incluindo página HTML.

O entusiasmo pela XML se justifica pelo seu potencial uso em duas grandes áreas: a primeira seria a possibilidade de uma linguagem universal para definição de documentos, com sintaxes específicas para os mais variados domínios do conhecimento (matemática, química, comércio, etc.); e a segunda, pelo seu uso como um transportador de documentos no emergente segmento de comércio eletrônico, atuando na integração de aplicações e nos sistemas B2B (*business-to-business*) (BARBIERI, 2000).

A criação de modelos pré-formatados, com informações básicas de aplicações, tais como transações financeiras, catálogos de registros de museus, estruturas moleculares e transmissões de arquivos, entre outras, passaram a ser referenciadas como vocabulários XML (ANDERSON, et al, 2000).

A XML procura alcançar o compromisso entre flexibilidade, simplicidade e legibilidade tanto para humanos quanto para máquinas.

Além da especificação da XML 1.0, há muitas outras normas envolvidas no desenvolvimento e criação de módulos opcionais que fornecem conjuntos de delimitadores e atributos ou ferramentas adicionais para tarefas específicas.

Dentre esses vários módulos e ferramentas disponíveis ou em desenvolvimento, pode-se citar:

a) Xlink - modo padrão para adição de hipertexto a um arquivo XML;
b) XPointer e XFragments - sintaxes para endereçamento de partes de um documento XML;
c) CSS - Cascade style sheet language, utilizado para definir forma e padrão de apresentação, aplicável tanto ao XML como ao HTML;
d) XSL - linguagem avançada para expressar style sheets, empregada para reordenação, adição ou deleção de delimitadores e atributos, sendo também utilizada como forma de apresentação dos dados;
e) DOM - conjunto de funções pré-definidas para manipulação de arquivos XML a partir de linguagens de programação, sendo essencialmente um Application Program Interface (API) que define um padrão de interação dos desenvolvedores com os dados estruturados;

f) XML Namespaces - especificação que descreve como associar um URL a cada delimitador ou atributo em um documento;

g) XML Schemas 1 e 2 - auxiliam desenvolvedores a definir de forma mais precisa suas próprias estruturas de dados no formato XML, como por exemplo a especificação de um elemento como um *integer*, *float*, ou outro tipo qualquer.

Arquivos XML são geralmente maiores que arquivos de conteúdo similar representados em formato binário. Tal decisão foi tomada de forma consciente pelos desenvolvedores de aplicações XML. A desvantagem do crescimento do arquivo pode ser compensada a partir de outras técnicas, como compressão de arquivos, transmissão de dados empregando protocolos de comunicação adequados e redução de custos do espaço em disco rígido (MOURA).

As descrições completas de todas estas tecnologias, seus documentos oficiais, recomendações, especificações e outras tecnologias relacionadas a estas, estão disponíveis no W3C *web site*.

3 AUTOMATIZAÇÃO DA ATUALIZAÇÃO DO MODELO DE DADOS

Dentro do processo de integração de modelos, existe o problema de manutenção do modelo integrado, para qualquer objetivo que seja proposto, desde unificar modelos operacionais, unificar plataformas ou esquemas de diversos sistemas aplicativos, passando também pela montagem de bases para um ambiente de *data warehouse*.

Nos casos onde a integração é feita uma única vez, como por exemplo na substituição de um sistema aplicativo, é possível que não seja necessário montar uma estrutura de manutenção do processo de integração, que por muitas vezes pode se tornar dispendioso para uma empresa.

Mas quando abordamos o problema da consolidação de dados em que o modelo anterior não é desativado, mas simplesmente fornece dados para um sistema integrado e tem que manter atualizado o processo de integração, o controle do processo torna-se essencial.

Os sistemas de informações gerenciais, de *data warehouse* e de gerenciamento de relacionamento com cliente, são típicos deste último tipo de integração. Os aplicativos operacionais continuam existindo, mas devem enviar informações de forma organizada, padronizada e periódica para uma base de dados que fará a união de diversas fontes.

Outra característica destes ambientes com dados consolidados e integrados é o tipo de modelagem de dados realizada. A integração das bases passa por um processo de análise das necessidades. Em alguns casos passa por algum processo de integração automática, sendo que em praticamente todas as integrações passa pela utilização da figura chamada de Integrador de Esquemas, o qual faz a análise final da integração e mapeamento para as entidades destino.

Uma forma de controle das atualizações de como os modelos foram integrados e de qual a estrutura de cada arquivo que está sendo recebido no ambiente de consolidação de dados pode ser feita através da utilização de um esquema que seja intermediário entre os esquemas

origem e destino. A função deste esquema intermediário é de armazenar informações de como estão sendo integrados os esquemas origem e qual o mapeamento existente entre os esquemas origem e o consolidado.

Para integrar dois ou mais modelos, pode-se utilizar uma das metodologias comentadas por BATINI, LENZERINI e NAVATHE (1986).

Segundo estes autores, cada metodologia de integração segue seus próprios procedimentos, podendo no entanto, qualquer metodologia ser considerada como uma mistura das seguintes atividades:

a) Integração prévia: Uma análise do esquema é feita antes da integração, para decidir sobre as políticas de integração. Isto direcionará as escolhas dos esquemas para serem integrados, a ordem de integração e a possibilidade de definir preferencias para esquemas inteiros ou parte deles. Estratégias globais para integração tais como quantidade de interação do integrador de esquemas e o numero de esquemas a serem integrados ao mesmo tempo. Buscar informações adicionais que sejam relevantes para a integração, como afirmações ou restrições que devem ser aplicadas ao modelo, também fazem parte desta fase.
b) Comparação de esquemas: Esquemas são comparados e analisados para determinar a correspondência entre conceitos e detectar possíveis conflitos.
c) Conformidade de esquemas: Uma vez que os conflitos são detectados, um esforço é feito para resolvê-los para que a fusão de vários esquemas seja possível. Resolução automática de conflitos geralmente não é possível, sendo requerida a interação de um integrador de esquemas ou um usuário para que as atividades de integração na vida real possam ser alcançadas.
d) Fusão e integração: Neste estágio, os esquemas já estão prontos para serem sobrepostos, dando origem a algum esquema integrado intermediário. Este resultado intermediário é

> analisado e, se necessário, reestruturado de forma a alcançar a qualidade desejada. Um esquema conceitual *global* pode ser testado contra os alguns critérios de qualidade. Estes critérios são a completeza e exatidão, indicando se todos os conceitos dos esquemas integrados estão presentes e se estão corretos. Um segundo critério diz respeito a definição única de um conceito que apareça em mais de um componente do esquema. Por último, o esquema integrado deve ser fácil de ser entendido tanto pelo integrador quanto pelo usuário final.

Analisando a atividade c) entre os itens acima, verificamos que com a integração inicial, atinge-se um percentual maior ou menor no que tange à totalidade dos atributos. Este processo pode ser executado de forma automática através de alguma ferramenta de integração de esquemas que já esteja desenvolvida ou que seja objeto de desenvolvimento.

Após este passo, se a integração não foi totalmente completada, é necessária a intervenção do Integrador, que pode ser o administrador de banco de dados, administrador de dados ou um usuário experiente, que deverá decidir como integrar os campos que restaram ou interferir numa integração proposta.

Neste trabalho, buscou-se a criação de uma ferramenta, composta por processos e um modelo de dados, que administrasse e recepcionasse informações tanto dos esquemas dos aplicativos origem quanto do esquema consolidado, criando então um esquema para intermediar e armazenar todos estes dados, o qual chamou-se de "modelo integrador".

Para os procedimentos de atualização do modelo integrador, representado na FIGURA 5, foi utilizado um protocolo escrito em XML, que contém um registro com alterações solicitadas.

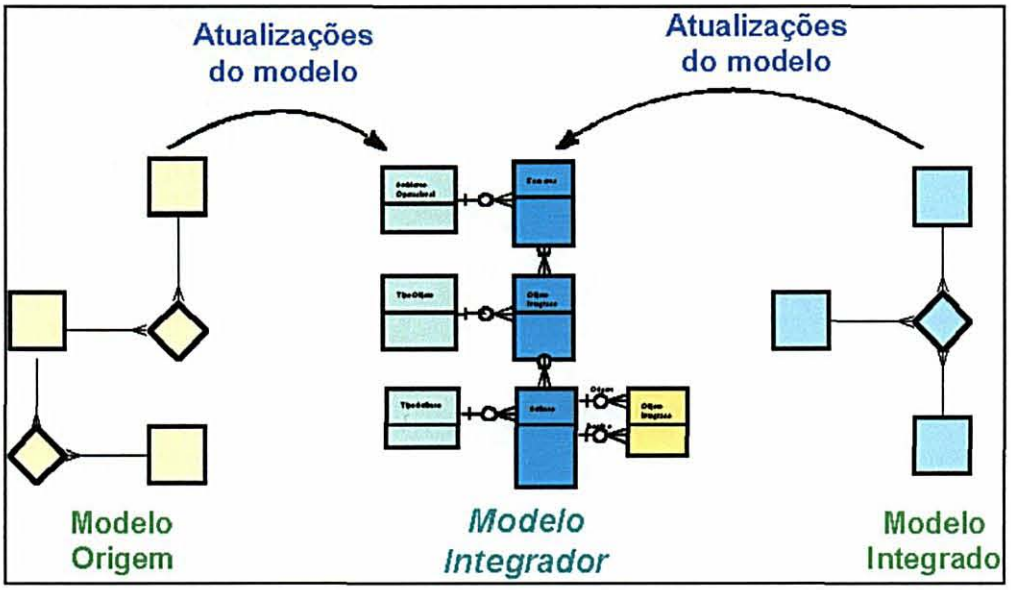
A utilização de XML na escrita do protocolo se deve ao interesse de estar independente de ferramentas comerciais e de linguagens

comerciais. O que se buscou foi um padrão genérico com possibilidade de implantação em qualquer plataforma e em qualquer banco de dados.

A informação do tipo de operação que deve ser executada, como inclusão, alteração ou exclusão de um elemento do esquema, podendo ser este elemento um atributo, objeto ou até o esquema completo, está no próprio registro. Um processo recebe este arquivo e faz a leitura do seu conteúdo, gerando um comando equivalente na linguagem SQL para ser executado no banco de dados.

Por meio deste tipo de solução, buscou-se uma alternativa de atualização assíncrona e independente da evolução dos modelos participantes de um ambiente integrado.

FIGURA 5 - ATUALIZAÇÃO DO MODELO INTEGRADOR



Com a atualização dos modelos sendo enviadas pelos esquemas origem e destino, é possível então administrar o mapeamento do ambiente consolidado. Como não há uma ligação direta entre os esquemas, tipo um *gateway*, os modelos são modificados sem que um interfira no outro, sendo necessário que exista o sincronismo somente no momento em que os dados sejam enviados da origem para o destino.

3.1 PROCEDIMENTOS DE ATUALIZAÇÃO

Os problemas a serem solucionados no processo de atualização do modelo integrador são os de inclusão, alteração e exclusão.

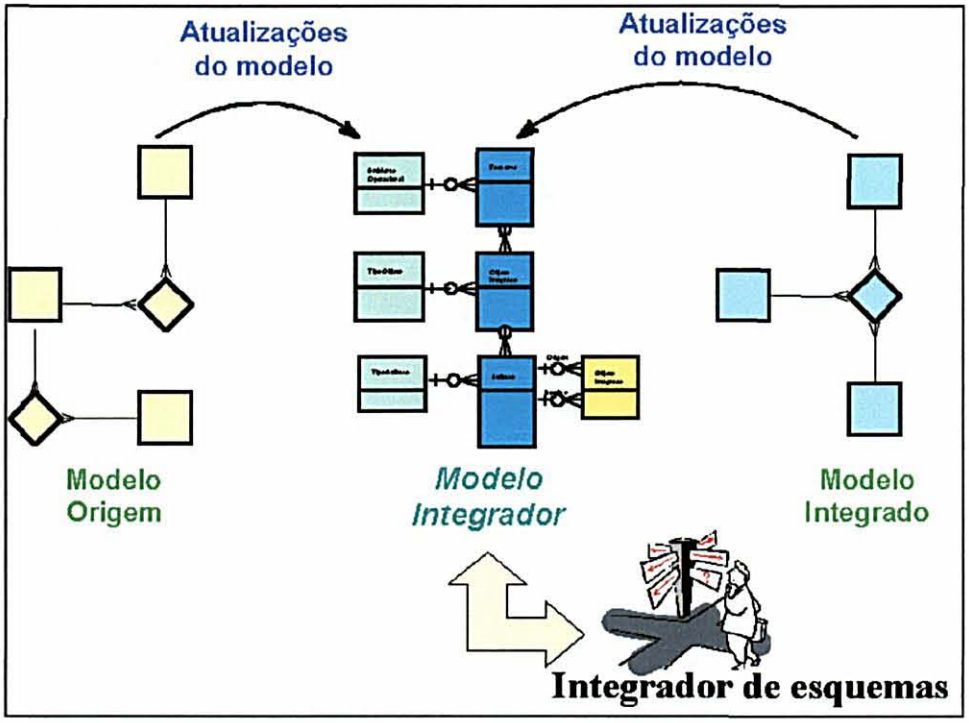
Existem dois procedimentos distintos para inserir um esquema novo no modelo integrador. O primeiro é a atualização dos cadastros do esquema, com suas tabelas e atributos no modelo integrador e o segundo a atualização dos relacionamentos entre atributos origem com os de destino.

No primeiro caso, a atualização do modelo integrador é de competência do esquema que está sofrendo alteração, seja ele o de origem ou o de destino, como mostra a FIGURA 5.

Quando há uma alteração no modelo origem e essa atualização não é informada ao modelo integrador, os arquivos com dados referentes a este modelo são rejeitados no momento da sua recepção no ambiente integrado. Essa rejeição se dá em função de que o ambiente integrado possui uma descrição do arquivo diferente daquela que está sendo recepcionada.

Para fazer o relacionamento entre atributos, que cai no segundo caso, poderíamos utilizar informações provenientes da integração dos modelos, fazendo com que os atributos já identificados sejam automaticamente relacionados. Para completar este relacionamento, deve ser utilizado o integrador de esquemas, que fará o refinamento, relacionando os atributos duvidosos e acertando possíveis relacionamentos equivocados, como mostra a FIGURA 6.

FIGURA 6 – UTILIZAÇÃO DE UM INTEGRADOR NO PROCESSO



Como passo inicial para a colocação dos atributos no modelo, o processo automático via uma ferramenta de integração de esquemas pode ser uma solução que agiliza a primeira integração. Para as manutenções subsequentes, a utilização do integrador no processo de atualização do modelo integrador, que pode ser o *Database Administrator* (DBA) ou um usuário experiente, passa a ser de muita importância. Manutenções automáticas podem ser solicitadas, mas em alguns casos o resultado pode ser imprevisível.

Suponhamos por exemplo que venha através de um arquivo a solicitação da exclusão de um campo do modelo origem e esta solicitação foi um engano. A retirada deste campo, pura e simples pode ocasionar erro nas informações passadas ao sistema integrado. O melhor seria que o Integrador confirmasse as retiradas de campos em que já exista relacionamento com atributos do modelo consolidado.

Outro problema a ser analisado com cuidado é a solicitação de manter cópias de como o modelo foi integrado nos sistemas de origem.

Esta replicação de dados pode gerar bases desatualizadas em diversos lugares, seja por problemas de comunicação, seja por erros em processos. Isto acaba engessando o modelo integrador pois manutenções em suas tabelas devem ser feitas em todas as bases replicadas e muitas vezes os tempos de atualização dos sistemas podem não coincidir.

3.2 MODELO DE DADOS DO INTEGRADOR

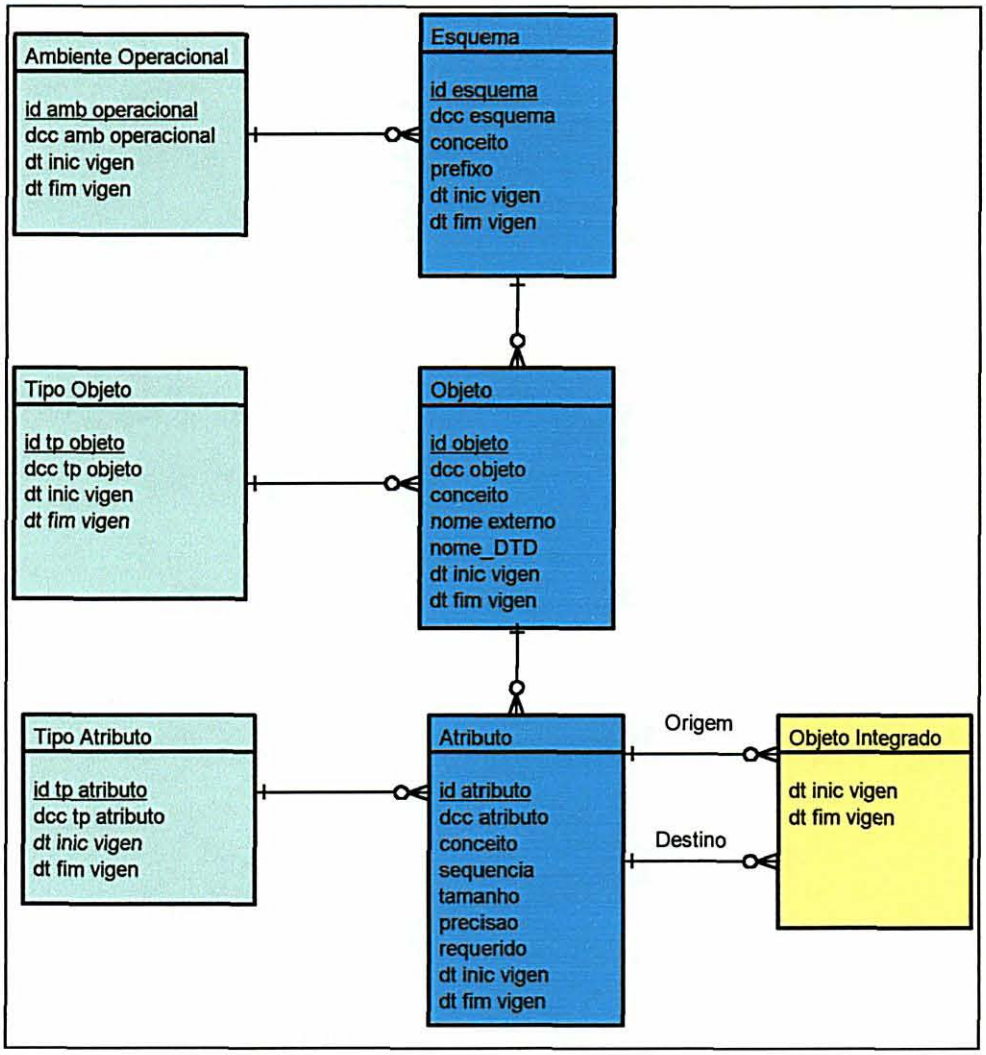
O modelo integrador foi criado para armazenar as informações de quais eram as características individuais de cada modelo antes da integração, como ficou o modelo integrado, qual a relação e como foram mapeados os atributos origem para o modelo consolidado. Este modelo é o ponto central da ferramenta desenvolvida para auxiliar no processo de validação dos arquivos recepcionados no ambiente do *data warehouse*, base do presente estudo.

O modelo integrador apresentado na FIGURA 7, permite que os esquemas participantes da integração atualizem as informações sobre si próprios, identificando a qual esquema os dados se referem. Para cada esquema deve-se também informar quais são os objetos que estão sendo integrados, tais como arquivos, tabelas ou visões.

A complementação final das informações dos esquemas participantes são os atributos. Todos os atributos que serão enviados pelos aplicativos origem, também chamados de "sistemas transacionais", devem ser informados, porque é com base nestas informações que serão geradas as DTD para validar os arquivos com os dados.

A DTD do aplicativo origem terá toda a definição de como o arquivo está estruturado, como ele deve ser lido e o que deve ser validado.

FIGURA 7 – ESQUEMA INTEGRADOR



Após a inclusão de todos os esquemas no modelo integrador, temos que realizar o mapeamento dos dados indicando para cada atributo do modelo integrado quais são as suas fontes.

A geração dos dados para este mapeamento é a parte mais complexa da integração, sendo a própria integração de esquema em si. Conseguir identificar com exatidão as fontes dos atributos é um trabalho que pode ser auxiliado por metodologias e ferramentas de integração.

Com as informações de quais são as fontes de um determinado atributo no ambiente consolidado, é possível identificar, no momento da

recepção dos dados dos aplicativos origem, qual atributo deve receber aquela informação.

Para armazenar os dados sobre os esquemas fornecedores de informação e sobre os esquemas consolidados, as sete entidades definidas no modelo são suficiente para atender ao estudo de caso.

Este modelo permite que novas entidades sejam anexadas para gerar outras informações, como por exemplo um controle de arquivos processados, com dados sobre qual objeto já foi processado, de qual esquema, quantos registros foram carregados, em quanto tempo, etc.

O modelo integrador compreende três grupos de entidades, as quais possuem as seguintes características:

a) validação dos tipos de informação, que são atualizados principalmente pela figura do integrador de esquemas. Entidades Ambiente Operacional, Tipo Objeto e Tipo Atributo;

b) grupo da estrutura das principais entidades do esquema, envolvendo as entidades Esquema, Objeto e Atributo, que serão a base para toda a integração de esquemas;

c) entidade Objeto Integrado, a qual cabe fazer todo o relacionamento e mapeamento dos atributos origem com os destino, recebendo dados de forma automática, através de um processo que lê as estruturas a partir de arquivos fornecidos pelos aplicativos origem, ou de forma manual através da intervenção do integrador de esquemas;

Estes conjuntos de entidades trabalham de forma integrada, portanto as regras de integridade referencial devem ser consideradas e respeitadas.

Isto significa que uma informação para ser incluída na entidade Objeto só poderá ser realizada caso exista um registro na entidade Tipo Objeto e de outro na entidade Esquema que garantam o relacionamento com o registro da entidade Objeto. A entidade Esquema por sua vez só

poderá conter dados após serem inseridos registros válidos na entidade Ambiente Operacional.

Qualquer tentativa de violação destas restrições, é rejeitada pelo gerenciador de banco de dados, pois estas regras foram implementadas diretamente no banco de dados.

As entidades utilizadas no modelo são assim descritas:

a) Entidade: Ambiente Operacional

Informações sobre os ambientes em que o esquema está baseado, como por exemplo *mainframe*, servidor Unix, servidor NT, etc

- Atributo: id amb operacional

  Identificação do ambiente operacional em que o esquema está baseado.

- Atributo: dcc amb operacional

  Descrição do ambiente operacional em que o esquema está baseado.

- Atributo: dt inic vigen

  Data de início de vigência desta informação na base de dados.

- Atributo: dt fim vigen

  Data de fim de vigência desta informação na base de dados.

\# Chave primária: id amb operacional

b) Entidade: Tipo Objeto

Tipos de objetos que estão sendo integrados, tais como tabelas, arquivos, visões, etc.

- Atributo: id tp objeto

  Identificação do tipo do objeto que está sendo inserido no modelo.

- Atributo: dcc tp objeto

Descrição do tipo de objeto que está sendo inserido no modelo.

- Atributo: dt inic vigen

  Data de início de vigência desta informação na base de dados.

- Atributo: dt fim vigen

  Data de fim de vigência desta informação na base de dados.

Chave primária: id tp objeto

c) Entidade: Tipo Atributo

Tipos de atributos que estão sendo integrados, tais como *number, varchar2, int, long, string,* etc. A identificação exata do tipo de atributo de cada esquema permite que sejam feitas análises de compatibilidade entre atributos para verificar se não poderão existir problemas de compatibilidade durante o processo de integração e carga.

- Atributo: id tp atributo

  Identificação do tipo de atributo que está sendo inserido no modelo.

- Atributo: dcc tp atributo

  Descrição do tipo de atributo que está sendo inserido no modelo.

- Atributo: dt inic vigen

  Data de início de vigência desta informação na base de dados.

- Atributo: dt fim vigen

  Data de fim de vigência desta informação na base de dados.

Chave primária: id tp atributo

d) Entidade: Esquema

Esquema que o modelo a ser integrado representa. Ex: Sistema conta corrente, sistema de biblioteca, ERP, DW, etc.

- Atributo: id esquema

  Identificação do esquema que está sendo inserido no modelo. Cada esquema deve ser único dentro do modelo de integração.

- Atributo: dcc esquema

  Descrição resumida do esquema que está sendo inserido no modelo.

- Atributo: conceito

  Conceituação detalhada do esquema.

- Atributo: prefixo

  Prefixo externo que todos os objetos terão quando enviarem dados para o modelo integrado. Será utilizado como diferenciador de objetos que tenham o mesmo nome externo dentro de esquemas distintos.

- Atributo: dt inic vigen

  Data de início de vigência desta informação na base de dados.

- Atributo: dt fim vigen

  Data de fim de vigência desta informação na base de dados.

Chave primária: id esquema

Chave estrangeira: id amb operl → Ambiente Operacional

e) Entidade: Objeto

Cada tabela, arquivo ou outro tipo de objeto que está sendo integrado no modelo.

- Atributo: id objeto

Identificação do objeto que está sendo inserido no modelo. Cada objeto deve ser único dentro do esquema ao qual ele pertence.

- Atributo: dcc objeto

  Descrição resumida do esquema que está sendo inserido no modelo.

- Atributo: conceito

  Conceituação detalhada do objeto, agregando informações que permitam identificar exatamente quais as informações que estão contidas no objeto.

- Atributo: nome externo

  Nome com o qual o objeto será reconhecido no sistema operacional durante o processo de recepção e tratamento para carga. Todo objeto deverá ter como prefixo o valor definido na entidade esquema, atributo prefixo.

- Atributo: nome DTD

  Nome da DTD (Definição de Tipo de Dado) com que o objeto será validado no momento da recepção dos dados a serem carregados.

- Atributo: dt inic vigen

  Data de início de vigência desta informação na base de dados.

- Atributo: dt fim vigen

  Data de fim de vigência desta informação na base de dados.

Chave primária: id esquema → esquema
  id objeto

Chave estrangeira: id esquema → esquema

f) Entidade: Atributo

Cada atributo que compõe o objeto e que poderá ou não ser integrado. A integração do atributo estará indicada na entidade Objeto Integrado, sendo portanto necessária a solicitação de integração via arquivo XML ou através da intervenção do integrador de esquemas.

- Atributo: id atributo

  Identificação do atributo que está sendo inserido no modelo. Cada atributo deve ser único no objeto, que por sua vez cada objeto deve ser único dentro do esquema ao qual ele pertence.

- Atributo: dcc atributo

  Descrição resumida do atributo.

- Atributo: conceito

  Conceito detalhado do atributo, incluindo informações tais como fórmulas, agregações e outras que venham a ser relevantes para quem estiver analisando o atributo com intuito de utilizá-lo.

- Atributo: seqüência

  Número identificador da posição seqüencial do atributo dentro do objeto. Exemplo: 3 → significa o terceiro atributo dentro do objeto.

- Atributo: tamanho

  Determina o tamanho que o atributo terá. Para dados numéricos identifica o tamanho exato da parte inteira, não importando quantos dígitos tem na parte fracionária, pois esta informação estará no campo precisão.

- Atributo: precisão

  Para os atributos numéricos, identifica quantas casas decimais deverão ser consideradas.

- Atributo: requerido domínio: verdadeiro ou falso

  Indica se o atributo não poderá ter valores nulos como conteúdo.

- Atributo: dt inic vigen

  Data de início de vigência desta informação na base de dados.

- Atributo: dt fim vigen

  Data de fim de vigência desta informação na base de dados.

Chave primária: id esquema → objeto
id objeto → objeto
id atributo

Chaves estrangeiras: id esquema → objeto
id objeto → objeto

g) Entidade: Objeto Integrado

Cada esquema que participar de alguma integração deverá informar quais objetos e seus respectivos atributos estão sendo utilizados nos mapeamentos. Deverão ser informados tanto os esquema/objeto/atributo origem quanto destino.

- Atributo: dt inic vigen

  Data de início de vigência desta informação na base de dados.

- Atributo: dt fim vigen

  Data de fim de vigência desta informação na base de dados.

Chave primária: id esquema (origem) → atributo
id objeto (origem) → atributo
id atributo (origem) → atributo
id esquema (destino) → atributo
id objeto (destino) → atributo

id atributo (destino) → atributo

Chaves estrangeiras: id esquema (origem) → esquema
id objeto (origem) → objeto
id atributo (origem) → atributo
id esquema (destino) → esquema
id objeto (destino) → objeto
id atributo (destino) → atributo

3.3 PROCESSO DE INTEGRAÇÃO DO MODELO

A atualização do modelo é feita através da entrada de dados diretamente utilizando um aplicativo e por um processo automático de recepção dos arquivos para manutenção dos dados.

Para o primeiro caso foi desenvolvido um aplicativo que gerencia todas as entidades, permitindo que o integrador de esquemas faça todas as manutenções necessárias para o correto funcionamento do modelo.

Algumas entidades são atualizadas somente por este administrador, para que se possa garantir a integridade das informações que serão recepcionadas. Estas entidades são: Ambiente Operacional, Tipo Objeto e Tipo Atributo. Caso seja necessário a criação de um novo tipo de atributo, por exemplo, o administrador faz a inclusão, após validar se é um tipo que realmente será utilizado e faz sentido estar na tabela de consistências.

Para resolver o problema de atualização automática, foi escolhido como procedimento receber um arquivo contendo todas as informações sobre o processo de integração.

Este arquivo deve estar dentro do padrão XML. A escolha por XML objetiva validar se a utilização deste padrão, publicado por um órgão independente de fornecedores, realmente pode trazer vantagens ao processo ou o torna mais complicado.

As regras para gerar dados para o modelo integrado, foram definidas através de um protocolo que indica qual o tipo de operação deve ser executada, como por exemplo a inclusão, alteração ou exclusão de atributo. O processo que recebe o arquivo faz a leitura e gera um comando SQL equivalente, que deve ser executado no banco de dados. Este protocolo foi definido na forma de uma DTD, que é utilizada para validar o arquivo recebido.

Para que a DTD pudesse ser mais abrangente, foi definida para aceitar qualquer uma das três alternativas, ou seja, inclusão, alteração e exclusão dos elementos da integração, como mostra a FIGURA 8.

FIGURA 8 – DTD DO MODELO DE INTEGRAÇÃO

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Integração ( Esquema+ | Objeto+ | Atributo+ )+>

<!-- Seção Esquema -->
<!ELEMENT Esquema (          id_esquema ,
                             descrição_esquema? ,
                             conceito_esq?,
                             prefixo?,
                             data_início_vig_esq ,
                             data_fim_vig_esq?,
                             id_ambiente_operacional?      )>

<!ATTLIST Esquema            função (incluir | excluir | alterar)  #REQUIRED >
<!ELEMENT id_esquema                 (#PCDATA ) >
<!ELEMENT descrição_esquema          (#PCDATA ) >
<!ELEMENT conceito_esq               (#PCDATA ) >
<!ELEMENT prefixo                    (#PCDATA ) >
<!ELEMENT data_início_vig_esq        (#PCDATA ) >
<!ELEMENT data_fim_vig_esq           (#PCDATA ) >
<!ELEMENT id_ambiente_operacional (#PCDATA ) >

<!-- Seção Objeto -->
<!ELEMENT Objeto (           id_esquema_obj,
                             id_objeto ,
                             descrição_objeto? ,
                             conceito_obj?,
                             nome_externo?,
                             nome_DTD?,
                             data_início_vig_obj ,
                             data_fim_vig_obj?,
                             id_tipo_objeto? )>

<!ATTLIST Objeto             função (incluir | excluir | alterar)  #REQUIRED >
<!ELEMENT id_esquema_obj             (#PCDATA ) >
<!ELEMENT id_objeto                  (#PCDATA ) >
```

```
<!ELEMENT descrição_objeto          (#PCDATA ) >
<!ELEMENT conceito_obj              (#PCDATA ) >
<!ELEMENT nome_externo              (#PCDATA ) >
<!ELEMENT nome_DTD                  (#PCDATA ) >
<!ELEMENT data_início_vig_obj       (#PCDATA ) >
<!ELEMENT data_fim_vig_obj          (#PCDATA ) >
<!ELEMENT id_tipo_objeto            (#PCDATA ) >

<!-- Seção  Atributo -->
<!ELEMENT Atributo (            id_esquema_atr,
                                id_objeto_atr ,
                                id_atributo,
                                descrição_atributo? ,
                                conceito_atr?,
                                seqüência?,
                                tamanho?,
                                precisão?,
                                requerido?,
                                data_início_vig_atr? ,
                                data_fim_vig_atr?,
                                tipo_atributo?          )>

<!ATTLIST Atributo              função (incluir | excluir | alterar)  #REQUIRED >
<!ELEMENT id_esquema_atr            (#PCDATA ) >
<!ELEMENT id_objeto_atr             (#PCDATA ) >
<!ELEMENT id_atributo               (#PCDATA ) >
<!ELEMENT descrição_atributo        (#PCDATA ) >
<!ELEMENT conceito_atr              (#PCDATA ) >
<!ELEMENT seqüência                 (#PCDATA ) >
<!ELEMENT tamanho                   (#PCDATA ) >
<!ELEMENT precisão                  (#PCDATA ) >
<!ELEMENT requerido                 (#PCDATA ) >
<!ELEMENT data_início_vig_atr       (#PCDATA ) >
<!ELEMENT data_fim_vig_atr          (#PCDATA ) >
<!ELEMENT id_tipo_atributo          (#PCDATA ) >
```

Esta DTD mostra que o arquivo pode conter dados sobre o esquema, objeto ou atributo e que eles podem aparecer em qualquer ordem e em qualquer seqüência. Isto está definido no comando:

<!ELEMENT Integração (Esquema+ | Objeto+ | Atributo+)+>

Poderia ter sido definido que para cada elemento esquema, objeto e atributo existisse pelo menos uma ocorrência, obrigando assim que esquema, objeto e atributo, estivessem sempre no arquivo. Porém não seria necessário que fossem do mesmo esquema ou objeto. O comando seria o seguinte:

<!ELEMENT Integração (Esquema+, Objeto+, Atributo+)>

Isto nos mostra que a forma do arquivo depende de padrões que são aplicados e da escolha pura e simples que o arquiteto do sistema faz.

Outra solução que poderia ter sido escolhida seria a de montar uma estrutura hierárquica, com a seqüência Esquema, Objeto e Atributo. Neste tipo de arquitetura, quando alguma operação fosse executada em um atributo, toda a hierarquia deveria ser informada. Este procedimento geraria mais dados para serem analisados e informados, mas também funcionaria.

A FIGURA 9 mostra um exemplo de alteração de um registro de esquema que já se encontra na base de dados e que está de acordo com a DTD da FIGURA 8.

FIGURA 9 – ALTERAÇÃO DE DADOS DE UM ESQUEMA

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Integração_Esquema SYSTEM "Integra_Esquema.dtd">

<Integração>
  <Esquema função="alterar">
    <id_esquema>Poupança</id_esquema>
    <descrição_esquema>Sistema de controle da Super Popança</descrição_esquema>
    <conceito_esq/>
    <prefixo/>
    <data_inicio_vig_esq>01/07/1999</data_inicio_vig_esq>
    <data_fim_vig_esq/>
    <id_ambiente_operacional/>
  </Esquema>
</Integração>
```

A

FIGURA **10** mostra a ocorrência de todas as entidades, criando um esquema, um objeto e diversos atributos.

FIGURA 10 – MANUTENÇÃO DE ELEMENTOS NO MODELO

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Integração_Esquema SYSTEM "Integra_Esquema.dtd"><Integração>
```

```
<Esquema Função="incluir">
    <!--=====INFORMAÇÕES DE ESQUEMA=====-->
    <id_esquema>Seguro</id_esquema>
    <descrição_esquema>Seguros de vida e saúde</descrição_esquema>
    <conceito_esq>Informações sobre seguros de pessoas, especificamente de vida e
saúde.</conceito_esq>
    <prefixo>PSS</prefixo>
    <data_início_vig_esq>01/02/2001</data_início_vig_esq>
    <data_fim_vig_esq/>
    <id_ambiente_operacional>Unix</id_ambiente_operacional>
  </Esquema>


  <Esquema Função="alterar">
    <!--=====INFORMAÇÕES DE ESQUEMA=====-->
    <id_esquema>Poupança</id_esquema>
    <descrição_esquema>Sistema de controle da Super Popança para pessoa física,
        juridica</descrição_esquema>
    <conceito_esq/>
    <prefixo>SPP</prefixo>
    <data_início_vig_esq>01/02/2001</data_início_vig_esq>
    <data_fim_vig_esq/>
    <id_ambiente_operacional>Unix</id_ambiente_operacional>
  </Esquema>


  <Esquema Função="incluir">
    <!--=====INFORMAÇÕES DE ESQUEMA=====-->
    <id_esquema>Data Warehouse</id_esquema>
    <descrição_esquema>Data Warehouse Corporativo</descrição_esquema>
    <conceito_esq>Data Warehouse Corporativo, abrangendo informações de
        Empréstimos, Investimentos e Conta Corrente</conceito_esq>
    <prefixo>DWH</prefixo>
    <data_início_vig_esq>01/02/2001</data_início_vig_esq>
    <data_fim_vig_esq/>
    <id_ambiente_operacional>Unix</id_ambiente_operacional>
  </Esquema>


  <Objeto Função="incluir">
    <!--=====INFORMAÇÕES DE OBJETO=====-->
    <id_esquema_obj>Seguro</id_esquema_obj>
    <id_objeto>Apólice Pessoa</id_objeto>
    <descrição_objeto>Apólices de Pessoas de vida e saúde</descrição_objeto>
    <conceito_obj>Apólice que uma pessoa pode contratar para segurar a vida e plano
de saúde</conceito_obj>
    <nome_externo>APPSSVDSD.XML</nome_externo>
    <nome_DTD>APPESSOAS.DTD</nome_DTD>
    <data_inicio_vig_obj>01/02/2001</data_inicio_vig_obj>
    <data_fim_vig_obj>31/12/2050</data_fim_vig_obj>
    <id_tipo_objeto>Tabela</id_tipo_objeto>
  </Objeto>


  <Objeto Função="incluir">
    <!--=====INFORMAÇÕES DE OBJETO=====-->
```

```
    <id_esquema_obj>Data Warehouse</id_esquema_obj>
    <id_objeto>Objeto Segurado</id_objeto>
    <descrição_objeto>Objetos que podem ser segurados</descrição_objeto>
    <conceito_obj>Coisas, Pessoas ou Locais que podem ser objeto de um
seguro</conceito_obj>
    <nome_externo>OBJSEG.XML</nome_externo>
    <nome_DTD>OBJSEG.DTD</nome_DTD>
    <data_inicio_vig_obj>01/02/2001</data_inicio_vig_obj>
    <data_fim_vig_obj/>
    <id_tipo_objeto>Tabela</id_tipo_objeto>
  </Objeto>


  <Atributo Função="incluir">
    <!--=====INFORMAÇÕES DE ATRIBUTO=====-->
    <id_esquema_atr>Seguro</id_esquema_atr>
    <id_objeto_atr>Apólice Pessoa</id_objeto_atr>
    <id_atributo>Companhia</id_atributo>
    <descrição_atributo>Nome da companhia seguradora</descrição_atributo>
    <conceito_atr>Nome da companhia que é a lider no seguro</conceito_atr>
    <seqüência>01</seqüência>
    <tamanho>50</tamanho>
    <precisão>0</precisão>
    <requerido>yes</requerido>
    <data_inicio_vig_obj>01/02/2001</data_inicio_vig_obj>
    <data_fim_vig_obj/>
    <id_tipo_atributo>varchar2</id_tipo_atributo>
  </Atributo>


  <Atributo Função="alterar">
    <!--=====INFORMAÇÕES DE ATRIBUTO=====-->
    <id_esquema_atr>Seguro</id_esquema_atr>
    <id_objeto_atr>Apólice Pessoa</id_objeto_atr>
    <id_atributo>Companhia</id_atributo>
    <descrição_atributo/>
    <conceito_atr>Nome da companhia que emite a apólice, podendo ou não ser a
líder</conceito_atr>
    <seqüência/>
    <tamanho/>
    <precisão/>
    <requerido/>
    <data_inicio_vig_obj>01/02/2001</data_inicio_vig_obj>
    <data_fim_vig_obj/>
    <id_tipo_atributo/>
  </Atributo>


  <Atributo Função="incluir">
    <!--=====INFORMAÇÕES DE ATRIBUTO=====-->
    <id_esquema_atr>Seguro</id_esquema_atr>
    <id_objeto_atr>Apólice Pessoa</id_objeto_atr>
    <id_atributo>Num_Apolice</id_atributo>
    <descrição_atributo>Número da apólice</descrição_atributo>
    <conceito_atr>Número sequencial identificador da apólice dentro do
ramo</conceito_atr>
    <seqüência>02</seqüência>
```

```
    <tamanho>15</tamanho>
    <precisão>0</precisão>
    <requerido>yes</requerido>
    <data_inicio_vig_obj>01/02/2001</data_inicio_vig_obj>
    <data_fim_vig_obj/>
    <id_tipo_atributo>number</id_tipo_atributo>
  </Atributo>


  <Atributo Função="incluir">
    <!--=====INFORMAÇÕES DE ATRIBUTO=====-->
    <id_esquema_atr>Seguro</id_esquema_atr>
    <id_objeto_atr>Apólice Pessoa</id_objeto_atr>
    <id_atributo>Vlr_Premio_Pg</id_atributo>
    <descrição_atributo>Valor do prêmio pago pelo cliente</descrição_atributo>
    <conceito_atr>Valor do prêmio pago pelo cliente pelo seguro contratado. Valor total,
não importando se houve parcelamento.</conceito_atr>
    <seqüência>10</seqüência>
    <tamanho>15</tamanho>
    <precisão>2</precisão>
    <requerido>yes</requerido>
    <data_inicio_vig_obj>01/03/2001</data_inicio_vig_obj>
    <data_fim_vig_obj/>
    <id_tipo_atributo>number</id_tipo_atributo>
  </Atributo>

</Integração>
```

Na

FIGURA **10** temos um exemplo de dois esquemas origem, Seguro e Poupança, e um esquema destino, Data warehouse. Isto mostra que a DTD pode ser utilizada para todos os esquemas envolvidos na integração.

Para o processo de relacionar os atributos origem com os de destino, foi definida a DTD da FIGURA 11, que exige que seja informado pelo menos um relacionamento no arquivo.

FIGURA 11 – DTD PARA RELACIONAMENTO DE ATRIBUTOS

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT Integração_Atributos (Relacionamento+)

<!-- Seção  Relacionamento de Atributos -->
<!ELEMENT Relacionamento (          id_esquema_origem    ,
                                    id_objeto_origem     ,
                                    id_atributo_origem   ,
                                    id_esquema_destino   ,
                                    id_objeto_destino    ,
```

```
                                    id_atributo_destino      )>

<!ATTLIST Relacionamento            Função (incluir | excluir ) #REQUIRED >
<!ELEMENT id_esquema_origem         (#PCDATA ) >
<!ELEMENT id_objeto_origem          (#PCDATA ) >
<!ELEMENT id_atributo_origem        (#PCDATA ) >
<!ELEMENT id_esquema_destino        (#PCDATA ) >
<!ELEMENT id_objeto_destino         (#PCDATA ) >
<!ELEMENT id_atributo_destino       (#PCDATA ) >
```

Um exemplo para a utilização da DTD acima, pode ser observado na FIGURA 12. Note-se que podemos colocar atributos de esquemas diferentes, o que normalmente não seria usual levando-se em consideração que cada origem seria um esquema. Mas nada impede que uma mesma fonte seja origem de mais de um esquema.

FIGURA 12 – EXEMPLO DE RELACIONAMENTO

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Integração_Relacionamento SYSTEM "Integra_Relacao.dtd">

<Integração_Relacionamento>

  <Relacionamento Função="incluir">
    <!--===== RELACIONAMENTO =====-->
    <id_esquema_origem>Seguro</id_esquema_origem>
    <id_objeto_origem>Apólice</id_objeto_origem>
    <id_atributo_origem>Companhia</id_atributo_origem>
    <id_esquema_destino>Data Warehouse</id_esquema_destino>
    <id_objeto_destino>Objeto Segurado</id_objeto_destino>
    <id_atributo_destino>Companhia</id_atributo_destino>
  </Relacionamento>


<Relacionamento Função="incluir">
    <!--===== RELACIONAMENTO =====-->
    <id_esquema_origem>Poupança</id_esquema_origem>
    <id_objeto_origem>Posição Diária</id_objeto_origem>
    <id_atributo_origem>Número Conta</id_atributo_origem>
    <id_esquema_destino>Data Warehouse</id_esquema_destino>
    <id_objeto_destino>Investimento</id_objeto_destino>
    <id_atributo_destino>Identificador Conta</id_atributo_destino>
  </Relacionamento>

</Integração_Relacionamento>
```

Uma pergunta que poderia ser feita é: Por que não foi colocada integridade referencial nos próprios arquivos?

Basicamente para não engessar o modelo e para não gerar informação desnecessária nos arquivos. Suponha que se queira fazer uma exclusão de um atributo de um objeto qualquer. Se a integridade referencial for colocada, o gerador dos dados deve informar além de um registro do atributo em questão, também uma ocorrência de Esquema e Objeto. Mas tudo isso não garante que o atributo cuja exclusão esta sendo solicitada, efetivamente existe no banco de dados integrado. Portanto, se esta validação for feita no momento da execução da solução, o resultado torna-se mais confiável.

A decisão adotada neste estudo foi a de que o controle de integridade referencial ficaria no próprio banco de dados, no processo que recebe os arquivos e faz a manutenção nas bases do sistema integrado.

Um modelo de DTD mais completo poderia utilizar o padrão XML *Schema*, que também está sendo definido pela W3C.

Através deste padrão, é possível definir quais os tipos de dados que poderíamos aceitar em cada atributo, como por exemplo *number* e *varchar2* para *Oracle*, *string* e *integer* para arquivos provenientes de uma outra fonte qualquer.

3.4 FERRAMENTA DE AUTOMATIZAÇÃO

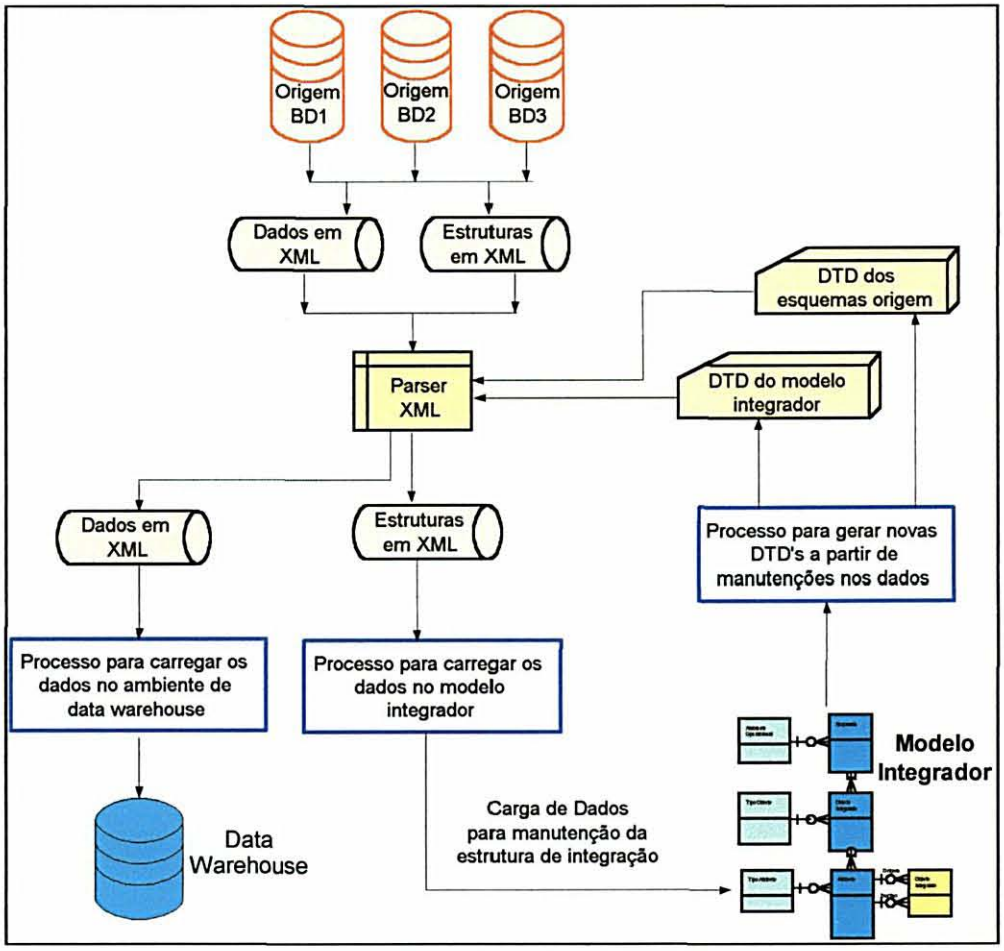
A ferramenta de automatização visa basicamente unir as definições do protocolo com os processos que efetivamente irão atualizar o banco de dados com as informações recebidas via arquivos.

O processo de atualização das estruturas é o primeiro a ser executado durante as tarefas diárias de carga. Isto porque as informações das estruturas é que serão a base para a validação dos arquivos com os dados que serão carregados no *data warehouse*.

Dois processos são os mais importantes. O primeiro é o que faz a carga dos dados nas tabelas, atualizando assim o modelo integrador. O

segundo é o que gera os arquivos com as definições das DTD's que serão utilizadas pelo analisador de arquivos XML (*parser*) no momento da recepção do arquivo para carga. A FIGURA 13 mostra estes fluxos.

FIGURA 13 – ALTERAÇÕES NO MODELO INTEGRADOR



A geração da nova DTD se faz necessária para que ela possa refletir as informações existentes na base de dados. As DTD são armazenadas em uma área específica, que é acessada pelo *parser*, com o objetivo de validar a estrutura dos arquivos que serão recebidos.

3.5 DESENVOLVIMENTO DA FERRAMENTA

O desenvolvimento da ferramenta fez uso de diversas tecnologias e linguagens de programação. A opção por *MS Visual Basic e MS Visual C* foi no sentido de usar o que havia disponível dentro do ambiente de informática onde estava sendo desenvolvido o estudo.

A linguagem utilizada para desenvolver a ferramenta foi a *MS Visual Basic* versão 6 (VB6). O modelo foi implantado no SGBD *Oracle*, versão 8i. Para validação do arquivo que estava sendo recebido foi utilizado o *parser* para linguagem C disponibilizado pela *Oracle* e a linguagem *MS Visual C*.

Após o arquivo ser validado pelo *parser*, é então processado pelo programa em VB6 que lê o arquivo e carrega no banco *Oracle*, inserindo cada registro nas tabelas respectivas.

A interface disponibilizada para o Integrador de esquemas, foi também desenvolvida em VB6. A partir deste aplicativo, o Integrador faz as manutenções nas tabelas que fazem a consistência dos dados, ou seja nas tabelas de Ambiente Operacional, Tipo Objeto e Tipo De Atributo.

4 ESTUDO DE CASO

Para o desenvolvimento do estudo de caso, era necessário que houvesse um ambiente onde existissem aplicativos que devessem ser integrados, um ambiente consolidado a ser construído e que o transporte dos dados dos aplicativos para o ambiente consolidado fosse constante e não uma única vez, como no caso de desativação de aplicativos.

Poderíamos ter criado um ambiente de teste, totalmente fictício e controlado. Porém, como tivemos a opção de trabalhar com um ambiente real, consideramos que o protótipo seria testado de maneira mais completa.

A empresa que permitiu o desenvolvimento e teste da ferramenta é do setor bancário.

A empresa passava por um processo de criação de um ambiente consolidado, integrando muitos sistemas aplicativos das áreas de investimento, empréstimo e contas correntes.

O objetivo final era a criação de um ambiente de *data warehouse* extraindo dados de todos aplicativos das áreas citadas.

Para atingir o objetivo de montar esta base com dados consolidados foram executadas as seguintes fases:

a) levantamento de requerimentos de negócio;

   Foram feitas reuniões com os usuários da futura base consolidada para identificar as necessidades de informações e quais os sistemas deveriam fornecê-las.

b) mapeamento das fontes de dados;

   Após a definição dos sistemas fornecedores de informação, foram identificadas os atributos que estes deveriam informar ao sistema consolidador.

c) arquitetura da solução proposta;

   Dentro desta fase procurou-se a identificação dos softwares a serem utilizados, os canais de comunicação bem como as

ferramentas para transmissão de arquivos, o gerenciador de banco de dados e as linguagens a serem utilizadas.

d) criação do ambiente consolidado;

Com base nos levantamentos das necessidade e nos mapeamentos das fontes foi realizada a integração inicial dos esquemas dos aplicativos fontes para a criação do ambiente do *data warehouse.*

e) requisitos para apresentação dos dados;

Junto com os usuários do *data warehouse* foram definidos como os dados deveriam ser mostrados e formatados.

f) administração do ambiente de *data warehouse;*

Fase que iniciou com os primeiros contatos com o usuário e continuou durante todo o projeto. Era a responsável pela identificação dos metadados e pelo controle de como os esquemas permaneceriam atualizados após a implantação do ambiente consolidado.

Este trabalho encontra-se inserido nesta última fase do protótipo, pois aqui estão os controles sobre as evoluções dos esquemas origem e destino.

Com este protótipo demonstra-se a aplicação do modelo integrador na prática, controlando as evoluções dos modelos que participam da integração e a ampliação deste mesmo modelo para atender as necessidades de administração dos movimentos de dados do *data warehouse.*

4.1 CARACTERÍSTICAS DO PROTÓTIPO

A premissa inicial do protótipo era a de que este deveria refletir de forma mais próxima possível a realidade do processo de carga do ambiente de *data warehouse* de uma instituição bancária e que só deveria

carregar os dados quando o movimento completo para um dia já tivesse sido recebido.

Dentro deste conceito, o processo de recepção dos dados deveria respeitar os tempos de fechamento de movimento das informações para um determinado dia, obedecendo a regras de negócio e/ou do Banco Central do Brasil. O intervalo de tempo entre a primeira entrega de dados para o *data warehouse* e a última ficaria em um intervalo entre 24 e 48 horas.

Devido a este intervalo de tempo e ao fato da carga só iniciar após todos os arquivos de um dia terem sido recebidos, podemos caracterizar o processamento das informações como sendo assíncrona.

Os aplicativos transacionais envolvidos estavam baseados em sistemas operacionais diversos, tais como:

a) IBM/VM

b) Sun/Solaris

c) Intel/NT

Na fase *criação do ambiente consolidado* foram integrados os sistemas origem visando a criação do esquema do *data warehouse*. A partir desta definição foram revisados os mapeamentos de dados para identificar quais informações deveriam ser extraídas dos aplicativos transacionais.

A forma escolhida para transporte dos dados foi a utilização de arquivos com dados formatados segundo o padrão XML 1.0. Este procedimento ficou de acordo com as necessidades da ferramenta de controle de carga de dados.

Em relação aos dados, há basicamente dois tipos de informações recebidas no *data warehouse*:

a) informações dos cadastros básicos, tais como cliente, produtos, moedas, órgãos de negócio e tipos de empresas;

b) informações das operações financeiras, que são o movimento dos saldos dos contratos. Um contrato é, por exemplo, uma conta corrente, uma conta de poupança ou um empréstimo pessoal.

4.2 PROCESSAMENTO DA CARGA DE DADOS DO PROTÓTIPO

Todo o processo de carga foi dividido em dois movimentos, os quais foram chamados de "famílias". A primeira família envolvia os dados cadastrais e tabelas de consistência e a segunda família tratava das operações efetivamente de cada contrato do cliente.

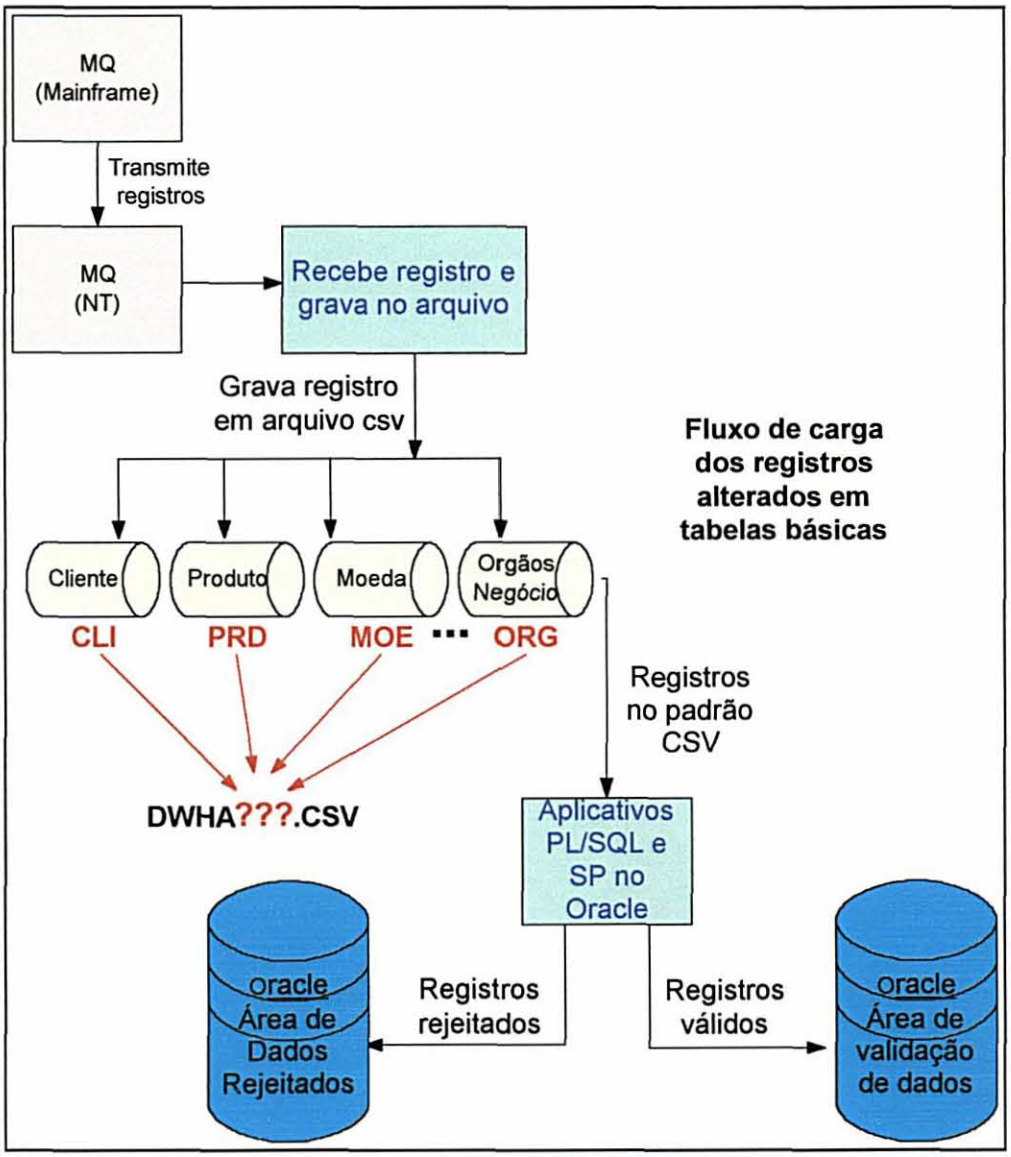
A família dos dados cadastrais trabalha com arquivos no formato *Comma Separated Value* (CSV) e a de operações com arquivos no formato XML.

Optamos pela utilização de CSV em uma família e XML em outra para analisarmos e compararmos o comportamento dos arquivos envolvidos no estudo de caso, já que teríamos um processo trabalhando no formato tradicional e outro no formato proposto.

Para a recepção de dados de atualização dos cadastros básicos, utilizados para validar as informações de negócio, foi considerado que poderia ser feita de forma ininterrupta durante as 24 horas do dia, sendo processadas pelo *data warehouse* somente no período noturno, porque sempre estarão sendo processadas com informações em D-1 (dia anterior) e as operações dos contratos em D-2 (2 dias de defasagem).

A FIGURA 14 mostra os passos para a atualização destes cadastros:

FIGURA 14 – ATUALIZAÇÃO DOS DADOS DE CADASTROS



Cada alteração feita em um cliente, moeda, cadastro de produto, de órgão de negócio ou em outro cadastro básico de interesse do *data warehouse* é automaticamente disponibilizada em um arquivo CSV no servidor do *data warehouse* para permitir o processamento no período da noite.

Para a transferência do *mainframe* para o servidor do *data warehouse* foi utilizado o software de transferência de mensagens *MQ*

Series da IBM. No nosso caso estas mensagens são registros com identificação da tabela à qual ele pertence.

Para receber a mensagem e interpretá-la, foi desenvolvido um programa em C que faz a distribuição da mensagem conforme o tipo de registro que está chegando, colocando os de clientes no arquivo de clientes, o de produtos no arquivo de produtos e assim por diante para todas as outras tabelas básicas.

Quando o processo de carga destas tabelas se inicia, são executados 18 programas em PL/SQL *Oracle*. Estes processos lêem os arquivos, validam o conteúdo do registro através da aplicação de regras de negócio e de regras de integridade referencial. Estando tudo certo, isto é, passando por todas as validações, os registros são gravados nas tabelas apropriadas.

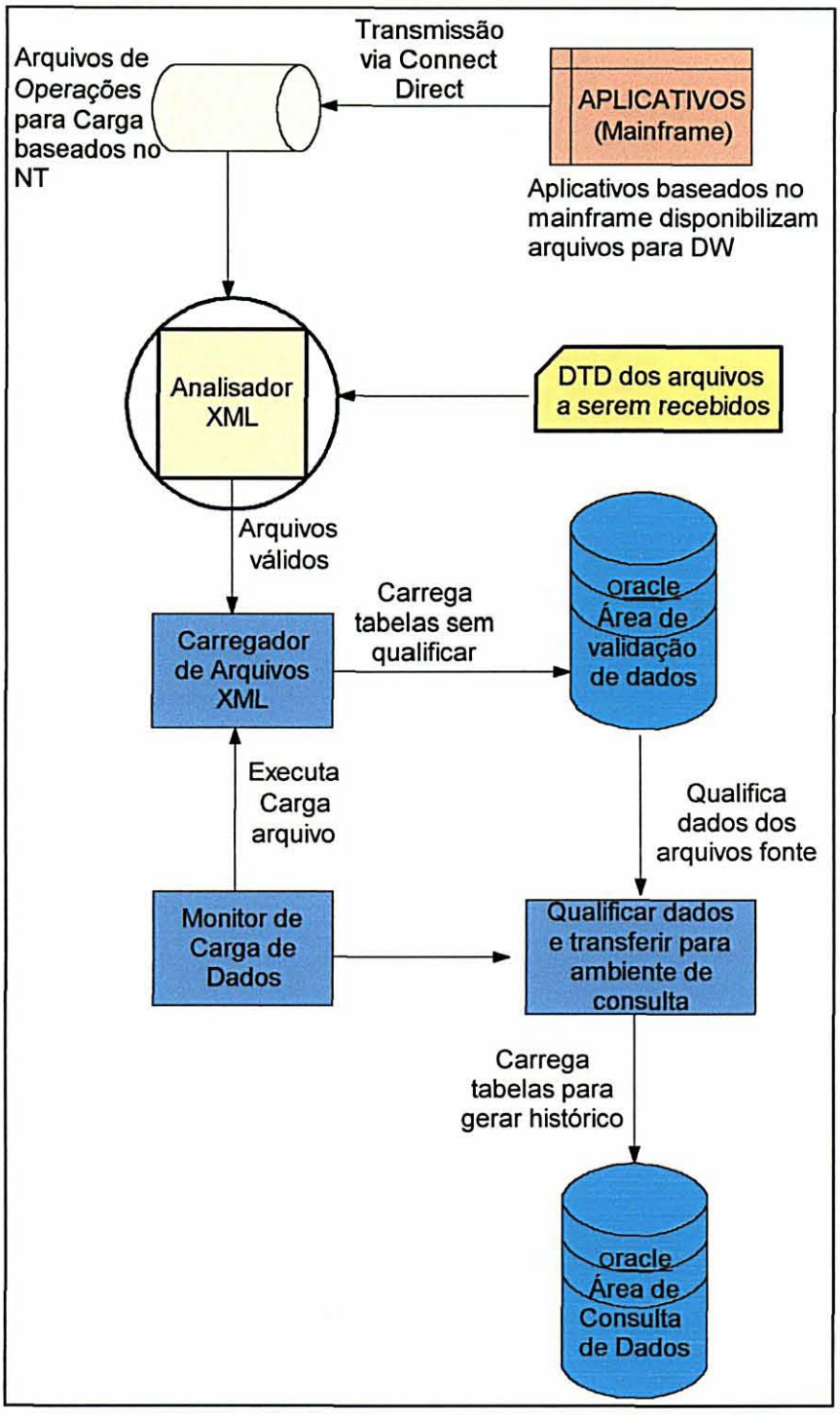
Caso exista algum problema no registro, ele é gravado em uma área de dados para registros rejeitados. Esta área funciona como um "hospital de dados", pois cada registro ali armazenado será analisado para ver quais providências devem ser tomadas para a sua regularização. Para registros com rejeições padrões são gerados relatórios a serem enviados aos gestores do projeto *data warehouse*. Exceções são analisadas de forma manual, para tentar identificar possíveis erros nos processos.

A família das operações dos clientes tem o seu processamento seguindo a premissa inicial, ou seja, só inicia após o recebimento de todos os arquivos no servidor do *data warehouse*.

A transferência dos arquivos do *mainframe* para o servidor do *data warehouse* é feita através de um software de comunicação chamado *Connect Direct*, que após transferir o arquivo chama um programa escrito em C. Este programa informa à ferramenta de administração do ambiente que um determinado arquivo, por exemplo DWH.FNB.TST.001 referente ao sistema de fundos, foi recebido.

Tendo recebido todos os arquivos do dia, inicia-se o processamento. O fluxo geral desta carga está representado na FIGURA 15. Todos os arquivos desta família estão formatados em XML.

FIGURA 15 – FLUXO CARGA DE DADOS DAS OPERAÇÕES



A utilização do padrão XML permitiu identificar erros na estrutura dos dados de forma mais rápida e simplificada, pois já no *parser* era possível verificar a formatação do arquivo.

Quando um arquivo é transformado de CSV ou *Tab Separated Value* (TSV) para XML, apresenta um crescimento na ordem 4 a 8 vezes o tamanho original em função da inserção das marcações (*tags*). Este crescimento pode ser maior ou menor, dependendo do tipo de informação que está presente no arquivo. Se forem dados predominantemente numéricos, divididos em muitas colunas, o crescimento no tamanho do arquivo tende a ser maior. Já se o registro for composto de muitos textos, que geralmente são de tamanho maior que um número, o crescimento é menor.

Para arquivos pequenos a diferença no tempo de processamento não era relevante, já para grandes arquivos, em função da inserção das *tags* identificadoras dos atributos, estes ficavam ainda maiores demorando mais para serem processados e carregados.

No formato XML, para cada atributo dentro do arquivo deve haver uma marcação informando qual o significado da informação. Por exemplo:

Formato CSV
"20000327", 2345.45, ...

Formato XML
<data_movto>20000327</data_movto>
<saldo_parcela_vencida>2345.45</saldo_parcela_vencida>
...

ou:

<data_movto>
  <ano>2000</ano>
  <mês>03</mês>
  <dia>27</dia>

</data_movto>

<saldo_parcela_vencida>2345.45</saldo_parcela_vencida>

...

Com isso notamos que cada informação agrega muitos caracteres a mais, trazendo informações da composição dos dados e consequentemente aumentando o tamanho do arquivo.

Este crescimento pode ser prejudicial caso o tamanho original já seja de um arquivo grande, como por exemplo 300 MB, e o tráfego gerado na rede seja relevante e prejudicial aos outros processos produtivos.

Porém, novas tecnologias de compressão de dados para tráfego em rede estão sendo apresentados no mercado, podendo vir a minimizar o problema de tamanho de arquivos num futuro próximo.

Há também o desenvolvimento de novas ferramentas de carga de dados nos bancos de dados, que interpretam arquivos XML, as quais estão sendo prometidas pelos seus respectivos fabricantes.

Com estas ferramentas, o processo tende a melhorar, pois com a interpretação do XML pela própria ferramenta não seria necessário um programa externo que fizesse a leitura de um arquivo em XML, interpretasse o seu conteúdo, para só depois inseri-lo nas tabelas específicas.

Os utilitários de carga de dados disponibilizados pelos fabricantes de bancos de dados, tendem a ter uma melhor performance do que programas que lêem arquivos e gravam tabelas no banco de dados.

4.3 FERRAMENTA DE CONTROLE DOS ESQUEMAS E DADOS

Toda a administração dos movimentos de arquivos de dados para o *data warehouse*, bem como o controle sobre as alterações dos esquemas envolvidos no processo de integração, ficou a cargo de uma ferramenta desenvolvida para este fim.

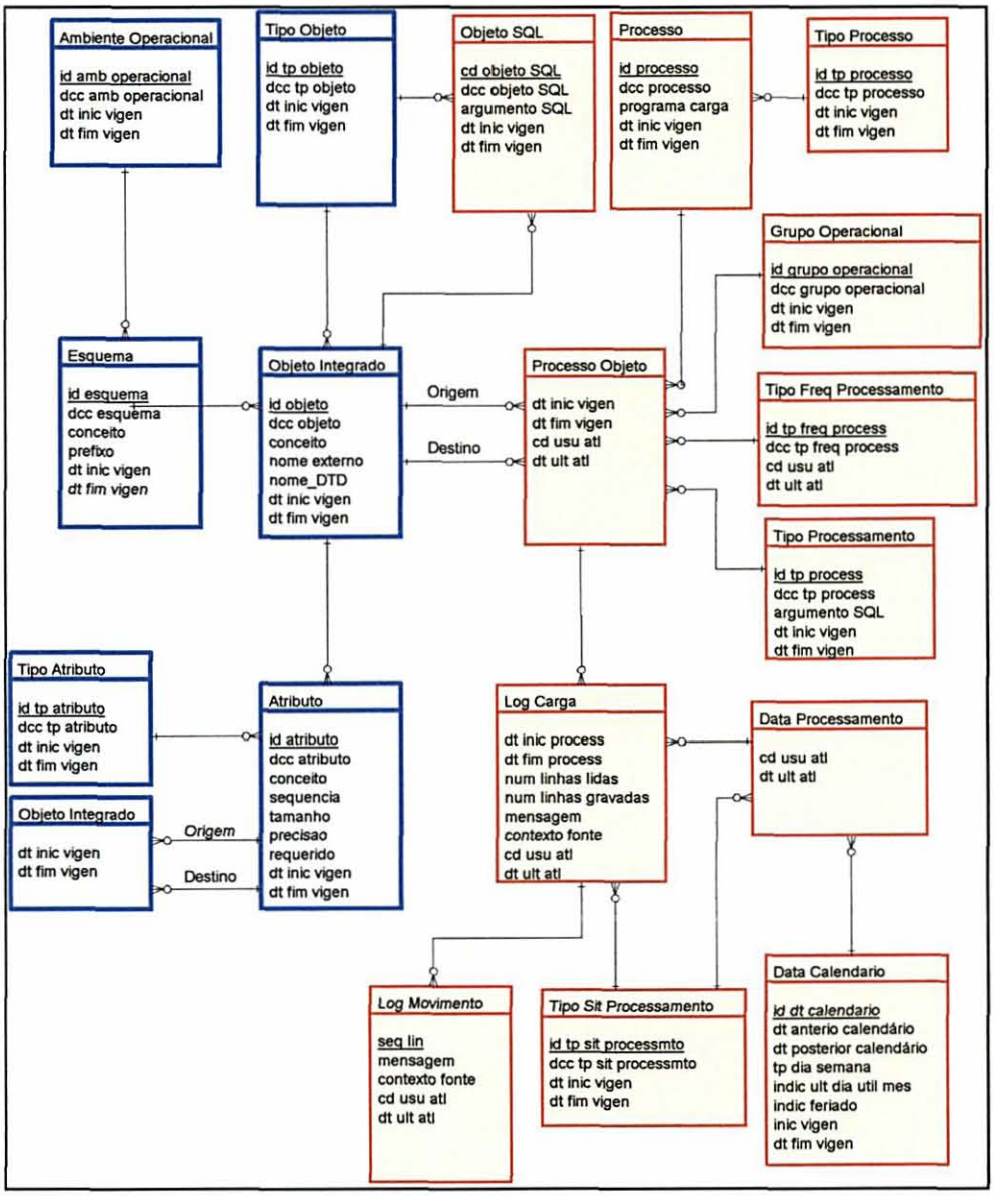
Para permitir o controle dos processos de quais arquivos já haviam sido carregados, quais os erros identificados e qual o movimento foi

rejeitado, foi projetado um esquema de dados, como mostra a FIGURA 16.

Este esquema foi criado com base no "esquema integrador" definido anteriormente. Foram agregadas entidades para permitir o controle dos processamentos e dos fluxos de dados.

Através deste "esquema de administração de cargas de dados", foi possível saber quais arquivos já haviam sido recebidos, quais foram as rejeições nos movimentos dos dias anteriores e qual o tempo de execução de cada processo de carga.

FIGURA 16 – ESQUEMA ADMINISTRAÇÃO DE CARGAS DE DADOS



As entidades agregadas ao modelo são assim descritas:

a) Entidade: Tipo Processo

Qual o tipo de execução que o processo associado irá fazer. Exemplo: carga de dados, validação de movimento ou transporte entre ambientes.

b) Entidade: Processo

Descrição do programa com a sua função básica. Qualquer programa ou *script* que seja executado deve ser cadastrado, assim teremos informação dos processos disponíveis em todo o ambiente.

c) Entidade: Objeto SQL

Comandos dinâmicos de SQL para auxiliar na manutenção do ambiente de carga. Utilizado para criação de tabelas intermediárias ou de índices em tabelas durante o processamento.

d) Entidade: Grupo Operacional

Cada conjunto de processos e objetos são classificados em grupos. Isto permite descobrir e processar grupos específicos, tais como a transferência de dados entre ambientes ou a carga de um grupo de sistemas, como os de investimento.

e) Entidade: Tipo Freq Processamento

Informa qual a periodicidade que um determinado processo deve ser executado, se diário, semanal ou eventual.

f) Entidade: Tipo Processamento

Informa se o processamento será uma carga de dados, uma limpeza de tabelas ou uma criação de objetos. Isto permite validar se o processamento que está sendo solicitado pode ser feito pelo usuário solicitante.

g) Entidade: Processo Objeto

Faz a combinação entre o processo, os objetos que este manipula, o grupo ao qual esta combinação pertence e o tipo de processamento. È a base para o controle das rotinas de execução de carga. Nenhum processo poderá ser executado se não houver uma combinação válida nesta entidade.

h) Entidade: Data Calendário

Possui todas as datas do calendário, com alguns dados tais como se é dia útil ou se é o último dia útil do mês.

i) Entidade: Tipo Sit Processamento

Informa qual a situação possíveis para o processamento de um determinado processo ou data de processamento. Aqui podem ser encontrados valores com Execução Ok, Execução Cancelada, Pendente de Execução, entre outras.

j) Entidade: Data Processamento

Informa a data que foi processada para um determinado conjunto de processo objeto e qual o tipo de situação em que se encontra.

k) Entidade: Log Carga

Para cada conjunto de processo objeto é registrado a situação final do processamento.

l) Entidade: Log Movimento

Caso exista uma rejeição em um movimento ou em parte dele, todos os registros afetados são guardados nesta entidade para que seja possível identificar qual o erro e como estava o registro no momento da inclusão na tabela.

Com a inserção dos dados das integrações nas tabelas do modelo, foi possível gerar as DTD's que são utilizadas na validação do arquivo quando este vai ser carregado.

Os programas necessários para inserir informações no modelo foram desenvolvidos em *Oracle* PL/SQL, e armazenados como procedimentos (*stored procedure*) no banco de dados *Oracle*.

Os monitores de todo o processamento de carga dos arquivos e que são os responsáveis pelas chamadas dos procedimentos no banco de dados *Oracle*, foram desenvolvidos em *Visual Basic* versão 6.0.

Para inserir dados nos cadastros que suportam o modelo de controle foi utilizada também a linguagem *Visual Basic* versão 6.0, fazendo acesso direto ao banco de dados *Oracle* versão 8i release 2.

4.4 RESULTADOS DO ESTUDO DE CASO

A avaliação dos resultados deve ser analisada tendo em vista dois pontos de vista. O primeiro é pelo controle da evolução dos modelos, com o auxilio do "modelo integrador". O segundo é na utilização de XML como padrão nos arquivos transportados dos aplicativos origem para o *data warehouse*.

A ferramenta de administração de cargas atingiu os objetivos de dar informações sobre o andamento dos processos e sobre a qualidade dos dados do *data warehouse*.

A ferramenta estava baseada no *modelo integrador* e em outras entidades adicionadas a ele para permitir o controle sobre as cargas. As informações obtidas a partir deste modelo permitiram identificar problemas nos dados enviados pelos aplicativos origem.

A utilização de XML como padronização dos arquivos envolvidos nos transporte de dados para o *data warehouse* mostrou-se muito vantajoso porque havia a validação deste na sua recepção através da utilização do *parser* e das DTD's.

Outro ponto positivo foi a possibilidade de verificar os dados no seu formato original, como enviado pelo aplicativo origem, através da utilização de um *browser*.

O maior problema enfrentado foi no momento das cargas dos arquivos no banco de dados. O tempo de processamento foi muito mais lento que a utilização de um utilitário de carga do próprio banco.

Para arquivos com 200.000 registros, a diferença foi de 3 minutos para o utilitário para 75 minutos para o programa em C. Os utilitários normalmente são muito mais rápidos que quaisquer programas, em função da forma de carga, que é otimizada ao máximo. Acreditamos que

com a liberação por parte dos fabricantes de bancos de dados de utilitários que leiam diretamente arquivos em XML, os tempos de cargas possam ser reduzidos para algo muito próximo aos que hoje são alcançados com arquivos tradicionais.

A diferença de tempo pode ter muitas razões, dentre elas podemos citar a parametrização do banco, que estava preparado para ser rápido em consultas dos usuários do *data warehouse*. Outro impacto era que o programa deveria interpretar o registro em XML e inseri-lo no banco, com as *constraints* ativas. Isto naturalmente prejudica a performance do processo.

O resultado final é que a ferramenta se comportou conforme esperado e o XML mostrou que agrega muito pelo fato de ser um descritor de dados. Melhorando a performance do processo de carga, esta solução pode realmente vir a ser aplicada nos processos produtivos de execução de sistemas das empresas.

5 CONCLUSÃO

O processo natural de informatização das empresas levou a geração de dados de forma dispersa, em diversos ambientes operacionais, variando o hardware e o software.

Tais dispersões de dados geraram a necessidade de pesquisa na área de sistemas integrados que sejam consolidadores de informações, sendo um exemplo do resultado destas pesquisas os ambientes de *data warehouse.*

Um dos problemas básicos em ambientes integrados e consolidados é fazer o transporte automático de dados de um ponto X para um ponto Y.

O transporte através de arquivos tipo *valores separados por vírgula (CSV)* ou *valores separados por tabulação (TSV),* são relativamente fáceis de implementar, mas exigem um esforço muito grande na manutenção e organização destes. Informações de como foram definidos, qual o significado de cada coluna, quais as fontes dos dados ou qual a hierarquia existente entre os dados, muitas vezes são perdidas, gerando transtornos e problemas para as áreas que irão trabalhar com estes pacotes de dados.

Com a utilização de XML, os arquivos puderam ser gerados com a definição explícita do significado da informação nele contida, vindo a facilitar a sua utilização e validação. As definições dos campos dos arquivos tornaram-se os metadados da aplicação.

Quando analisamos uma integração no âmbito de um único centro de processamento em uma empresa, os ganhos do XML podem parecer não justificáveis se forem somente para transportar arquivos entre aplicativos e um modelo integrado. Mas se analisarmos a integração dentro de um contexto de uma empresa com múltiplas filiais com processamentos independentes, ou um grupo de instituições montando uma base de dados federada, os ganhos do XML justificam o preço do tamanho dos arquivos.

Em relação aos gerenciadores de banco de dados, os grandes fornecedores, entre eles a *IBM, Oracle e Microsoft*, já estão testando versões de gerenciadores que manipulam diretamente dados no formato XML. Consultas via comandos em SQL são executadas e o resultado gerado em XML. Testes também estão sendo feitos com ferramentas que fazem a leitura de arquivos em XML e inserem dados diretamente no banco de dados. Isto significa que processos que hoje são complexos e exigem muita programação, em um futuro próximo poderá ser facilitado pelos próprios gerenciadores de banco de dados e pelos fornecedores de softwares através de ferramentas que trabalham com o padrão XML.

Este trabalho demonstrou a necessidade do aumento da flexibilidade na construção de ferramentas que permitem a interoperabilidade entre bases de dados, exemplificando através da criação de um ambiente de *Data warehouse*. Esta flexibilidade é conseguida de duas formas:

a) através da geração dos dados e comandos entre as bases em um formato padrão, que neste trabalho foi utilizado o XML, tornando a ferramenta independente das bases origem e destino;
b) ao controlar a evolução dos esquemas envolvidos no processo de integração e consolidação de dados, tanto origem quanto destino;

Com a utilização de XML, os arquivos puderam ser gerados com a definição explícita do significado da informação nele contida, vindo a facilitar a sua utilização e validação. As definições dos campos dos arquivos tornaram-se a sua documentação.

Esses pontos são o grande mérito do XML, dentro do foco estudado.

O controle da evolução dos esquemas, com o uso do *modelo integrador*, mantendo a independência dos esquemas origem e destino se

apresentou como uma alternativa viável de ser implementada de forma produtiva.

6 TRABALHOS FUTUROS

Este trabalho não está abordando nem analisando possíveis implementações para integrações síncronas, ficando aberta a possibilidade de avaliações neste sentido.

A implantação completa da solução, utilizando XML, abrangendo tanto os arquivos de dados de operações quanto os cadastros auxiliares, com todos os módulos em modo produtivo é outro trabalho que pode ser complementado.

O fato de o modelo de integração ter informações sobre a base origem e destino nos permite pensar em aplicar este tipo de solução, com modelo e manutenção via arquivos XML, em bases heterogêneas, não somente para transporte de arquivos, mas para acesso direto das informações nas bases originais.

Outro ponto a ser estudado seria a conversão das DTD's para uma das variações do XML, principalmente a que trata de validação de tipos de dados, XML *Schema*. Isto permitiria que os tipos de dados recepcionados tivessem uma validação mais apurada já no momento da análise de tipos feita pelo *parser*.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, R. et al. XML: Professional XML. Birminghan: Wrox Press Ltd. April/2000.

BARBIERI, C. XML: a linguagem do futuro. Computerworld - Edição 322 - 26/06/2000

BATINI, C.; LENZERINI, M.; NAVATHE, S. B. A comparative analysis of methodologies for database schema integration. ACM Computing Surveys, v18, n.4 p. 323-364, Dec. 1986.

BOBAK, A. Distributed and Multi-Database System. Boston: Artech House. 1996.

BRAY, T.; HOLLANDER, D.; LAYMAN, A. Namespaces in XML. Word Wide Web Consortium (W3C). Disponível em: http://www.w3.org/TR/REC-xml-names Acesso em: 16 mar. 2000

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M. Extensible Markup Language (XML) 1.0. Word Wide Web Consortium (W3C). Disponível em: http://www.w3.org/TR/REC-xml Acesso em: 16 mar. 2000

COLBY, L. S. et al. Supporting Multiple View Maintenance Policies. Disponível em: http:// citeseer.nj.nec.com/colby97/colby97suporting.html Acesso em 31 out. 1999.

CONRAD, S. et al. Schema Integration with Integrity Constraints. Advances in Databases, 15th British National Conf. on Databases LNCS 1271, Springer-Verlag, pp. 200-214, 1997.

FRIEDMAN, M.; LEVY, A.; MILLSTEIN, T. Navigational Plans for Data Integration – University of Washington. Disponível em: http://www.cs.washington.ed/homes/todd/papers/aaai99.html Acesso em 27 abr. 2000.

DUSCHKA, O. M.; GENESERETH, M. R.; LEVY, A. Y. Recursive Query Plans for Data Integration. Department of Computer Science, Stanford University. Disponível em:

http://citeseer.nj.nec.com/duschka97recursive.html Acesso em 11 jan. 2000.

ELMAGARMID, A.; RUSINKIEWICZ, M.; SHETH, A. Management of Heterogeneous and Autonomous Database Systems. San Francisco: Morgan Kaufmann Publishers, Inc, 1999.

ELMASRI, R.; NAVATHE, S. B. Fundamentals of Database System. California: Addison-Wesley Publishing. 1994.

GODFREY, P.; GRANT, J.; GRYZ, J.; MINKER, J. Integrity Constraints: Semantics and Applications. In: Logics for Databases and Information Systems. USA: Kluwer, 1998.

GOTTHARD, W.; LOCKEMANN, P. C.; NEUFELD, A. System-Guided View integration for Objetct-Oriented Databases. IEEE Transactions on Knowledge and Data Engeneering, vol 4, num 1, pp 1-22, february 1992.

GREFEM, P.; WIDOM, J.Integrity Constraint Checking in Federated Databases. Departament of Computer Science University of Twente and Department of Computer Science, Stanford University. Disponível em: http:// citeseer.nj.nec.com/71115.html Acesso em 25 abr. 2000.

GUPTA, H.; MUMICK, I. S. Selection of Views to Materialize Under a Maintenance Cost Constraint. Department of Computer Science, Stanford University, Stanford and Savera Systems, Summit.

GUPTA, A.; WIDOM, J. Local Verification of Global Integrity Constraints in Distributed Databases. Department of Computer Science, Stanford University and IBM Almaden Research Center. Disponível em: http:// dbpubs.stanford.ed:8090/pub/1993-13 Acesso em 31 out. 1999.

HAMMER, J.; MCLEOD, D. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. International Journal of Intelligent & Cooperative Information Systems, World Scientific, 2:1, pp 51-83, 1993. Disponível em: http://www.cise.ufl.edu/~jhammer/publications.html Acesso em 15 abr. 2000.

INMON, W. H. Building the Data Warehouse. USA, Wiley Computer Publishing, 1996.

IVES, Z. G. et al. An Adaptive Query Execution System For Data Integration. Disponível em: http://bauhaus.cs.washington.ed/homes/zives/sigmod99/index.html Acesso em 20 fev. 2000.

KIM, W. Modern Database Systems: The Object Model, Interoperability, and Beyond. USA: ACM Press, 1995.

KIMBALL, R. The Data Warehouse Toolkit. New York: Wiley, 1996.

KIMBALL, R.; REEVES, L.; ROSS, M.; THORNTHWAITE, W. The Data Warehouse Lifecycle Toolkit. New York: Wiley, 1998.

LIGHT, R. Iniciando em XML. São Paulo: Makron Books, 1999.

MOURA, D. F. C. XML Extensible Markup Language, por David Moura. COPPE Universidade Federal do Rio de Janeiro.

MURPHY, M.; KULKARNI, U.; ROTHENBERGER, M. A Framework for Evaluating Heterogeneous Database Schema Integration Methodologies. In: Association for Information Systems Americas Conference, Indianapolis, August 1997. Disponível em: http://www.uwm.edu/~rothenb/ Acesso em 25 mar. 2000.

RODACKI, A. Aplicação de estratégias de integração de bancos de dados: Um estudo de caso. Curitiba, 2000. 131 f. Dissertação (Mestrado em Informática) – Departamento de Informática, Universidade Federal do Paraná.

SATTLER, K.; SAAKE, G. Supporting Information Fusion with Federated Database Technologies. Department of Computer Science, University of Magdeburg, Germany.

SPACCAPIETRA, S.; PARENT, C.; ERC+: an object based entity relationship approach. In: CONCEPTUAL MODELING, DATABASES AND CASE: AN INTEGRATED VIEW OF INFORMATION SYSTEM DEVELOPMENT. John Wiley, 1992.

SPACCAPIETRA, S.; PARENT, C. Issues and Approaches of Database Integration. COMMUNICATIONS OF THE ACM, vol 41, no 5, pp 166-178, may 1998.

WIENER, J. L. et al. A System Prototype for Warehouse View Maintenance. Department of Computer Science - Stanford University. Disponível em: http://www-db.stanford.edu/warehouseing/warehouse.html Acesso em: 25 abr. 2000.

WIENER, J. L.; ZHUGE, Y.; GARCIA-MOLINA, H. Multiple View for Data Warehousing. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 13. IEEE, 1997. Disponível em: http://db.stanford.edu/pub/zhuge/1996/mvc.ps Acesso em 25 abr. 2000.

ZISMAN, A. XML: An Overview. Department of Computing, City University – London - UK. Disponível em: http://www.soi.city.ac.uk/~zisman/PublicationsAZ.html

ZISMAN, A.; EMMERICH, W.; FINKELSTEIN, A. Using XML to Build Consisency Rules for Distributed Specifications. In: Proc. of the 10th International Workshop on Software Specification and Design. IEEE Computer Society Press. San Diego, November 2000. Disponível em: http://www.cs.ucl.ac.uk/staff/W.Emmerich/publications/

ZISMAN, a.; KRAMER, J. Towards Interoperability in Heterogeneous Database Systems. Disponível em: http://citeseer.nj.nec.com/zisman95toward.html Acesso em 25 abr. 2000.