

**Uma Metodologia para Integração de Sistemas
Legados e Bancos de Dados Heterogêneos.**

Hélio Rubens Soares

Dissertação de Mestrado

Uma Metodologia para Integração de Sistemas Legados e Bancos de Dados Heterogêneos. ¹

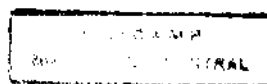
Hélio Rubens Soares

Dezembro de 1998

Banca Examinadora:

- Profa. Dra. Claudia Bauzer Medeiros (Orientadora)
- Profa. Dra. Karin Becker (PUC-RS)
- Profa. Dra Maria Beatriz Felgar de Toledo (UNICAMP)
- Prof. Dr. Neucimar Jerônimo Leite (UNICAMP)

¹Trabalho financiado pelo CNPq, CAPES e FAPESP.



19981105

UNIDADE	BC
N.º CHAMADA:	UNICAMP
	Solim
V.	
TÍTULO	37827
PREÇO	229/99
	IX
PREÇO	R\$ 11,00
DATA	10/06/99
N.º CPO	

CM 00.124523-4

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IMECC DA UNICAMP

Soares, Hélio Rubens

Solim Uma metodologia para integração de sistemas legados e bancos de dados heterogêneos / Hélio Rubens Soares -- Campinas, [S.P. :s.n.], 1998.

Orientador : Claudia Bauzer Medeiros

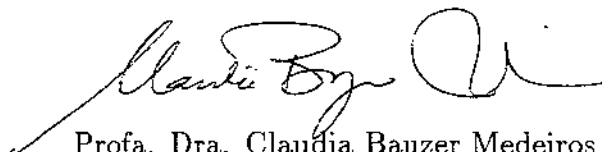
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Banco de dados. 2. Sistemas de informação geográfica. I. Medeiros, Claudia Bauzer. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Uma Metodologia para Integração de Sistemas Legados e Bancos de Dados Heterogêneos.

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Hélio Rubens Soares e aprovada pela Banca Examinadora.

Campinas, 11 de dezembro de 1998.

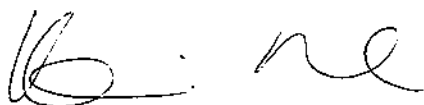


Profa. Dra. Cláudia Bauzer Medeiros
(Orientadora)

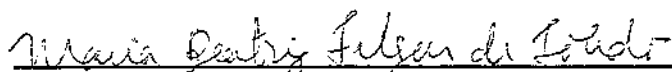
Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

TERMO DE APROVAÇÃO

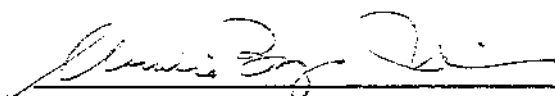
Dissertação defendida e aprovada em 11 de dezembro de 1998,
pela Banca Examinadora composta pelos Professores Doutores:



Profa. Dra. Karin Becker
PUC-RS



Profa. Dra. Maria Beatriz Felgar de Toledo
IC - UNICAMP



Profa. Dra. Claudia Maria Bauzer Medeiros
IC - UNICAMP

© Hélio Rubens Soares, 1998.
Todos os direitos reservados.

"Quando uma criatura humana desperta para um grande sonho e sobre ele lança toda a energia de sua alma, todo o universo conspira a seu favor."

Goethe.

Resumo

Cada vez mais, aplicações precisam acessar diferentes fontes de dados para obter as informações que lhes são necessárias. Muitas destas fontes são gerenciadas por sistemas antigos, chamados de sistemas legados, e precisam passar por um processo de integração ou migração de dados para se tornarem mais flexíveis e passíveis de modificações.

Esta dissertação propõe uma metodologia que auxilia o processo de integração destas fontes de dados heterogêneas e que leva em consideração dados de sistemas legados. A integração de dados é feita através de um sistema de bancos de dados federados. Para isto, foram apresentados vários conceitos que permitem identificar as melhores alternativas a serem aplicadas em cada caso de integração e um algoritmo que constrói o sistema federado e processa consultas submetidas a ele. A proposta foi validada a partir de sistemas legados e bancos de dados existentes na cidade de Paulínia-SP e que fazem parte de secretarias e repartições públicas ligadas à prefeitura.

Abstract

Applications increasingly need to access different data sources to get information. Many of these sources are managed by legacy systems, and need to be integrated or migrated to become more flexible and manageable.

This work proposes a methodology to help the integration of these heterogeneous data sources and which takes legacy data into account, considering the features of each system. Data integration is done through a federated database system. The methodology takes into account several factors that help the choice of the better solution to be applied on each case, and an algorithm to make the federated system and to process queries using this system. The proposed methodology was validated by a case study on databases and legacy systems of the city hall of Paulínia, SP.

Agradecimentos

Primeiramente gostaria de agradecer a meus pais, Milton e Raquel pela motivação e o apoio presentes desde o início. Difícil expressar com palavras o quanto eles foram importantes nestes anos.

Agradeço a minha irmã Cláudia pelo carinho sincero demonstrado a cada encontro.

Ao meu tio Paulo pelas opiniões e incentivo, ajudando-me a solucionar as dúvidas e me fazendo questionar as decisões tomadas e a minha avó Diva pela torcida.

Aos meus grandes e eternos amigos de Uberlândia que não é um, são tantos, e que também estiveram ao meu lado, incentivando e torcendo para que tudo desse certo.

Aos amigos Guilherme Pimentel, Guilherme Albuquerque e Sandro pelo companheirismo e paciência durante estes anos. Obrigado pelas dicas e conselhos que me fizeram diminuir os caminhos a serem percorridos e pelos momentos de lazer e alegria indispensáveis para qualquer ser humano.

Ao Walter Buiatti por despertar em mim a vontade de aprender mais e buscar o mestrado como uma forma de começar este caminho.

A minha orientadora Claudia que, além de uma grande mestra, é, sem dúvidas, uma das pessoas mais humanas que já conheci.

Obrigado às professoras Karin Becker, Maria Beatriz e ao professor Neucimar pelas críticas e sugestões ao trabalho.

Agradeço também aos tantos amigos que conquistei durante o tempo que aqui estive e que me proporcionaram tantos momentos alegres, permanecendo presentes também nos momentos mais difíceis. Em especial, agradeço ao grupo de banco de dados pelas críticas e sugestões ao longo deste trabalho e ao Ricardo Torres pelas implementações e opiniões.

Agradeço a Deus, pela força e coragem que não me deixaram desistir.

Obrigado àquelas pessoas que sentiram minha falta e que souberam entender a minha ausência. Elas sabem quem são.

Este trabalho foi desenvolvido com auxílio do CNPq, CAPES e FAPESP e faz parte do projeto SAI (Sistemas avançados de informação) do PRONEX II do MCT.

Não é fácil afastar-se de tantas pessoas importantes para alcançar um objetivo. Mas as dificuldades sempre farão parte de toda conquista, seja ela qual for, e são elas que nos mantêm em constante evolução...

Conteúdo

Resumo	vii
Abstract	viii
Agradecimentos	ix
1 Introdução	1
2 Revisão Bibliográfica	4
2.1 Bancos de Dados Heterogêneos	4
2.2 Sistemas de Bancos de Dados Federados	8
2.2.1 Problemas Relacionados com SBDF	8
2.2.2 Soluções para os Problemas de SBDF	10
2.3 Outros Trabalhos Correlatos	14
2.4 Sistemas Legados	18
2.4.1 Trabalhos Correlatos	20
2.4.2 Metodologia Chicken Little	23
2.5 Resumo do Capítulo	24
3 Metodologia Proposta	26
3.1 Visão Geral	26
3.2 Padronização e Exportação	31
3.2.1 Padronização dos Sistemas de Bancos de Dados	33
3.2.2 Padronização dos Sistemas Legados	37
3.2.3 Exportação	39
3.3 Criação do Sistema de Bancos de Dados Federados	40
3.3.1 Algoritmo para Construção da Federação Proposto por Zhao	40
3.3.2 Exemplo de Aplicação do Algoritmo de Zhao	45
3.3.3 Modificações Propostas ao Algoritmo de Zhao	51
3.4 Síntese da Metodologia Proposta	59

3.5	Resumo do Capítulo	61
4	Aplicação da Metodologia Baseada em Bancos de Dados Reais	62
4.1	Padronização e Exportação	63
4.2	Construção da Federação	66
4.3	Resumo do Capítulo	72
5	Metodologia Aplicada a Sistemas de Informações Geográficas	74
5.1	Sistemas de Informações Geográficas	74
5.1.1	Modelo de Dados para Aplicações Geográficas	75
5.1.2	Formato dos Dados	76
5.1.3	Caracterização das Aplicações	76
5.2	SIGs e a Metodologia Proposta	77
5.3	Resumo do Capítulo	81
6	Conclusões e Extensões	83
	Bibliografia	86

Lista de Tabelas

2.1	Taxonomia dos sistemas de compartilhamento de informações.	6
3.1	Principais diferenças entre federações fracamente acopladas e fortemente acopladas.	29
3.2	Características gerais dos sistemas legados e soluções aplicáveis a cada caso.	38
3.3	Tabela semântica.	46
3.4	Matriz de correspondência de atributos.	46
3.5	Tabela exportada referente ao componente de banco de dados B_1	47
3.6	Tabela exportada referente ao componente de banco de dados B_2	48
3.7	Tabela exportada referente ao componente de banco de dados B_3	48
3.8	Tabela exportada referente ao componente de banco de dados B_2'	54
3.9	Tabela exportada referente ao componente de banco de dados B_2' , com suporte para chaves estrangeiras compostas.	55
3.10	Tabela exportada referente ao componente de banco de dados <i>Gerente</i> . . .	56
3.11	Tabela exportada referente ao componente de banco de dados <i>Gerente-Seção</i> .	57
3.12	Tabela exportada referente ao componente de banco de dados B_3 , com informações dos domínios dos atributos.	58
4.1	Tabela exportada referente ao componente de banco de dados Educação. .	67
4.2	Tabela exportada referente ao componente de banco de dados Esporte. . .	67
4.3	Tabela exportada referente ao componente de banco de dados SAE.	68
4.4	Tabela semântica.	68
4.5	Matriz de correspondência de atributos.	69

Lista de Figuras

2.1	Taxonomia dos sistemas integrados de bancos de dados.	5
2.2	Sistema de Bancos de Dados Federados e suas componentes.	8
2.3	Exemplo de conversão de esquemas de bancos de dados.	15
2.4	Conversão de aplicações.	17
2.5	Arquitetura dos sistemas legados.	19
3.1	Entradas e saídas da metodologia de integração.	27
3.2	Fases do processo de integração de bancos de dados heterogêneos e sistemas legados. Fase 1: Padronização; Fase 2: Exportação; Fase 3: Criação do sistema integrado.	28
3.3	Processos da metodologia de integração.	31
3.4	Arquitetura dos SBDFs - 5 níveis.	32
3.5	Tradução dos modelos dos bancos de dados locais para o modelo de dados canônico (MDC).	34
3.6	Entradas e saídas do algoritmo para construção do esquema coordenado.	41
3.7	Lei de formação para as tabelas exportadas	42
3.8	Esquemas exportados de três bancos de dados componentes.	45
3.9	Processo de tradução da subconsulta envolvendo os componentes B_3 e B_1	50
3.10	Processo de tradução da subconsulta envolvendo os componentes B_3 e B_2	50
3.11	Esquema exportado do banco de dados componente B_2' , com chave estrangeira composta.	53
3.12	Esquema exportado do banco de dados componente <i>Gerente</i>	54
3.13	Esquema exportado do banco de dados componente <i>Gerente-Seção</i>	56
3.14	Ligações explícitas e implícitas entre as relações de cada banco de dados componente.	57
4.1	Diagrama Entidade-Relacionamento do componente legado Esporte.	64
4.2	Esquemas dos componentes <i>Educação</i> , <i>Esporte</i> e <i>SAE</i>	65
4.3	Sistema Federado com componentes <i>Educação</i> , <i>Esporte</i> e <i>SAE</i>	66

4.4	Processo de tradução da subconsulta envolvendo os componentes Educação e Esporte.	71
4.5	Processo de tradução da subconsulta envolvendo os componentes Educação e SAE.	72
4.6	Tela para formulação de consultas no sistema federado.	73
5.1	Primeiro caso de integração de dados envolvendo SIG.	78
5.2	Segundo caso de integração de dados envolvendo SIG.	80
5.3	Terceiro caso de integração de dados envolvendo SIG.	80

Capítulo 1

Introdução

A necessidade de integração de dados e aplicações vem aumentando dentro das organizações. Cada vez mais, as informações estão dispersas por vários setores e departamentos. É preciso disponibilizar aos usuários ferramentas que permitam a automatização dos processos de integração de dados e de aplicações sem que os usuários precisem ter conhecimento de toda a estrutura interna necessária para armazená-los. O problema de integração é agravado pela presença de *sistemas legados*. Basicamente, sistemas legados são sistemas antigos com pouca documentação e que resistem a evoluções e modificações.

Exemplos deste cenário são as administrações municipais que passam por processos de informatização, muitas vezes acompanhado por implantação de sistemas de informações geográficas (SIGs). Geralmente, já existem nas prefeituras sistemas responsáveis por parte da administração pública, localizados em algumas secretarias e que precisam continuar em funcionamento após a implantação dos novos sistemas. Os sistemas já existentes são, neste caso, sistemas legados a serem integrados ao novo sistema. Existem, no caso, dois problemas de integração: a integração de dados e sistemas legados a novos sistemas em desenvolvimento (migração) e a integração de dados heterogêneos visando seu uso unificado.

O objetivo desta dissertação é abordar o problema de integração de sistemas legados e bancos de dados heterogêneos, motivado pelo processamento de aplicações urbanas. A solução proposta é baseada na criação de um sistema federado. O tratamento de sistemas legados pode se dar tanto em nível de dados quanto de aplicações. Esta dissertação concentra-se na parte de tratamento de dados. Como resultado, a dissertação estabelece uma metodologia que parte da padronização dos sistemas envolvidos na integração até a especificação do sistema federado. Isto envolve três atividades, do ponto de vista de bancos de dados:

- Modelagem dos diferentes bancos de dados e sua integração na federação - problemas de heterogeneidade e autonomia dos bancos de dados que irão compor a federação.

- Metodologia de migração de sistemas legados e sua integração ao sistema federado - problemas de engenharia reversa, migração e evolução de dados e aplicações.
- Construção de um sistema de bancos de dados federados que permita o compartilhamento dos dados dos bancos de dados.

O enfoque básico desta dissertação está calcado em dois mecanismos: migração e construção da federação. Para a migração, a base é uma adaptação da abordagem de migração de sistemas legados proposta em [BS95] que sugere uma seqüência de passos baseada em um processo de *migração paulatina*. O processo de migração é acoplado a metodologias de integração de bancos de dados heterogêneos. Para construção do sistema federado, a dissertação estende o trabalho de Zhao [Zha97], que utiliza um *esquema coordenado* para estabelecer o compartilhamento de dados.

O ponto de partida para o desenvolvimento desta dissertação foi um estudo de caso junto à prefeitura da cidade de Paulínia-SP em 1997, que levantou os problemas relativos à integração de sistemas já existentes a um SIG. A prefeitura de Paulínia pretendia desenvolver um projeto de longo prazo de integração dos dados e sistemas disponíveis, visando a construção de um grande sistema de informação. Este projeto foi denominado projeto SIGGER - *Sistema de Informações Gerenciais Geograficamente Referenciado*. O objetivo final seria realizar todas as aplicações de planejamento urbano usando um sistema geográfico, integrando os departamentos e secretarias da prefeitura através de uma rede para um melhor atendimento à população. No entanto, já existiam alguns sistemas em funcionamento como por exemplo, gerenciamento de cadastro urbano, cadastro de desempregados, controle de atividades esportivas e outros, que deveriam ser incorporados ao novo projeto.

Os problemas constatados neste estudo de caso motivaram o trabalho desenvolvido na dissertação e nortearam o desenvolvimento da solução apresentada. Como mostrado no capítulo 4, a solução é adequada para projetos como o proposto em Paulínia, permitindo integração de dados legados a outros sistemas. As principais contribuições desta dissertação são, portanto:

- Análise das opções de integração e migração de dados de sistemas legados visando sua utilização em sistemas de bancos de dados federados;
- Proposta de uma metodologia de integração de bancos de dados heterogêneos e sistemas legados através da construção de um sistema de bancos de dados federados.
- Aplicação da metodologia a um caso real composto por sistemas legados e sistemas de bancos de dados da cidade de Paulínia-SP;

- Linhas gerais para adaptação da metodologia, buscando suportar a inserção de SIGs ao sistema de bancos de dados federados;
- Validação através de um protótipo para um sistema de bancos de dados federados.

Os capítulos seguintes estão organizados da seguinte forma. O capítulo 2 apresenta uma revisão bibliográfica que aborda questões de gerenciamento de bancos de dados heterogêneos e sistemas legados, destacando os trabalhos mais importantes do ponto de vista desta dissertação. O capítulo 3 apresenta a metodologia proposta para integração de bancos de dados heterogêneos e sistemas legados, detalhando todos os passos necessários para se estabelecer o sistema integrado. O capítulo 4 mostra uma aplicação da metodologia proposta, baseando-se em alguns bancos de dados heterogêneos e sistemas legados da cidade de Paulínia. O capítulo 5 discute como a metodologia se comporta quando a integração de dados envolve sistemas de informações geográficas. Por fim, o capítulo 6 apresenta as conclusões e extensões desta dissertação.

Capítulo 2

Revisão Bibliográfica

Esta dissertação analisa o problema de integração de sistemas de informações a sistemas legados para gerenciamento de aplicações urbanas. Este capítulo aborda os principais conceitos relativos a integração de sistemas de informações, com ênfase em sistemas federados e apresenta uma visão geral de sistemas legados.

2.1 Bancos de Dados Heterogêneos

Os dados existentes nas organizações, na grande maioria das vezes, são heterogêneos, o que dificulta sua integração e migração para sistemas mais flexíveis. Este problema é tratado no contexto de bancos de dados heterogêneos.

A integração de bancos de dados pode ser real ou virtual. No primeiro caso, há apenas um banco de dados resultante enquanto que no segundo caso o usuário tem visões do banco de dados integrado. A criação destas visões pode ser entendida como um processo que recebe como entrada o conjunto dos esquemas dos diferentes sistemas a serem integrados, mais as transações neles processadas, e oferece como saída visões integradas que permitem ao usuário realizar suas transações sem se preocupar com a localização e organização dos dados.

Existem várias propostas para integração de bancos de dados heterogêneos. Esta seção apresenta uma revisão de alguns dos principais trabalhos encontrados na literatura e que servirão como base teórica para os capítulos seguintes. Alguns trabalhos apresentam uma revisão bibliográfica parcial sobre o assunto, como é o caso de [Bre90, Hul97] e [SM97], que são revistos e expandidos nesta seção.

Vários termos são usados para referenciar ambientes heterogêneos de bancos de dados, ou seja, ambientes que integram bancos de dados diferentes. O próprio termo *heterogêneo* pode ser encontrado na literatura sob várias formas. Segundo Aguiar [Agu95], um sistema de banco de dados heterogêneo é sinônimo de sistema de bancos de dados federados (seção

2.2) mas, de modo geral, o termo heterogêneo é usado para referenciar um conjunto de bancos de dados com diferentes esquemas, protocolos de comunicação, hardware e linguagens de consulta.

O termo *sistemas com múltiplos bancos de dados* (*multidatabase system*) é também bastante usado para referenciar bancos de dados heterogêneos. Um nome um pouco menos abrangente é *sistema global*, geralmente usado para sistemas que mantêm um esquema global responsável por representar os esquemas de todos os bancos de dados locais, ou seja, os sistemas de banco de dados que compõem o sistema global.

Esta dissertação usará os termos *sistema integrado* ou *sistema de bancos de dados federados* para expressar a idéia de união de bancos de dados heterogêneos, por ser a solução escolhida como base do desenvolvimento deste trabalho.

Várias abordagens são factíveis para a criação de um sistema integrado de banco de dados, dependendo do nível de acoplamento e, conseqüentemente, da autonomia que se deseja manter nos bancos de dados locais. A figura 2.1, retirada de [SL90], mostra as principais abordagens.

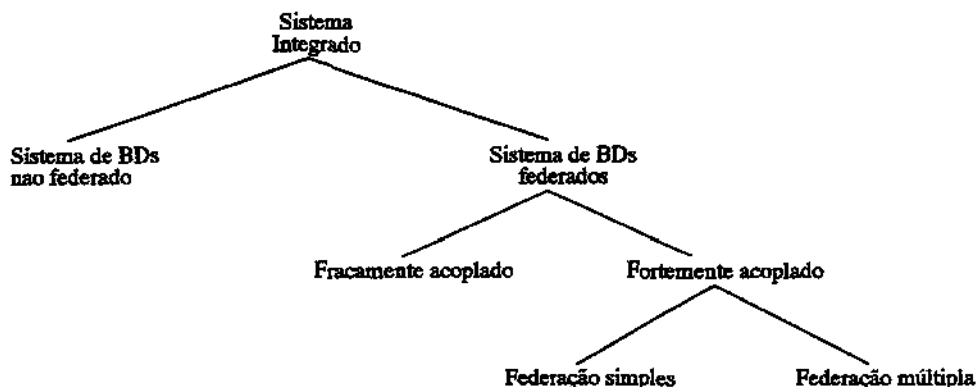


Figura 2.1: Taxonomia dos sistemas integrados de bancos de dados.

Um *sistema não federado* visa um tipo de organização que não mantém a autonomia dos bancos de dados locais, existindo apenas um nível de gerenciamento, sendo que as operações locais e globais são processadas da mesma forma. Já um *sistema federado* consiste em um conjunto de bancos de dados autônomos em que cada banco de dados permite o compartilhamento de parte dos seus dados por outros bancos de dados. Além disto, um sistema de bancos de dados federado pode ainda ser classificado em *fracamente* ou *fortemente acoplado*, dependendo de como é feito o compartilhamento dos dados.

Uma federação fortemente acoplada é caracterizada pelo baixo grau de autonomia dos bancos de dados componentes da federação, bem como pela presença de um esquema global integrado que representa todas as informações dos componentes que serão compartilhadas. Estas informações são partes de esquemas integrados, criando para os usuários

	Fortemente acoplado (+)				(-) Fracamente acoplado	
	BDs Distribuídos	Esquema Global de Múltiplos BDs	BDs Federados	Linguagem p/ Sistemas com Múltiplos BDs	Linguagem p/ Sistemas com Múltiplos BDs Homogêneos	Sistemas Inter-Operáveis
O sistema global tem acesso ...	às funções internas do SGBD local	à interface de usuário do SGBD local	à interface de usuário do SGBD local	à interface de usuário do SGBD local	à interface de usuário do SGBD local + funções internas	às aplicações sobre o SGBD local
Os nós locais são...	BDs homogêneos	BDs heterogêneos	BDs heterogêneos	BDs heterogêneos	BDs homogêneos	quaisquer tipos de dados
Possui funções globais completas de BDs?	sim	sim	sim	sim	sim	não

Tabela 2.1: Taxonomia dos sistemas de compartilhamento de informações.

a ilusão de que apenas um único banco de dados está sendo utilizado. O administrador da federação fica responsável por criar e gerenciar a federação, além de controlar os acessos aos dados compartilhados. Se mais de um esquema federado for disponibilizado aos usuários, esta federação é chamada de *federação múltipla*, caso contrário, ela é chamada de *federação simples*.

Já a federação fracamente acoplada contempla um pouco mais a manutenção da autonomia dos bancos de dados componentes, criando ferramentas que possibilitam o acesso aos dados compartilhados sem a criação de um esquema global. Neste caso, cabe aos usuários e ao administrador a responsabilidade de gerenciar e controlar o acesso aos dados na federação. Desta forma, a escolha entre os dois tipos de federação é baseada no grau de autonomia que se deseja manter nos bancos de dados componentes e no grau de responsabilidade que se deseja atribuir aos usuários do sistema federado.

Hurson et al. [HB91] apresentam uma comparação entre seis sistemas globais, usando como critério o grau de acoplamento de cada um. Dentre estes sistemas, alguns são mais usados para bancos de dados homogêneos e outros, mais importantes para os objetivos desta dissertação, para bancos de dados heterogêneos. A tabela 2.1, adaptada de [HB91], resume esta classificação.

O primeiro sistema da tabela, classificado como o mais fortemente acoplado (1a. coluna), é um *banco de dados distribuído* em que os SGBDs locais, geralmente homogêneos, ficam praticamente indistinguíveis uns dos outros. Um esquema global é mantido com toda a informação disponível do sistema e os usuários acessam o sistema através de consultas submetidas a este esquema global. Como o sistema global mantém todo o controle dos processos realizados, esta alternativa apresenta o melhor desempenho mas, por outro

lado, requer mudanças significativas nos SGBDs locais.

O segundo sistema é o *esquema global de múltiplos BDs* em que as funções globais acessam as informações locais através de interfaces externas dos SGBDs locais. Neste caso, bancos de dados heterogêneos podem participar da integração com mais facilidade mas o sistema continua mantendo um esquema global. No entanto, existe maior cooperação por parte dos bancos de dados locais.

Bancos de dados federados correspondem ao nível médio de acoplamento. Tanto a autonomia dos SGBDs locais quanto o desempenho das operações globais ficam menos comprometidos se comparados com os demais sistemas de integração. Não existe, neste caso, um único esquema global e cada banco de dados local mantém sua própria porção do esquema global, contendo apenas as informações globais necessárias ao uso local. Esta porção pode ser construída através de visões de bancos de dados em que cada membro da federação possui uma visão sobre os demais membros, o que requer um nível mais alto de cooperação dos bancos de dados locais. Um exemplo deste tipo de sistema, proposto por Zhao [Zha97], será usado nesta dissertação como base para construção do sistema federado da metodologia proposta. Uma explicação mais detalhada desta proposta está apresentada na seção 3.3.

No caso da *linguagem para sistemas com múltiplos BDs*, o esquema global desaparece completamente, aproximando-se mais da característica de fracamente acoplado. O sistema integrado garante suas funcionalidades através de linguagens de consulta. Da mesma forma que o segundo sistema apresentado, este sistema permite integrar SGBDs heterogêneos preexistentes sem modificá-los.

Também voltada mais especificamente para bancos de dados homogêneos, a *linguagem para sistemas com múltiplos BDs homogêneos* possui uma importância considerável do ponto de vista de pesquisa por caracterizar alguns dos primeiros sistemas disponíveis comercialmente para ambientes com múltiplos bancos de dados como, por exemplo, o Empress e o Sybase [HB91].

Por fim, os *sistemas interoperáveis* são considerados os sistemas mais fracamente acoplados e as funções globais são limitadas a simples trocas de mensagens. Estes sistemas não suportam todas as funções de banco de dados como, por exemplo, processamento de consultas. O que acontece neste caso é a definição de protocolos padrão para simples comunicação entre os SGBDs locais.

Esta dissertação está voltada aos sistemas de bancos de dados federados por serem estes uma alternativa mais equilibrada para o problema de bancos de dados heterogêneos, pois apresentam um nível de acoplamento moderado, possibilitando funções globais eficientes sem comprometer a autonomia dos SGBDs locais. A próxima seção explora este aspecto de sistemas heterogêneos.

2.2 Sistemas de Bancos de Dados Federados

Uma das propostas mais importantes do ponto de vista desta dissertação para a manipulação de banco de dados heterogêneos é a criação de um *sistema de bancos de dados federados* (SBDF). Formalmente, um SBDF pode ser definido como uma coleção de sistemas de bancos de dados independentes, cooperativos, possivelmente heterogêneos, que são autônomos e que permitem o compartilhamento de todos ou alguns de seus dados, sem afetar as suas aplicações locais [SL90, SD94]. Um esboço de um SBDF pode ser visto na figura 2.2, adaptada de [SL90].

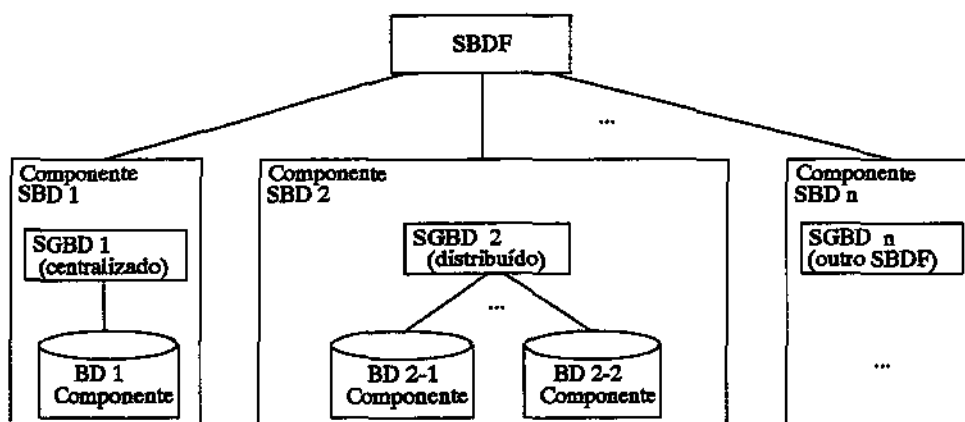


Figura 2.2: Sistema de Bancos de Dados Federados e suas componentes.

De acordo com a figura 2.2, cada banco de dados local pertencente à coleção é chamado de sistema de banco de dados (SBD) componente (ou apenas componente), podendo ter diferentes características como modelo de dados, linguagens de consulta e capacidade de gerenciamento de transações e podendo participar de uma ou mais federações. Além disso, cada componente pode conter os dados centralizados, distribuídos ou ser um outro SBDF. O software responsável por controlar e coordenar a manipulação dos diversos componentes é chamado de *sistema de gerenciamento de banco de dados federado* (SGBDF). A principal característica de uma federação é a cooperação de cada sistema independente.

2.2.1 Problemas Relacionados com SBDF

Os SBDFs caracterizam-se pela heterogeneidade, presença de dados distribuídos e autonomia de cada componente. A *heterogeneidade* pode ser identificada em diversos níveis e uma das principais causas refere-se às diferenças entre os bancos de dados componentes, como estrutura de dados, nomes e interpretações semânticas dos atributos. Outro problema acontece quando os esquemas de cada componente possuem diferentes restrições.

Por exemplo, uma relação entre cursos e instrutores pode ser $1:1$ em um componente (um curso possui um único instrutor) e $1:N$ em outro componente (um curso possui vários instrutores). A heterogeneidade envolvendo restrições de esquema não será abordada nesta dissertação e o trabalho de Lee e Ling [LL97] propõe uma solução para este caso. Quando sistemas legados aparecem como um dos componentes da federação, o grau de heterogeneidade do SBDF aumenta porque alguns sistemas legados não utilizam bancos de dados, ou então os utilizam de forma rudimentar.

O problema não fica restrito apenas a resolver as diferenças, mas é necessário também identificá-las. Apontar e formalizar os conflitos existentes entre os componentes do SBDF não é uma tarefa trivial. Uma das formas de se resolver este problema é propor um dicionário de dados com os mapeamentos necessários para uma descrição mais precisa dos conflitos, como o dicionário semântico proposto em [CA97]. Outro tipo de solução é o trabalho apresentado por Bright et al. [BHP94] que propõe uma extensão dos sistemas com múltiplos bancos de dados chamado *Summary Schemas Model* (SSM). Através de cálculos de similaridade usando um conjunto de medidas denominado *Semantic-Distance Metric* (SDM), dados semanticamente similares são identificados em diferentes bancos de dados locais a partir de um conjunto de esquemas de entrada.

Outro problema envolvendo SBDF é a *distribuição dos dados*. Na maioria dos casos, cada componente do SBDF foi criado independentemente, sem um objetivo prévio de integração. Os dados armazenados em cada sistema podem estar geograficamente distribuídos e interconectados por algum sistema de comunicação que, por sua vez pode ser diferente para cada componente. Esta distribuição pode incluir partições horizontais e verticais, duplicação ou fragmentação dos dados. A questão de distribuição dos dados não será abordada nesta dissertação.

Um ponto crítico em SBDF é a *autonomia* inerente aos bancos de dados componentes. Como não existe um único sistema central para coordenar as ações da federação, a arquitetura utilizada em um SBDF deve prover mecanismos para balancear dois objetivos: cada componente deve manter sua autonomia tanto quanto possível; e cada componente deve ser capaz de assegurar os recursos para compartilhamento de dados [HM85]. Os componentes do SBDF possuem quatro tipos diferentes de autonomia: de projeto, de comunicação, de execução e de associação [Agu95].

A *autonomia de projeto* abrange a liberdade de escolha da linguagem de consulta, do modelo de dados, da implementação, das restrições a serem implementadas e da forma de gerenciamento de dados a ser usado, além dos nomes e significados semânticos dos dados bem como das funcionalidades suportadas pelo SBDF.

Por outro lado, a *autonomia de comunicação* permite que cada componente decida quando se comunicar com os outros componentes. Desta forma, um SGBD com autonomia de comunicação é capaz de decidir quando e como irá responder a uma requisição de outro

componente [SL90].

A *autonomia de execução* busca garantir que cada componente possa executar suas operações locais sem interferir na execução de operações não locais, ou seja, operações processadas por usuários de outros componentes.

A *autonomia de associação* pode ser entendida como a capacidade que os componentes têm de decidir quando e como irão compartilhar suas funcionalidades. Isto inclui a capacidade dos componentes de se associarem ou se desligarem da federação, além da capacidade de participar de outras federações ao mesmo tempo.

Heterogeneidade, distribuição e autonomia são problemas inerentes aos SBDFs. No entanto, existem outros fatores que são diretamente afetados com a criação e manipulação de um SBDF. O primeiro deles é a utilização de mecanismos para gerenciar as transações globais destes sistemas, como por exemplo, técnicas eficientes de *recuperação de informação*. O segundo problema refere-se aos mecanismos de *segurança* que devem ser estabelecidos de modo a manter a integridade e as restrições de acesso ao SBDF sem comprometer a autonomia dos SGBDs locais.

Vários trabalhos buscam soluções para os problemas aqui citados como, por exemplo, [Ber96, VL97, Qia93, QL94, SSR94, Agu95, Jr95] e [BFK95], descritos a seguir.

2.2.2 Soluções para os Problemas de SBDF

As principais soluções para SBDF estão relacionadas ao uso das seguintes ferramentas:

- Mediadores
- Tradutores/Adaptadores
- Visões

Mediadores

Um *mediador* é um software usado para permitir a interoperabilidade entre dois ou mais SGBDs, ou seja, permitir que a comunicação entre SGBDs seja possível para atender às necessidades dos usuários. Cada operação que necessite da participação de mais de um SGBD para sua execução é chamada de interoperação. De acordo com Bernstein [Ber96], os mediadores devem apresentar serviços do tipo: gerenciadores de formulários e servidores de diretório; serviços de processamento algorítmico do tipo ordenação, conversão de dados e manipulação de *strings*; serviços de comunicação como mensagens ponto a ponto e troca eletrônica de dados; serviços de controle como gerenciadores de *threads* e transações; gerenciamento de sistema como detector de falhas e controles de acesso, dentre outros.

No caso desta dissertação, o objetivo é utilizar mediadores como um dos mecanismo de acesso aos dados de sistemas legados.

Com a utilização de mediadores, o acesso aos dados heterogêneos é efetuado através de consultas que são submetidas ao mediador, que por sua vez as transforma em subconsultas a serem enviadas aos SGBDs componentes. As subconsultas geradas pelo mediador devem ser traduzidas para as linguagens de consulta de cada SGBD componente. Ao final, os resultados das consultas são traduzidos por *softwares* tradutores para a linguagem de consulta do SGBD componente que gerou a consulta (origem) e a resposta final é devolvida ao usuário [VL97].

Já as aplicações dos sistemas legados, segundo Bernstein [Ber96], podem utilizar serviços de mediadores da seguinte forma. Os mediadores encapsulam as aplicações legadas com um conjunto de funções e usam seus serviços de comunicação para prover acesso remoto a estas funções.

Qian [Qia93] também propõe mediadores para automatizar a interoperação de dados heterogêneos. A principal característica deste trabalho é a separação entre heterogeneidade semântica e de representação dos dados. A heterogeneidade semântica envolve, dentre outros, granularidade de atributos e período das transações (mensal, semanal) e é resolvida através de um “comunicador inteligente”, enquanto que a heterogeneidade de representação envolve, dentre outros, granularidade dos relacionamentos (1:1, 1:N, M:N) e escolha de identificadores para uma relação e é resolvida através de um “compilador inteligente”. Assim, a autonomia de cada componente considerado é preservada, já que os usuários de cada banco de dados componente não são afetados quanto à forma de representação dos dados e tampouco quanto à sua visão do mundo real. Além disto, são possíveis comparações semânticas dos dados sem haver necessariamente o compartilhamento da representação destes dados.

Qian et al. [QL94] sugerem o uso de um mediador de consultas para interoperações de bancos de dados autônomos e heterogêneos considerando também as heterogeneidade semântica e de representação de dados separadamente, como é feito em [Qia93]. O trabalho propõe uma arquitetura para interoperação de dados e aplicações para facilitar o processamento e a formalização semântica das consultas e detectar os conflitos existentes. A abordagem permite interoperabilidade dos dados e aplicações que acessam dados em sistemas legados.

Um pouco diferente das propostas citadas anteriormente que utilizam mediadores, Sciore et al. [SSR94] propõem o uso de *valores semânticos* que funcionam como “unidades de troca” para facilitar a interoperabilidade semântica entre sistemas de informação heterogêneos. O elemento chave desta proposta é o *mediador de contexto*, cujo trabalho é identificar e construir os valores semânticos a serem transmitidos entre componentes para determinar quando a troca de informação é necessária, além de converter os valores

semânticos para uma forma aceitável pelo receptor. Um valor semântico é a associação de um contexto a um valor simples. O termo valor simples é definido como sendo uma instância de um tipo, e cuja semântica é determinada apenas pelo tipo. Assim, se 3 é do tipo Dólar então 3 dólares não podem ser comparados com instâncias do tipo Yen.

Tradutores e Adaptadores

Tradutores/adaptadores, outra solução para SBDF, convertem os dados das fontes de informação para um modelo de dados comum e convertem consultas de aplicações em consultas específicas das fontes de informação envolvidas na consulta [VL97]. Muitas vezes os tradutores/adaptadores são confundidos com mediadores, mas diferenças existem. Os tradutores/adaptadores são geralmente usados apenas na construção do sistema unificado de banco de dados, não fazendo, portanto, a mediação entre os bancos de dados componentes. Enquanto o mediador está sempre presente nas transações entre componentes, os tradutores/adaptadores são ferramentas que, tendo sido usadas, podem ser dispensáveis e o sistema unificado continuará em operação. Alguns mediadores podem usar tradutores/adaptadores como ferramentas para resolver uma parte específica da conversão de dados.

Visões

Finalmente, *visões*, em geral, são usadas como um mecanismo que auxilia a integração dos componentes de um SBDF. Os dados de uma visão podem ficar armazenados fisicamente, constituindo uma *visão materializada*, ou serem gerados a cada utilização da visão e, portanto, ficarem armazenados na memória ou em alguma estrutura temporária, neste caso chamada de *visão virtual* [Hul97].

Em sistemas gerenciadores de bancos de dados relacionais, visões são encaradas como relações virtuais não-normalizadas, definidas em função de relações preexistentes e obtidas com o resultado do processamento de uma consulta [Cer96].

No paradigma de banco de dados orientado a objetos, ainda não existe um consenso sobre o que é uma visão. Para Mamou et al. [MM91], uma *visão orientada a objetos* é uma restrição especial de integridade aplicada ao banco de dados. Esta restrição deve assegurar a ocultação de informação e a proteção dos dados, bem como preservar o encapsulamento, composição e herança do esquema do banco de dados. Para Pitrik [Pit96], uma visão em banco de dados orientado a objetos deve suportar funções análogas a uma visão em bancos de dados relacionais, além de outras funcionalidades como reestruturação de tipos/classes, independência de dados através dos esquemas da visão, integração de banco de dados e uma notação simplificada para pré-definição de consultas que ocorram freqüentemente.

No caso de sistemas federados, visões são utilizadas para prover aos usuários uma

interface comum através da qual ele possa acessar os dados armazenados, independentemente de onde e quais SGBDs estejam sendo federados. Alguns trabalhos utilizam visões como ferramenta para integração de dados, mudanças de esquemas ou para reutilização de softwares. Exemplos destes trabalhos são [Agu95, Jr95] e [BFK95].

Aguiar [Agu95] descreve um exemplo do uso de visões para integração de sistemas geográficos distintos, onde mecanismos de integração de visões são propostos para integrar aplicações da Telebrás e da Eletropaulo (Eletricidade de São Paulo S/A).

Já o trabalho de Novak [Jr95] usa visões em outro contexto: reutilização de software com diferentes representações de dados. Neste trabalho, a reutilização de software é suportada pela construção semi-automática de visões que irão fazer a associação entre os tipos de aplicações de um algoritmo particular e os tipos abstratos usados em um algoritmo genérico. O trabalho apresenta alguns algoritmos que criam visões através de correspondências especificadas pelo próprio usuário com o uso de uma interface gráfica. A principal barreira a ser enfrentada por esta abordagem é manter um algoritmo genérico para uma quantidade razoável de aplicações já que, sem isto, a utilização desta proposta ficaria extremamente limitada. Este problema não é específico deste trabalho, pelo contrário, é característico da grande maioria dos trabalhos que envolvem bancos de dados heterogêneos porque é muito difícil propor uma solução genérica de integração de dados e aplicações devido à complexidade e diversidade das aplicações existentes.

Em outro contexto de utilização de visões, Breche et al. [BFK95] apresentam uma simulação de mudança de esquemas voltada para migração de aplicações. Esta abordagem sugere que sejam desenvolvidas aplicações em cima de visões que integrem os dados, para serem consideradas posteriormente como aplicações do sistema migrado. Desta forma, a visão funciona como um mecanismo de prototipação da migração.

Evidentemente, vários problemas surgem durante o uso de visões em bancos de dados. Um dos problemas mais importantes no âmbito de visões é o processo de *atualização da visão* e dos bancos de dados subjacentes, ou seja, os bancos de dados que fornecem dados para a visão. Quando o usuário processa uma atualização na visão, deve-se disparar um mecanismo de *propagação* para que sejam efetuadas as alterações sobre os bancos de dados subjacentes. Por outro lado, quando uma atualização é processada em algum banco de dados subjacente, deve-se fazer o *refresh* das visões dependentes destes bancos de dados para manter a consistência dos dados. A atualização de visões é um problema em aberto na literatura, indo desde a sistemas relacionais [BS81] até, recentemente, a aplicações em *data warehouse* [ZWM97].

2.3 Outros Trabalhos Correlatos

A necessidade sempre crescente de se ter mecanismos que possibilitem a interoperação de bancos de dados, principalmente heterogêneos, funcionou como uma das maiores motivações para o surgimento de vários trabalhos relativos a este assunto. Além dos trabalhos já citados, outras propostas não menos importantes contribuem para compor o cenário teórico que fundamenta a área de bancos de dados heterogêneos. Estas propostas estão inseridas em vários contextos como bancos de dados orientados a objetos [HMZ90]; prototipação de sistemas integrados [HMN91]; meta banco de dados [HBRY91]; conversão de esquemas e modelos de dados [DK97, OM93, LH90, BDK92, Qia96]; ferramentas baseadas em cliente-servidor [LS97]; conversões de aplicações [SLL81]; conversões de consultas [QR95]; gerenciamento de transações [Geo91, GRS91, Tri96]; e segurança [GQ94].

Bancos de Dados Orientados a Objetos

O trabalho de Heiler et al. [HMZ90] apresenta um sistema de banco de dados heterogêneo orientado a objetos, FUGUE, para prover suporte a sistemas distribuídos. A intenção é estabelecer uma federação que permita: (1) acesso às representações de comportamento e estados heterogêneos, (2) suporte à construção dinâmica de objetos, (3) controle da integração distribuída de objetos, (4) recuperação e, principalmente, (5) segurança.

Prototipação de Sistemas Integrados

Hornick et al. [HMN91] apresentam um trabalho referente à integração de aplicações heterogêneas, autônomas e distribuídas, oferecendo um protótipo que provê um ambiente de testes para computação distribuída baseado no protótipo DOM (Distributed Object Management) [MHG+92]. Esse trabalho foi motivado inicialmente pela GTE Telephone Operations que possui uma grande variedade de hardwares e softwares, caracterizando um ambiente tipicamente heterogêneo e com vários problemas de interoperabilidade. O protótipo desenvolvido foi testado usando uma aplicação que integra três softwares comerciais com diferentes modelos: relacional (Sybase), orientado a objetos (Ontos) e hipertexto (HyperCard).

Meta Banco de Dados

A proposta de Hsu et al. [HBRY91] também apresenta um sistema a ser usado em ambientes heterogêneos e distribuídos. Este sistema é um meta banco de dados chamado GIRD (Global Information Resources Dictionary) usado como um modelo para representação e gerenciamento de metadados unificados. Segundo os autores, para que a

tecnologia de metadados suporte integração de sistemas, o escopo de metadados precisa ser estendido, contendo não só uma simples representação de dados mas também contendo conhecimento dos recursos.

Conversão de Esquemas e de Modelos de Dados

Davidson et al. [DK97] apresentam uma linguagem para conversão de esquemas e restrições chamada WOL (Well-founded Object Logic). Esta linguagem é baseada em um modelo de dados definido no próprio artigo e possibilita a manipulação de bancos de dados envolvendo identidade de objetos, estruturas de dados recursivas e a complexidade destas estruturas.

Já o trabalho de Oliveira et al. [OM93] está voltado para conversão de esquemas de bancos de dados visando a transparência de modelos, propriedade que garante a cada usuário a possibilidade de manipular os dados do sistema global através de um único modelo de dados e da linguagem de manipulação de dados associada a esse modelo. Esta propriedade é obtida em duas etapas: uma estrutural e outra operacional. Para se mapear os modelos e linguagens existentes em um sistema de bancos de dados heterogêneos, Oliveira et al. [OM93] adotam um modelo de dados comum e uma linguagem de manipulação de dados intermediária (muitos para muitos). Assim, os esquemas originais dos sistemas componentes são convertidos em esquemas no modelo comum que são então integrados em um ou mais esquemas virtuais no mesmo modelo (esquema federado). A figura 2.3 apresenta um exemplo desta proposta em que dois esquemas, um utilizando modelo de dados rede e outro utilizando modelo de dados relacional, são traduzidos para um modelo de dados comum e, finalmente, integrados em um sistema federado.

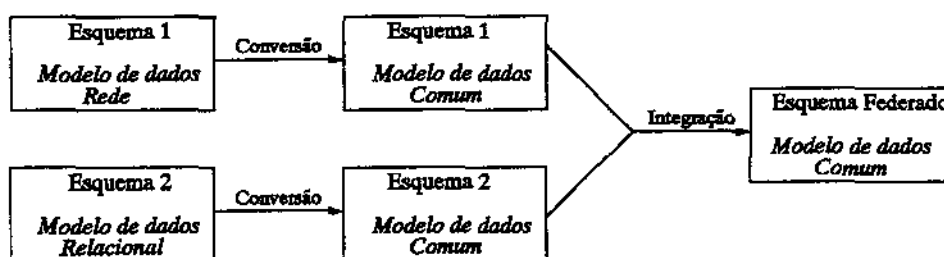


Figura 2.3: Exemplo de conversão de esquemas de bancos de dados.

Também relacionado a mudanças em esquemas de bancos de dados, mas voltados para bancos de dados orientados a objetos está o trabalho de Lerner et al. [LH90] que visa automatizar a modificação das instâncias face à mudança de esquema. Isto é realizado por um programa que compara as versões de esquemas e produz um outro programa que

efetua a migração dos dados. Modificações de esquema são armazenadas em tabelas que são processadas para permitir a migração automaticamente e que podem ser atualizadas diretamente pelo administrador de banco de dados. Os três problemas principais enfrentados em [LH90] foram: (1) identificar os tipos de mudanças que podem ser feitas nas classes, (2) definir o conjunto *default* de transformações indicando como os objetos existentes devem ser transformados para cada tipo de mudança da classe e (3) prover um mecanismo para que o administrador possa redefinir as transformações *default* sem violar as regras do modelo.

Outra operação comum em um processo de integração de bancos de dados é a junção de esquemas (*merge*). Buneman et al. [BDK92] mostram uma caracterização generalizada de esquemas de bancos de dados que permite aos usuários dar uma definição mais natural para esta operação em termos das informações contidas nos próprios esquemas a serem juntados. O objetivo é que a junção de esquemas seja independente da ordem com que a operação é processada.

Em um nível mais baixo, o trabalho de Qian [Qia96] propõe um formalismo baseado em *tipos abstratos de dados* (ADTs) para tratar conversão, reestruturação, integração, remoção de anomalias e redundâncias de esquemas, capturando as informações contidas nestes esquemas para construção dos novos.

Ferramentas Baseadas em Cliente-Servidor

Le et al. [LS97] apresentam uma ferramenta para programas de migração de dados em ambientes heterogêneos visando compartilhamento de recursos. A ferramenta proposta é baseada no modelo cliente-servidor e no uso de chamadas remotas de procedimentos pela interconexão de sistemas heterogêneos e distribuídos através de interfaces.

Conversões de Aplicações

Considerando a conversão de aplicação, Su et al. [SLL81] propõem uma metodologia de conversão resumida na figura 2.4. De acordo com esta figura, um banco de dados origem *DBs* é mapeado para um destino *DBt* através de um conjunto de operadores de conversão *C*. Uma aplicação origem *Ps* opera sobre o banco de dados *DBs* para produzir algum resultado *Rs* que representa dados recuperados do banco de dados ou um novo estado do banco de dados produzido após operações de eliminação, inserção ou alteração. O objetivo do programa de conversão é produzir (semi) automaticamente um programa *Pt* que opere sobre o banco de dados *DBt* e que produza os resultados *Rt* de tal forma que *Rs* seja equivalente a *Rt*, isto é, que a representação semântica de *Ps* possa ser transformada na representação semântica de *Pt*. No caso dos problemas abordados na dissertação, o *DBs* pode ser, por exemplo, um sistema legado e o *DBt* um sistema de informações geográficas.

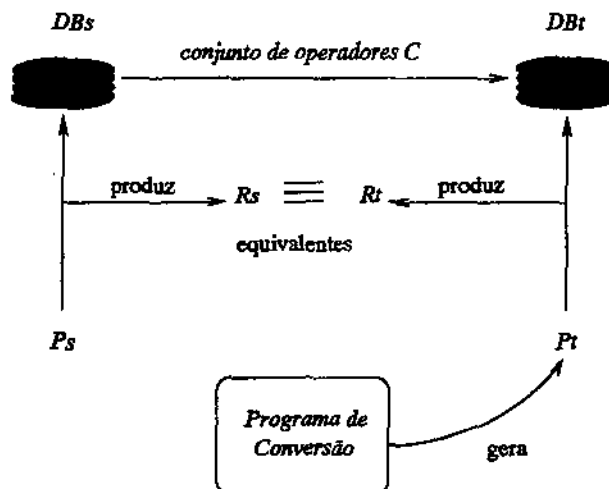


Figura 2.4: Conversão de aplicações.

Conversões de Consultas

Qian et al. [QR95] trata heterogeneidade segundo o prisma de processamento e conversão de consultas. Neste caso, o usuário tem uma visão orientada a objetos de um banco de dados relacional. Cada consulta OO é traduzida para uma representação canônica intermediária dedutiva de primeira ordem, que é por sua vez otimizada e mapeada em uma consulta relacional. A representação canônica é calculada uma única vez para todas as classes e atributos da visão OO.

Gerenciamento de Transações

Os trabalhos de Georgakopoulos [Geo91], Georgakopoulos et al. [GRS91] e Triantafillou [Tri96] estão voltados ao problema do gerenciamento de transações em ambientes distribuídos. Trata-se, assim, de um outro nível de preocupação (tolerância a falhas e recuperação). Estas pesquisas propõem novas formas de gerenciar transações em ambientes de SDBF de forma a garantir as propriedades de transações em ambientes com um único SGBD.

Segurança

O trabalho de Gong et al. [GQ94] trata da questão de segurança em bancos de dados heterogêneos. Dois princípios básicos devem ser respeitados para que os bancos de dados locais mantenham-se seguros sem comprometer a autonomia dos mesmos [GQ94]. São eles:

- Princípio da autonomia: Qualquer acesso permitido nos bancos de dados locais deve ser permitido nas operações entre os bancos de dados locais providos de mecanismos de segurança.
- Princípio da segurança: Qualquer acesso não permitido nos bancos de dados locais não deve ser permitido nas operações entre os bancos de dados locais providos de mecanismos de segurança.

Para que estes dois princípios sejam mantidos, Gong et al. [GQ94] apresentam os problemas que devem ser resolvidos sendo, na maioria, problemas NP-completos. Em outras palavras, manter a segurança dos bancos de dados locais em um ambiente integrado sem comprometer a autonomia não é trivial.

2.4 Sistemas Legados

Como já foi mencionado anteriormente, os componentes de um SBDF podem ser tanto SGBDs como sistemas legados.

De forma mais detalhada, um *sistema de informação legado*, freqüentemente chamado de *sistema legado*, consiste geralmente de um sistema de grande porte, com milhões de linhas de código, antigo, autônomo, organizado em algum sistema de arquivos, não sendo gerenciado por um SGBD e que resiste a evoluções e modificações [BS95].

Para determinar a maneira mais eficaz de resolver o problema de sistemas legados é necessário identificar, primeiramente, o grau de organização existente, ou seja, o tipo de arquitetura presente nestes sistemas. De acordo com Brodie et al. [BS95], existem três camadas básicas da arquitetura de um sistema legado: (1) interface, (2) aplicações e (3) serviços de banco de dados. Assim, o grau de organização de um sistema legado é determinado, dentre outras, pela capacidade de decomposição desta arquitetura que pode ser: (1) *totalmente decomposta*, (2) *parcialmente decomposta* ou (3) *não decomposta*. A figura 2.5, adaptada de [BS95], ilustra estes três tipos de arquitetura.

Em sistemas com arquiteturas totalmente decompostas, pode-se diferenciar as interfaces, aplicações e serviços de banco de dados, considerando-as como camadas distintas. Nas arquiteturas parcialmente decompostas, apenas as interfaces são vistas como uma camada separada, sendo as aplicações e os serviços de banco de dados considerados de forma conjunta. Arquiteturas não decompostas, por sua vez, não apresentam diferenciações entre interfaces, aplicações e serviços de banco de dados. O grau de organização de um sistema legado diminui à medida que sua arquitetura perde a capacidade de decomposição.

Desta forma, existem basicamente três enfoques no tratamento de sistemas legados, dependendo do seu grau de organização:

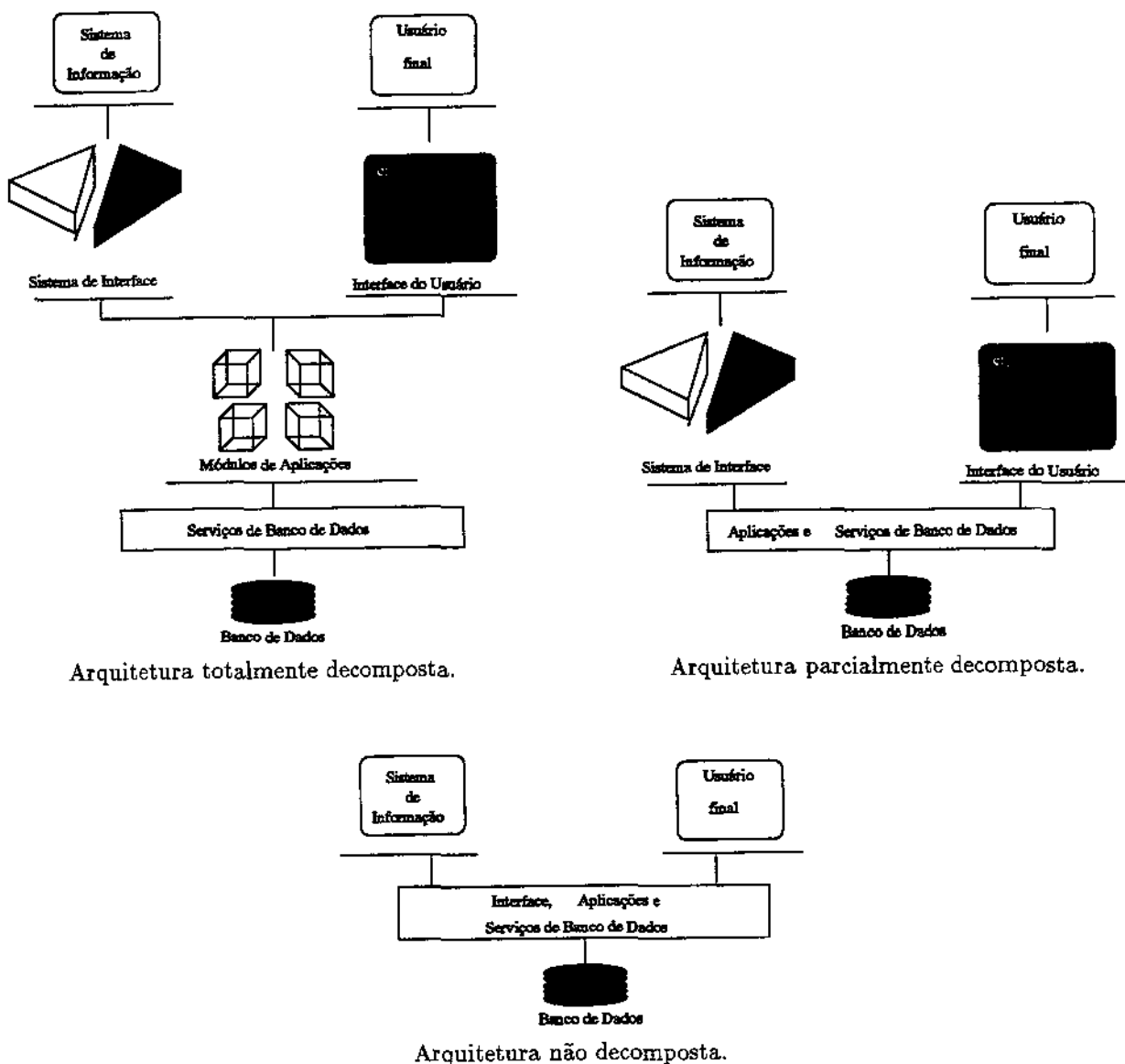


Figura 2.5: Arquitetura dos sistemas legados.

- Migração dos dados legados para um outro sistema que suporte modificações de forma mais organizada;
- Construção de um novo sistema que substitua o sistema legado;
- Conversão de esquema.

A opção de migrar os dados legados para uma outra forma de armazenamento, consiste em partir de um *sistema legado* e chegar a um *sistema destino* que possua as mesmas funcionalidades do anterior.

Devido à falta de documentação dos sistemas legados, o que acontece em muitos casos é a criação de um sistema completamente novo, sem aproveitar o que já existe e com uma alta probabilidade de fracassar, além da necessidade de altos investimentos.

Se a arquitetura do sistema legado permitir a documentação de um esquema básico, através de técnicas de engenharia reversa, este esquema pode ser convertido para um outro esquema mais robusto e flexível. A conversão de esquema e a migração dos dados legados, muitas vezes, são usadas comitadamente.

2.4.1 Trabalhos Correlatos

Não há muitas propostas para SBDF envolvendo sistemas legados. Alguns trabalhos já citados, relacionados com conversão de esquemas, podem auxiliar no tratamento de aplicações e dados legados, mas existem outros trabalhos que estão diretamente envolvidos com sistemas legados, sejam com propostas metodológicas de migração de dados e aplicações [BS95, NEK94], sejam com técnicas de engenharia de software para mapear ou migrar os dados [PH95, MNB⁺94, AMR94, Rob97].

Do ponto de vista de propostas metodológicas, Brodie et al. [BS95] apresentam duas estratégias de migração de sistemas legados bastante interessantes para esta dissertação: *Cold Turkey* e *Chicken Little*. Usando a estratégia *Cold Turkey*, o sistema legado é totalmente recodificado, visando a geração do sistema destino. No entanto, esta opção traz um risco de insucesso muito grande causado, dentre outros, pelos itens que se seguem:

- Na construção de um novo sistema, geralmente são prometidas funções adicionais, buscando justificar o investimento necessário. Estas novas funções aumentam o risco de falha do novo sistema.
- Durante o processo de recodificação, as novas funções que continuam sendo inseridas no sistema legado devem ser incorporadas ao sistema destino, causando mudanças no projeto inicial de recodificação.

- Raramente existe uma especificação do sistema legado. Além disto, geralmente não se tem conhecimento dos sistemas dependentes do sistema legado a ser migrado.
- Alguns sistemas legados dependem de grandes volumes de dados. A transferência destes dados para o sistema destino pode ser muito demorada e os dados não ficam, durante esse processo, disponíveis para as aplicações.
- Antes de começar a migração propriamente dita, deve-se analisar e entender o funcionamento do sistema legado como um todo. Porém, o sistema pode ser tão grande e tão complexo que esta análise é muito demorada e, enquanto isso, nenhuma migração é executada.

A segunda alternativa proposta em [BS95] é realizar a migração de sistemas legados através de pequenos passos incrementais. Este processo, chamado de *Chicken Little*, permite estimar o risco de falhas em cada passo, facilitando a correção de erros e aumentando as chances de sucesso.

Cada passo da migração requer uma alocação pequena de recursos e pouco tempo para ser concluída. Desta forma, *Chicken Little* é classificada como uma *metodologia incremental de migração* sendo considerada muito mais controlável e administrável [BS95]. Vários dos problemas apresentados pela metodologia *Cold Turkey* são também enfrentados pela metodologia *Chicken Little*, porém estes problemas ficam restritos a cada passo da migração. Se um passo da migração falha, apenas este passo deve ser repetido ou ajustado e não o projeto inteiro.

O trabalho de Ning et al. [NEK94] apresenta uma opção de integração híbrida, ou seja, parte do sistema legado é substituído por um novo sistema e a outra parte é usada mantendo-se a forma original. Neste caso, componentes funcionais do sistema legado são reconhecidos, recuperados e adaptados para serem utilizados em um novo sistema desenvolvido. Isto é chamado de *reusable component recovery* (RCR).

A proposta de Pernul et al. [PH95] combina técnicas de engenharia de software (*engenharia reversa* e *engenharia clássica*) para se realizar migrações, utilizando três tipos de modelos: estático, funcional e dinâmico. O modelo estático representa um diagrama entidade-relacionamento com as características do sistema legado utilizando engenharia reversa, ou seja, a partir do sistema legado constrói-se o diagrama entidade-relacionamento. Em um segundo passo, o modelo funcional é construído através da engenharia clássica, isto é, começa-se a construção deste modelo partindo-se do modelo estático e o resultado é um DFD (Diagrama de Fluxo de Dados) apresentando as funcionalidades do banco de dados. Por último, também através da engenharia clássica, o modelo dinâmico é gerado descrevendo o ciclo de vida de um determinado objeto e os métodos associados a ele.

Markosian et al. [MNB⁺94], também usando técnicas de engenharia de software, descrevem um sistema para automatizar o processo de reengenharia em grandes sistemas legados. O sistema é composto pelos seguintes módulos: Dialect, Refine, WorkBench e Entrevista. O módulo Dialect é um compilador gramatical responsável pela divisão e especificação gramatical do sistema legado. O Refine corresponde a um software de análise e transformação responsável por especificar o sistema legado em uma linguagem de alto nível, aplicar modelos orientados a objetos a este sistema, aplicar as transformações necessárias e os padrões de engenharia de software. O módulo WorkBench é usado na reutilização de software para ferramentas de engenharia reversa, estabelecendo a navegação entre o sistema legado e o projeto que está sendo desenvolvido. Por fim, o módulo Entrevista é uma ferramenta de interface para facilitar a manipulação do sistema legado.

Aiken et al. [AMR94] utilizam a engenharia reversa para resolver o problema de sistemas legados do Departamento de Defesa dos EUA. Este trabalho propõe um framework para aplicação de engenharia reversa em sistemas com, aproximadamente, 1,4 bilhões de linhas de código associadas a milhares de sistemas de informações heterogêneas.

Por fim, a proposta de Robertson [Rob97] baseia-se em *programação orientada a objetos, reflexão* e uma *linguagem de programação de domínio específico* de forma a construir uma ponte entre os sistemas legados existentes e um novo sistema baseado em cliente/servidor.

De todos os trabalhos aqui citados, diretamente relacionados com sistemas legados, a proposta metodológica Chicken Little [BS95] apresenta características mais consistentes para se processar uma migração. A proposta de Robertson [Rob97], baseada em programação, não considera o fato de se processar a migração através de passos incrementais e o enfoque básico é o uso de programação orientada a objetos, sem preocupação com a parte dos dados. Assim sendo, este enfoque não será considerado. O trabalho de Ning et al. [NEK94] baseia-se mais nas aplicações legadas e não apresenta soluções voltadas para os dados em si e, portanto, também não será considerado nesta dissertação.

Por outro lado, as propostas [PH95, MNB⁺94] e [AMR94] podem ser utilizadas durante a aplicação do processo Chicken Little para oferecerem soluções a alguns passos desta metodologia. Por exemplo, a análise incremental do sistema legado seguida da decomposição incremental de sua estrutura podem ser realizadas através da engenharia reversa. Outros passos como o projeto incremental das aplicações e do banco de dados destino podem ser realizados através da engenharia clássica.

A metodologia Chicken Little é um dos componentes básicos da solução proposta na dissertação para integração de sistemas legados e bancos de dados sendo, desta forma, detalhada a seguir.

2.4.2 Metodologia Chicken Little

A ligação entre o sistema legado e o sistema destino utiliza o conceito de *gateway*. Um gateway nada mais é do que um mediador, ou seja, um módulo de software colocado entre os sistemas legado e destino com o objetivo de estabelecer a comunicação entre eles. Dentre suas funções, o gateway é responsável por traduzir as requisições e os dados entre os sistemas, assegurar que uma certa modificação feita por um sistema seja realizada e coordenar os sistemas para assegurar atualizações consistentes. A escolha do local para instalação do gateway é crítica e pode influenciar fortemente a complexidade da migração.

Dois conceitos são importantes para a metodologia Chicken Little [BS95]: *forward gateways* e *reverse gateways*. Forward gateways permitem que aplicações legadas acessem os dados destino, ou seja, os dados que já foram migrados. Por outro lado, reverse gateways permitem que as aplicações destino acessem os dados legados.

Algumas ações são necessárias para se estabelecer a migração, tais como: (1) reduzir radicalmente as funções e os dados a serem migrados, (2) desenvolver um projeto global do sistema destino, (3) estabelecer o ambiente destino e (4) estabelecer o ambiente computacional para a migração. Além disto, deve-se assegurar que aspectos como segurança, robustez e acesso para leitura continuem sendo oferecidos de forma a suportar as necessidades da organização durante a migração.

Tendo finalizado esta parte inicial, alguns passos são definidos pela metodologia Chicken Little [BS95]:

- Passo 1: Análise incremental do sistema legado.
- Passo 2: Decomposição incremental da estrutura do sistema legado.
- Passo 3: Projeto incremental da interface destino.
- Passo 4: Projeto incremental das aplicações destino.
- Passo 5: Projeto incremental do banco de dados destino.
- Passo 6: Instalação incremental do ambiente destino.
- Passo 7: Criação e instalação incremental dos gateways necessários.
- Passo 8: Migração incremental dos dados legados.
- Passo 9: Migração incremental das aplicações legadas.
- Passo 10: Migração incremental da interface legada.
- Passo 11: Transmissão incremental do sistema legado para o sistema destino.

Estes passos não necessariamente precisam ser executados nesta ordem. Os passos 2 até 6 podem ser executados em qualquer ordem. Todavia, o passo 7 apenas poderá ser executado depois do passo 6 e o passo 2, depois da execução do passo 1. Da mesma forma, o passo 11 deverá ser executado depois de todos os outros.

Algumas variações destes passos podem ocorrer quando é considerado o tipo de migração que se está utilizando. Segundo Brodie et al. [BS95], pode-se caracterizar a migração em três tipos: (1) *forward migration*, (2) *reverse migration* e (3) *general migration*. A primeira é caracterizada por processar primeiramente a migração dos dados, seguida pela instalação de um *forward gateway*. Por outro lado, a *reverse migration* adia a migração dos dados para o final do processo e, enquanto isso, um *reverse gateway* é usado para permitir o acesso aos dados legados. A terceira opção de migração é uma combinação das duas primeiras e é necessário um gateway mais sofisticado.

Baseado nos trabalhos encontrados na literatura e nas análises feitas neste capítulo, é possível propor uma metodologia que permita processar a integração de bancos de dados heterogêneos com mais segurança e controle, considerando ainda a presença de sistemas legados como componentes da integração. O capítulo seguinte apresenta a metodologia proposta, que descreve de maneira detalhada todas as seqüências de passos e atividades necessárias para se estabelecer um sistema de bancos de dados integrado.

2.5 Resumo do Capítulo

Este capítulo apresentou alguns dos principais trabalhos encontrados na literatura referentes a bancos de dados heterogêneos, destacando os sistemas de bancos de dados federados, além de analisar as diferentes abordagens de tornar os sistemas legados utilizáveis e integráveis.

As soluções propostas na literatura para integrar bancos de dados heterogêneos variam segundo dois eixos. O primeiro referente ao tipo de integração: real ou virtual; e o segundo referente ao grau de acoplamento: fracamente acoplado ou fortemente acoplado.

No caso de integração real de dados, a partir dos esquemas dos bancos de dados locais, gera-se um esquema único e integrado, capaz de representar todos os dados dos esquemas locais. Em contrapartida, a integração virtual não oferece um esquema único mas sim visões integradas dos dados que permitem, a partir de mapeamentos com os bancos de dados locais, processar as consultas submetidas a estas visões.

Já o grau de acoplamento da integração está relacionado com a autonomia dos bancos de dados locais. Uma integração fortemente acoplada possui um bom desempenho no processamento de consultas mas compromete a autonomia dos bancos de dados locais. Por outro lado, uma integração fracamente acoplada mantém esta autonomia, transferindo algumas responsabilidades aos usuários mas, com isto, piorando o desempenho.

Como opção de integração de bancos de dados heterogêneos, destacam-se os sistemas de bancos de dados federados (SBDF) que buscam, basicamente, resolver a heterogeneidade entre seus componentes sem ferir a autonomia. Para isto, as ferramentas utilizadas para se estabelecer a federação são: mediadores, tradutores/adaptadores e visões.

As soluções dispensadas aos sistemas legados seguem três enfoques principais: migração dos dados legados para um novo sistema, construção de um novo sistema ou conversão de esquema. A escolha por uma destas soluções é influenciada pelo tipo de arquitetura do sistema legado que pode ser: totalmente decomposta, parcialmente decomposta ou não decomposta. Escolhida a solução desejada, esta pode ser executada de maneira mais abrupta ou através de passos incrementais como os apresentados pela metodologia *Chicken Little*.

Capítulo 3

Metodologia Proposta

Este capítulo apresenta a metodologia proposta para integração de bancos de dados heterogêneos e sistemas legados. Esta metodologia é baseada nos seguintes conceitos: integração de bancos de dados, bancos de dados federados, sistemas legados e tradução de modelos.

3.1 Visão Geral

Como já foi visto, a integração de bancos de dados pode ser real ou apenas virtual. Nesta dissertação, a integração de bancos de dados é virtual. A metodologia de integração recebe como entrada o conjunto de esquemas locais, os sistemas legados, os requisitos e restrições dos usuários em relação à exportação de dados e as consultas e transações processadas por estes usuários. Como saída, o usuário tem à disposição, visões integradas de dados e mapeamentos entre visões e sistemas locais, permitindo o processamento de consultas envolvendo dados de vários sistemas, além de chamadas de consultas globais. A figura 3.1, adaptada de [BL86], estabelece as entradas e saídas da metodologia de integração.

No caso desta dissertação, os sistemas locais são bancos de dados heterogêneos e sistemas legados que precisam passar por uma fase de padronização e filtragem dos dados a serem integrados antes de servirem como entrada para o sistema integrado.

Desta forma, a metodologia proposta nesta dissertação, divide-se em três fases principais:

- Fase 1: Padronização das entradas do sistema integrado através do mapeamento dos modelos de dados de todos os sistemas de bancos de dados (SBD) e sistemas legados para um *modelo de dados canônico* (MDC).
- Fase 2: Definição dos esquemas exportados, que vão determinar o subconjunto de dados de cada esquema local a ser integrado.

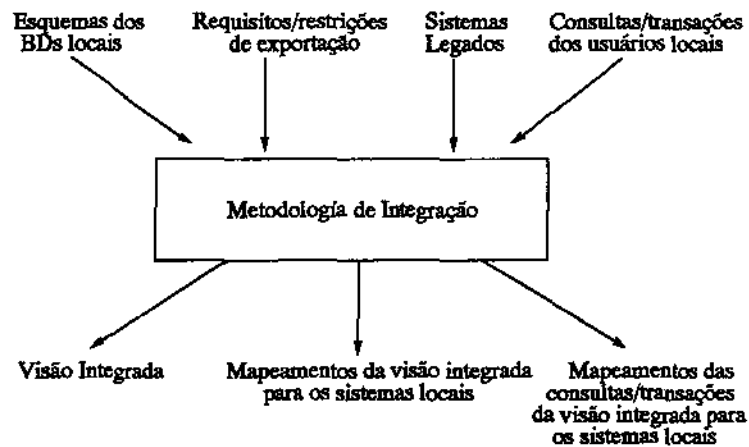


Figura 3.1: Entradas e saídas da metodologia de integração.

- Fase 3: Criação de um sistema integrado (SBDF) que possibilite o acesso a todos os esquemas exportados.

A figura 3.2 ilustra, em alto nível, as três fases descritas anteriormente.

A primeira fase do processo de integração inicia-se com a escolha e análise dos esquemas componentes a serem integrados, definindo as prioridades de integração. Feito isto, estes esquemas são representados utilizando um *modelo de dados canônico*. Este modelo canônico facilita o tratamento da heterogeneidade entre os componentes e possibilita armazenar informações semânticas que não estão representadas nos esquemas locais de cada banco de dados.

Os sistemas legados necessitam de um trabalho adicional, dependendo do grau de organização e de sua arquitetura - totalmente, parcialmente ou não decomposta (seção 2.4). Este trabalho adicional é responsável por definir um esquema inicial dos dados em questão para que os sistemas legados possam também ser mapeados para o MDC ou, se possível, definir um esquema inicial já utilizando o MDC.

Nesta dissertação, o MDC é composto pelo modelo relacional [Cod70], acrescido de informações semânticas adicionais. A opção pelo modelo relacional foi feita baseada no fato de que a grande maioria das aplicações atuais já utiliza este modelo, além da sua ampla utilização por SGBDs comerciais. Assim, a metodologia proposta fica mais próxima da realidade das organizações. As informações adicionais são usadas para facilitar o tratamento da heterogeneidade, já que o modelo relacional não possui muitos recursos para armazenar informações semânticas.

Depois de padronizados os modelos de representação dos bancos de dados componentes e dos sistemas legados, passa-se para a segunda fase de integração, responsável por definir o subconjunto de dados de cada esquema a ser integrado, chamado de *esquema exportado*.

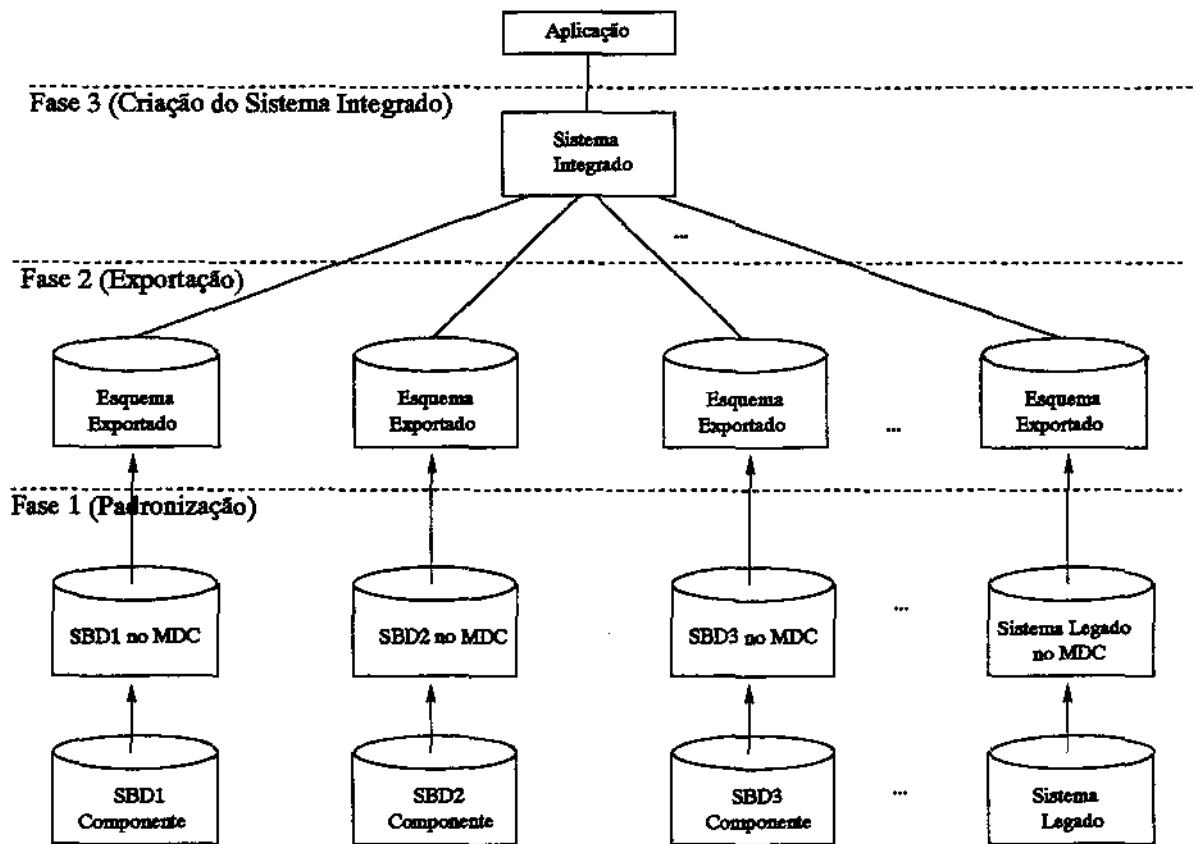


Figura 3.2: Fases do processo de integração de bancos de dados heterogêneos e sistemas legados. Fase 1: Padronização; Fase 2: Exportação; Fase 3: Criação do sistema integrado.

	Fracamente Acoplada	Fortemente Acoplada
Autonomia dos BDs locais	Alta	Moderada
Participação dos usuários no gerenciamento das transações	Alta	Baixa
Facilidade de inserção de novos BDs componentes na federação	Alta	Baixa
Adaptação às modificações nos BDs componentes da federação	Alta	Baixa
Complexidade de construção da federação em relação ao número de componentes a serem federados	Baixa	Alta
Tipos de operações permitidas	Consulta	Consulta e atualização

Tabela 3.1: Principais diferenças entre federações fracamente acopladas e fortemente acopladas.

Feito isto, passa-se para a fase de criação do sistema integrado. Seguindo a classificação proposta por [SL90] e apresentada na seção 2.1, um sistema integrado pode ser *não federado* ou *federado*. Este último, por sua vez, pode ser *fracamente acoplado* ou *fortemente acoplado*.

Os sistemas não federados são, claramente, os sistemas menos flexíveis e mais resistentes a modificações e, portanto, são os primeiros a serem descartados para compor a metodologia proposta nesta dissertação. Já a escolha por um sistema federado fracamente ou fortemente acoplado não é tão simples e necessita de um maior detalhamento de suas vantagens e desvantagens, como mostra a tabela 3.1.

Uma federação fortemente acoplada possui a vantagem de oferecer ao usuário um esquema integrado dos bancos de dados componentes, o que facilita o gerenciamento das operações globais e diminui a participação dos usuários neste processo. No entanto, modificações nos esquemas dos bancos de dados componentes desencadeiam um processo caro de adaptação da federação, pois o esquema integrado precisa ser revisto ou, dependendo da intensidade das modificações, precisa ser refeito, comprometendo as aplicações já desenvolvidas e em utilização pelos usuários. Isto significa que a autonomia dos bancos de dados componentes fica abalada, inviabilizando a criação de esquemas integrados em ambientes dinâmicos. Um outro problema é a difícil tarefa de construir um esquema integrado para um grande número de bancos de dados componentes, uma tarefa que cresce exponencialmente no número de esquemas a serem integrados [Zha97]. Os trabalhos de

Motz [Mot98] e Batini et al. [BLF97] são alguns exemplos da utilização de federações fortemente acopladas para realizar integração de bancos de dados.

Na federação fracamente acoplada, a autonomia dos bancos de dados componentes é mantida mas, por outro lado, requer do usuário uma maior habilidade e conhecimento dos bancos de dados envolvidos na federação, já que algumas decisões precisam ser tomadas diretamente por eles. Com isto, atualizações nos componentes da federação geralmente não são permitidos quando se está usando um acoplamento fraco, de modo a evitar o comprometimento da integridade dos dados. No entanto, uma federação fracamente acoplada permite que modificações estruturais possam ser feitas nos esquemas dos componentes da federação sem comprometer o sistema como um todo, já que estas modificações podem ser traduzidas (semi) automaticamente para a federação, mantendo as aplicações correntes em funcionamento.

A escolha feita nesta dissertação é usar uma federação fracamente acoplada como parte da metodologia de integração. Esta escolha foi feita considerando as efetivas necessidades de integração de dados dentro de uma organização. Geralmente são organizações de grande porte que necessitam de um grande esforço para permitir o compartilhamento de seus dados. Estas organizações mantêm esquemas de vários bancos de dados de diversos departamentos, seções e até mesmo companhias diferentes. Neste tipo de ambiente, modificações acontecem freqüentemente para suportar as necessidades dos usuários e o número de esquemas de bancos de dados tende a crescer. Isto requer um sistema integrado compatível com este ambiente, que ofereça facilidades de adaptação às modificações. Além disto, o número de esquemas a serem integrados não pode vir a ser um problema em potencial.

As organizações de pequeno porte que desejam proceder à integração de bancos de dados visando o crescimento futuro também podem optar pela federação fracamente acoplada mesmo que, neste caso, seja uma alternativa que talvez não compense por se tratar de integrações mais simples.

O algoritmo para a construção da federação adotado nesta dissertação é baseado no trabalho de Zhao [Zha97]. Este algoritmo foi escolhido pelas seguintes razões: estabelece regras claras e objetivas de integração de dados, definindo as responsabilidades da federação e dos usuários; e considera que os bancos de dados componentes obedecem ao modelo relacional, o mesmo modelo adotado aqui para compor o modelo de dados canônico.

Embora exista uma ordem de processamento das três fases da metodologia, raramente elas serão executadas uma única vez, tendo em vista a complexidade destes passos. De fato, o mais provável é que ocorram repetidas realimentações entre os passos, solicitados pelo usuário, até que se atinja a estabilização do esquema federado. A figura 3.3 ilustra a presença destes ciclos, chamados de *ciclos de realimentação* e representados pelas linhas

tracajadas. Assim, quando os esquemas exportados estão sendo construídos na fase de exportação, é possível voltar à fase de padronização. Da mesma forma, quando a federação já está sendo implementada, é possível voltar às fases de exportação e padronização sempre que necessário. Este aspecto oferece uma maior flexibilidade à metodologia e auxilia, principalmente, o tratamento de sistemas legados, já que estes sistemas não oferecem condições de serem compreendidos e manipulados de uma forma determinística.

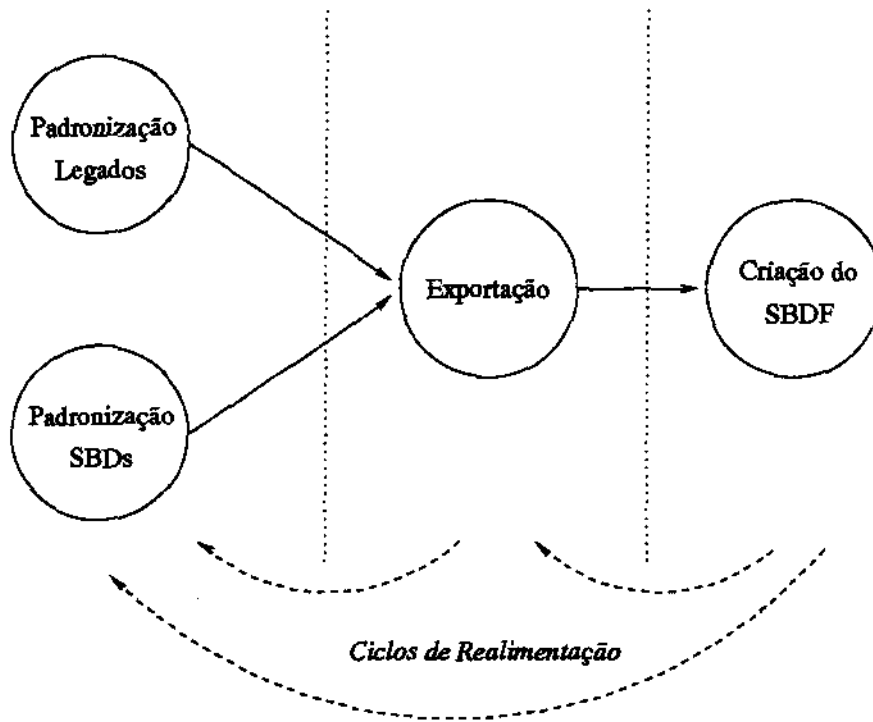
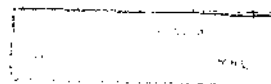


Figura 3.3: Processos da metodologia de integração.

3.2 Padronização e Exportação

As fases de padronização e exportação dos bancos de dados componentes da federação são os passos mais importantes do processo de integração. É nelas que se define quais dados serão compartilhados e para qual finalidade. Estas fases são baseadas na arquitetura de 5 níveis para descrição de dados proposta por Sheth et al. [SL90] e ilustrada pela figura 3.4, já embutindo as repetidas realimentações da figura 3.3.

O grupo de estudos de sistemas de bancos de dados ANSI/X3SPARC propôs uma arquitetura de descrição de dados de três níveis para a representação de SGBDs centralizados: *esquema conceitual*, *esquema interno* e *esquema externo*. [TK78]. O esquema



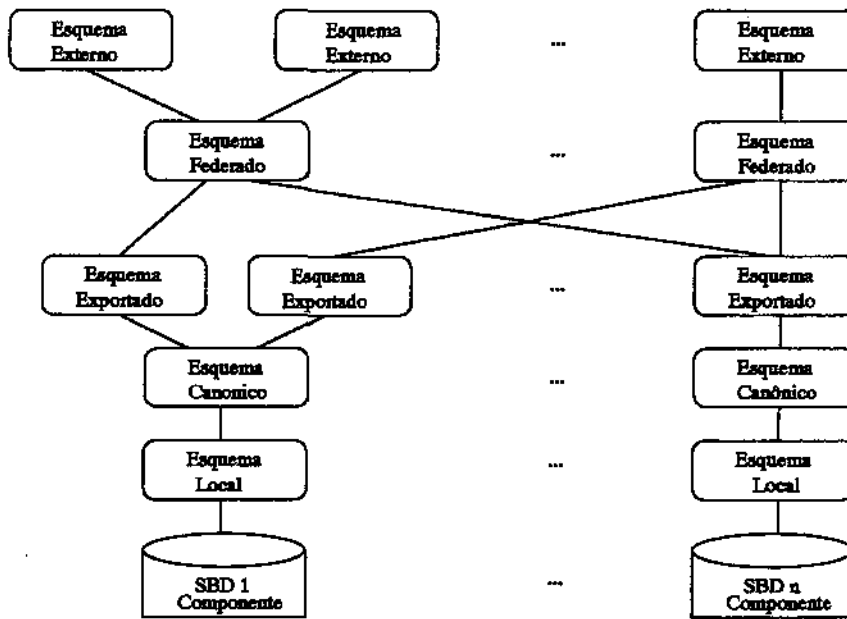


Figura 3.4: Arquitetura dos SBDFs - 5 níveis.

conceitual descreve a estrutura lógica dos dados e os relacionamentos existentes entre eles. Já o esquema interno descreve as características físicas dos dados, ou seja, informações sobre localidades dos registros, índices e formas de armazenamento. Por fim, é necessário determinar os subconjuntos dos dados que serão usados por cada usuário ou grupo de usuários. Esses subconjuntos são os esquemas externos que representam as visões que cada usuário terá do banco de dados. No entanto, estes três níveis não são suficientes para suportar as características adicionais dos SBDFs como distribuição, heterogeneidade e autonomia dos bancos de dados componentes. Desta forma, a arquitetura de representação dos dados é modificada, passando a ter os seguintes níveis [SL90]: *esquema local*, *esquema canônico*, *esquema exportado*, *esquema federado* e, finalmente, *esquema externo*.

Mapeando a figura 3.4 para a metodologia proposta, o esquema local corresponde ao esquema conceitual de cada banco de dados componente. Os modelos destes esquemas são traduzidos para o modelo de dados canônico (neste caso, o modelo relacional, mais algumas informações semânticas). A seguir, é necessário determinar quais dados de cada componente serão colocados à disposição da federação para serem compartilhados, definindo-se assim os esquemas exportados que são subconjuntos dos esquemas canônicos. O SBDF não considera quais os dados que estão efetivamente armazenados em cada componente da federação, mas apenas aqueles representados nos esquemas exportados.

Por fim, esquemas externos são oferecidos aos usuários, da mesma forma que nos sistemas de bancos de dados centralizados, permitindo um maior controle de acesso aos

dados federados e aumentado a eficiência no processamento de consultas.

Alguns destes passos podem ser omitidos, dependendo das características dos bancos de dados componentes da federação. Um exemplo disto é o caso em que um componente já esteja modelado de acordo com o modelo relacional. A seleção do que exportar pode ser omitida se todos os dados de um componente forem exportados para a federação.

Desta forma, a fase de padronização é responsável por agrupar os esquemas locais a serem integrados, definir as prioridades desta integração e traduzi-los para o modelo canônico. A partir daí, passa-se para a fase de exportação responsável por identificar quais dados serão integrados, tendo como resultado final os esquemas exportados. A terceira fase recebe os esquemas exportados como entrada e realiza a construção do SBDF. É importante ressaltar que quanto mais criterioso for o projetista do SBDF na escolha dos dados a serem integrados, mais eficiente tende a ser a federação. Por isto, é recomendável que apenas as integrações necessárias sejam realizadas, vetando aqueles dados que não necessariamente precisam compor o SBDF.

Os sistemas legados não podem, no entanto, ser tratados na fase de padronização da mesma forma que os sistemas de bancos de dados heterogêneos, porque possuem graus de organização diferentes entre si e necessitam de uma análise mais criteriosa. Assim, a aplicação da primeira fase da metodologia de integração (padronização) segue passos distintos, dependendo se o componente em questão é um sistema de banco e dados ou um sistema legado. As duas subseções seguintes especificam como o projetista do SBDF deve proceder em cada caso.

3.2.1 Padronização dos Sistemas de Bancos de Dados

O primeiro caso a ser considerado na fase de padronização está relacionado com os bancos de dados que possuem um SGBD associado e que portanto já obedecem a algum modelo de dados. Supõe-se que exista uma documentação destes bancos de dados em algum modelo de representação de dados como o modelo relacional, entidade-relacionamento (ER), entidade-relacionamento-estendido (ECR), orientado a objetos (OO) etc.

Neste caso, a padronização, resumida na figura 3.5, se dá pela aplicação de algoritmos de tradução dos modelos para o modelo canônico (modelo relacional, mais informações semânticas). O principal inconveniente do modelo relacional é a ausência de mecanismos de representação de todas as restrições estruturais e semânticas contidas em outros modelos, como por exemplo o ECR e o OO. Assim, as informações semânticas podem ser adicionadas ao modelo relacional através de relações suplementares, considerando também as informações obtidas com os usuários e com os administradores dos bancos de dados.

Os sistemas de bancos de dados que não possuem um grau de organização satisfatório para a aplicação dos algoritmos de tradução devem ser tratados como sistemas legados, já que requerem algum trabalho preliminar ao processo de tradução de modelo.

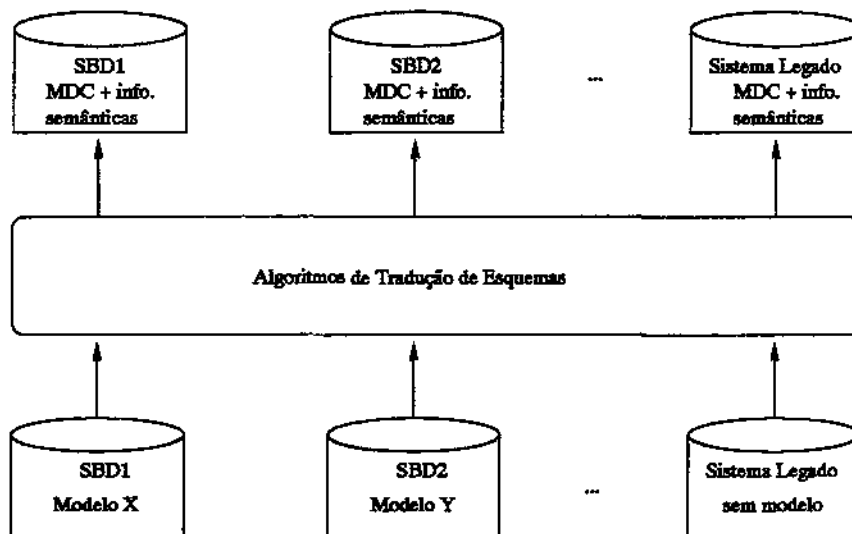


Figura 3.5: Tradução dos modelos dos bancos de dados locais para o modelo de dados canônico (MDC).

Existem vários algoritmos propostos na literatura que permitem a tradução entre modelos. Os trabalhos de Oliveira [Oli93] e Aguiar [Agu95] apresentam os algoritmos mais importantes. Considerando o escopo desta dissertação, apenas os algoritmos que oferecem como resultado o modelo relacional serão citados nesta seção.

Tradução Modelo ER → Modelo Relacional e Modelo ECR → Modelo Relacional

O modelo ECR é uma extensão ao modelo ER proposto por [Che76] que introduziu os conceitos de *categoria de generalização*, *categoria de subconjuntos* (agregação), atributos mono e multivalorados, dentre outros. Nestes modelos, o mundo real é modelado através de uma coleção de objetos básicos chamados *entidades*, interligados através de relacionamentos. Uma entidade é um objeto que existe e que é distinguível dos outros objetos por um conjunto específico de propriedades. Um *tipo-entidade* é um conjunto de entidades do mesmo tipo, ou seja, entidades que possuem propriedades similares. Já os relacionamentos representam associações entre vários tipos-entidades. Os *atributos* são propriedades que descrevem entidades e relacionamentos e podem ser classificados em *atributos básicos* ou *atributos adquiridos*. Atributos básicos são aqueles inerentes à entidade, independente dos relacionamentos de que esta participa e os atributos adquiridos são aqueles atributos não fundamentais à entidade e que dependem do contexto em que a entidade está inserida. Além disto, um atributo pode ser simples, composto, monovalorado ou multivalorado.

O modelo ECR oferece também o conceito de restrição de unicidade através do qual

pode-se determinar um atributo ou um conjunto de atributos que identificam univocamente uma entidade. Este conceito é similar ao de chave primária no modelo ER.

Outro conceito importante do modelo ECR é o de *categoria*. Uma categoria é uma abstração utilizada para o agrupamento de entidades pertencentes a um ou mais tipos-entidade, de acordo com os papéis que estas entidades desempenham nos relacionamentos. Através das categorias é possível representar subconjuntos de entidades (especialização) de um tipo-entidade ou agrupamentos de entidades (generalização) de diversos tipos-entidade.

Na tradução dos modelos ER e ECR para o modelo relacional, duas observações são importantes. A primeira é a quantidade de informações semânticas presentes nos modelos ER e ECR que precisam ser armazenadas paralelamente ao modelo relacional. O trabalho de Dogac et al. [DC83] mostra como transformar generalização e especialização de um modelo ECR em restrições de integridade referenciais. Isto é feito através da especificação de gatilhos (*triggers*) para definir ações que devem ser executadas sempre que um determinado evento de atualização de dados ocorrer. A segunda observação diz respeito à conversão de relacionamentos e categorias em relações já que o modelo relacional não possui construtores específicos para modelar relacionamentos. Essa conversão exige a duplicação de atributos de algumas relações em outras relações, para que as operações de junção entre elas possam ser executadas.

Vários trabalhos apresentam algoritmos para tradução do modelo ER para o modelo relacional [DA83, CNC83] e do modelo ECR para o modelo relacional [AP87, EWH85, EN89, Oli93]. Em geral, estes trabalhos baseiam-se nas seguintes correspondências:

- Cada tipo-entidade é traduzido para uma relação correspondente.
- Cada relacionamento gera uma relação correspondente cuja chave é composta pela concatenação das chaves dos tipos-entidades que participam do relacionamento.
- As abstrações de generalização/especialização são traduzidas através da criação de relações para as subclasses e superclasses tal que, os atributos das classes passam a ser atributos das relações.

Os trabalhos de Oliveira [Oli93] e Elmasri et al. [EWH85] utilizam o conceito de *surrogates* no caso de relações que não possuem atributos chave definidos. Devido à objetividade do trabalho de Oliveira, a seguir são apresentados os principais passos do seu algoritmo, ilustrando o uso do *surrogate* e detalhando o processo de tradução de modelos:

- Cada tipo-entidade do esquema ECR é mapeado para uma relação com o mesmo nome e a chave primária da relação é a chave do tipo-entidade que a originou. Caso

esta chave não exista, é criado um *surrogate* formado pela concatenação do nome da relação com o sufixo *ID*.

- Cada atributo monovalorado do tipo-entidade passa a ser atributo da relação e os atributos compostos são decompostos em atributos simples e incluídos na relação. Cada atributo multivalorado do tipo-entidade é colocado em uma relação à parte com o nome formado pela concatenação do seu próprio nome com o nome do tipo-entidade. Neste caso, a chave da relação é a concatenação do atributo multivalorado com o *surrogate* da relação que representa o tipo-entidade que possui tal atributo.
- Cada categoria de subconjunto é mapeada para uma relação cuja chave é a chave da categoria ou o *surrogate* criado por ela, e os atributos específicos das categorias devem ser incluídos na relação gerada.
- Cada tipo-entidade que compõe a categoria de generalização é mapeado para uma relação cuja chave é a chave associada à categoria ou ao *surrogate* criado para ela. Neste caso, os atributos da relação são todos os atributos da categoria mais os atributos específicos do tipo-entidade em questão.
- Cada relacionamentos binários M:N ou relacionamento de grau maior do que dois é mapeado para uma relação cuja chave é formada pela concatenação dos *surrogates* das relações que participam deste relacionamento.
- Cada relacionamento 1:1 ou 1:N que possui atributos próprios é mapeado para uma relação à parte. Caso este relacionamento não possua atributos próprios, então o *surrogate* da relação que representa a outra classe participante é incluído na relação que representa a classe com participação funcional. Neste caso, o *surrogate* incluído funciona como chave estrangeira.

Tradução Modelo Orientado a Objetos → Modelo Relacional

Aguiar [Agu95] apresenta uma síntese de dois importantes algoritmos [SL94, MYW⁺93] para a tradução de modelos de dados orientados a objetos (OO) para o modelo relacional. Para ilustrar este tipo de tradução, o algoritmo proposto em [SL94] é apresentado a seguir.

Sacramento et al. [SL94] utiliza o modelo de dados do sistema O_2 como entrada do seu algoritmo de tradução e realiza duas simplificações neste modelo para que a tradução possa ser realizada mais facilmente. A primeira delas é considerar todas as entidades como objetos e a segunda é introduzir o conceito de identificador como um atributo da classe. Isto permite distinguir um objeto de outro por suas propriedades descritivas. Além disto, um relacionamento é considerado como representado por um par de ponteiros inversos, em que objetos relacionados entre si apontam um para o outro. Isto possibilita determinar

relacionamentos de cardinalidade 1:1, 1:N e M:N, dependendo se o objeto possui um ponteiro para uma única ou para um conjunto de instâncias de outra classe.

A parte estática do modelo OO é mapeada para um modelo relacional, em termos de relações e restrições de integridade e a parte dinâmica é representada pelos cabeçalhos dos procedimentos que implementarão as operações definidas para as classes. Dois tipos de restrições de integridade são especificados: restrições de chave e restrições de integridade referencial. Os passos principais do algoritmo são:

- Cada classe pode ser mapeada para uma ou mais relações, dependendo da cardinalidade dos seus relacionamentos. Os atributos das relações são os mesmos atributos definidos na classe em questão.
- Cada relacionamento 1:1 ou 1:N é mapeado através da inserção de um atributo que representa o relacionamento como chave estrangeira em uma das relações. No caso de relacionamentos 1:N, este atributo deve ser inserido na relação associada à cardinalidade N.
- Cada relacionamento M:N é mapeado em uma relação a parte que possui como atributos as chaves primárias de ambas as classes relacionadas.
- Cada hierarquia de generalização é mapeada em várias relações sendo que a relação que representa a superclasse possui somente os atributos da superclasse enquanto que as relações que representam as subclasses possuem, além dos seus próprios atributos, a chave primária da superclasse.

Além destes algoritmos, outros algoritmos também são encontrados na literatura e podem ser utilizados para se chegar a um modelo intermediário e, a partir daí, obter o modelo relacional, como por exemplo: modelo Rede \rightarrow modelos ER e ECR [DA83, Fon92, Oli93]; modelo OMT \rightarrow modelo ECR [Agu95] e modelo OMT \rightarrow modelo O_2 [Agu95].

A opção por um ou outro algoritmo depende das características particulares de cada modelo que se deseja traduzir. É importante realizar uma análise geral do modelo a ser traduzido para escolher aquele algoritmo que consegue tratar de forma mais eficaz as suas peculiaridades.

3.2.2 Padronização dos Sistemas Legados

O segundo caso a ser tratado na fase de padronização está relacionado com os sistemas legados. A padronização neste caso depende do nível de organização do sistema legado em questão. A participação dos usuários e dos administradores destes sistemas é muito

Características	Soluções
<i>Grupo 1:</i> Arquitetura não decomposta e/ou sistemas fechados.	Reconstruir o sistema legado e, se possível, migrar os dados.
<i>Grupo 2:</i> Arquitetura parcial/totalmente decomposta e/ou sistemas de arquivos e/ou organização moderada dos dados e/ou SGBDs rudimentares e/ou documentação desatualizada ou insatisfatória.	1a. opção a) Manter o sistema legado que gerencia os dados; b) Utilizar um mediador. 2a. opção a) Substituir o sistema legado por um SGBD mais robusto; b) Realizar a migração dos dados.

Tabela 3.2: Características gerais dos sistemas legados e soluções aplicáveis a cada caso.

importante no sentido de fornecer informações semânticas sobre os dados e eliminar as ambigüidades que aparecem no decorrer do processo de padronização.

Dê acordo com as arquiteturas apresentadas pelos sistemas legados (seção 2.4), estes sistemas podem ser *totalmente decompostos*, *parcialmente decompostos* ou *não decompostos*. Para cada arquitetura, deve ser considerada uma forma de padronização diferente.

A tabela 3.2 mostra um resumo das principais características dos sistemas legados freqüentemente encontrados nas organizações e as soluções que podem ser a eles aplicadas com o objetivo de se chegar ao modelo de dados canônico.

O *Grupo 1* de sistemas legados corresponde àqueles que apresentam uma arquitetura não decomposta ou os sistemas fechados, com ausência total de documentação. Sistemas fechados são programas que não oferecem acesso direto aos seus arquivos através de programas externos ou macros. Estas características não permitem que um modelo de dados seja construído e tampouco que um mediador seja criado para fazer o acesso aos dados. Assim sendo, estes sistemas precisam ser refeitos para que sua desorganização não comprometa a eficiência do SBDF. Pode-se dizer que estes tipos de sistemas legados são os mais críticos na fase de padronização. Todas as aplicações e funcionalidades devem continuar sendo suportadas pelo novo sistema e todos os dados devem ser migrados, corrigindo-se os problemas de integridade que possam existir. Dados armazenados em tabelas de alguma planilha eletrônica são um exemplo de sistema legado que se encaixa no *Grupo 1*. Estes dados devem ser migrados para um sistema mais flexível e gerenciável para permitir que outros componentes da federação possam compartilhá-los.

Ao se refazer estes sistemas é importante modelar os dados já utilizando o modelo de dados canônico, evitando uma futura fase de tradução de modelos.

A metodologia *Chicken Little* (seção 2.4.2) é aplicável nos sistemas do *Grupo 1* para permitir a construção incremental do novo sistema e auxiliar no processo de migração

de dados, quando possível, além de permitir que as aplicações correntes continuem em funcionamento durante todo o processo.

Já o *Grupo 2* considera aqueles sistemas legados que apresentam uma arquitetura parcialmente ou totalmente decomposta, englobando os sistemas de arquivos, sistemas de bancos de dados rudimentares ou programas com pouca documentação ou documentação desatualizada, mas que possuem um nível de organização suficiente para o acesso aos dados através de mediadores, sem a necessidade de reestruturação total dos dados. Duas soluções são factíveis para o *Grupo 2*. A diferença primordial entre as duas soluções é a manutenção ou não dos programas que eles utilizam de gerenciamento de dados. A escolha por uma ou outra solução depende da decisão de se manter ou não o sistema legado. Se a decisão do projetista do SBDF for manter o gerenciamento dos dados através do sistema legado, o acesso a estes dados pode ser feito através de um mediador. Por outro lado, se houver a necessidade, por exemplo, de gerenciar os dados do sistema legado através de um SGBD mais robusto, deve-se optar por migrar os dados do sistema legado para o novo SGBD. Neste caso, a metodologia *Chicken Little* também pode ser utilizada para que a migração de dados seja incremental e segura.

É impossível descrever as características e soluções associadas a todas as variações possíveis de sistemas legados que podem ser encontradas nas organizações. O objetivo é destacar as características mais marcantes de cada sistema, sugerindo uma solução viável para cada caso.

3.2.3 Exportação

A fase de exportação é subsequente à padronização, embora sejam necessários vários ciclos entre exportação, testes, padronização até estabilizar o que deve ser exportado. Esta fase define o subconjunto de dados de cada esquema local a ser integrado. De cada esquema local são extraídos os dados que efetivamente irão participar da integração, formando-se assim, sub-esquemas.

A especificação de quais esquemas ou sub-esquemas devem ser exportados depende do conjunto de aplicações que se deseja executar sobre o SBDF e dos dados necessários à execução destas aplicações.

Assim sendo, esta fase depende totalmente da análise dos usuários e não é passível de automatização, sendo seu tratamento algorítmico bastante reduzido.

3.3 Criação do Sistema de Bancos de Dados Federados

Uma vez definidos os esquemas a serem integrados, o passo seguinte corresponde à construção do esquema federado. Em alguns casos, mais de um esquema federado pode ser disponibilizado, dependendo da necessidade dos usuários. A dissertação considera apenas um esquema federado, sem perda de generalidade. Nesta fase, como os sistemas legados já foram padronizados e já estão representados através do MDC, tanto os bancos de dados quanto os sistemas legados que compõem a federação serão chamados de *bancos de dados componentes*, apenas para simplificar a notação. Ressalte-se que os bancos de dados componentes constituem na verdade apenas os esquemas exportados.

A criação do sistema federado é baseada no trabalho de Zhao [Zha97]. Este trabalho propõe um algoritmo para construir um esquema de federação, chamado de *esquema coordenado*, permitindo o processamento de consultas envolvendo vários bancos de dados componentes autônomos, sem a necessidade de criação de um esquema integrado.

O esquema coordenado é responsável por resolver a heterogeneidade semântica em nível de nomes de atributo. Os demais tipos de heterogeneidade, como domínios de atributos, não são tratados pelo esquema, cabendo aos usuários a resolução de possíveis ambigüidades.

Zhao utiliza o conceito de *relação universal* [FMU82] durante a criação da federação. Isto implica em considerar todas as relações de todos os bancos de dados envolvidos como integradas em uma única relação. Para suportar as características desta federação, Zhao ainda propõe uma extensão à linguagem SQL.

As próximas seções descrevem inicialmente o trabalho original de Zhao e em seguida mostram quais as modificações devem ser feitas para atender os objetivos da construção de federação.

3.3.1 Algoritmo para Construção da Federação Proposto por Zhao

O algoritmo de Zhao para a construção do esquema federado recebe como entrada esquemas relacionais (no caso da metodologia, a entrada são os esquemas exportados a serem integrados utilizando o MDC). Como saída, o algoritmo constrói:

1. uma tabela contendo o nome e o significado semântico de cada atributo da federação (atributo federado), chamada de *Tabela Semântica*;
2. uma matriz de correspondência entre os atributos federados e os atributos dos esquemas exportados, chamada de *Matriz de Correspondência de Atributos*; e

3. uma tabela para cada esquema exportado contendo os nomes dos atributos tal qual eles aparecem nos esquemas exportados, os tipos destes atributos (chave, não chave, chave estrangeira) e um ponteiro que representa as dependências entre as relações criadas pelas chaves estrangeiras. Este ponteiro armazena nomes de atributos, a partir dos quais é possível realizar as operações de junção entre as relações dos esquemas. Cada tabela é chamada de *Tabela Exportada*.

A figura 3.6 apresenta os dados de entrada e saída para o algoritmo proposto por Zhao.

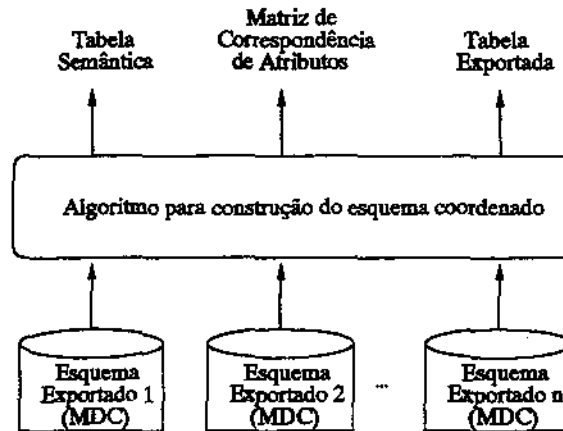


Figura 3.6: Entradas e saídas do algoritmo para construção do esquema coordenado.

A matriz de correspondência de atributos e a tabela semântica devem ser construídas simultaneamente da seguinte forma: Seja M a matriz de correspondência de atributos para um conjunto genérico n de esquemas exportados E_i ($i = 1..n$); e S a tabela semântica associada à matriz M . A matriz M possui os seguintes atributos: *Atributo-Federado*, *Componente- E_1* , *Componente- E_2* , ..., *Componente- E_n* e a tabela S possui os atributos: *Semântica* e *Atributo-Federado*. As construções de S e M seguem os seguintes esquemas:

S = (*Semântica*, *Atributo-Federado*), onde:

$S.Semântica$ = Significado semântico para um conjunto de atributos dos esquemas exportados.

$S.Atributo-Federado$ = Nome do atributo federado escolhido para referenciar um significado semântico único.

M = (*Atributo-Federado*, *Componente- E_1* , *Componente- E_2* , ..., *Componente- E_n*), onde $M.Atributo-Federado = S.Atributo-Federado$ e

$M.Componente- $E_i$$ = Nome do atributo no esquema E_i cujo significado semântico é igual a $S.Semântica$, relativo a $S.Atributo-Federado$.

<i>E.Nome</i>	<i>E.Tipo</i>	<i>E.Ponteiro</i>
$R_i.K$	chave primária	$R_i.E$
$R_i.A$	não chave	$R_i.K$
$R_i.E$	chave estrangeira	$R_j.K$

Figura 3.7: Lei de formação para as tabelas exportadas

A escolha do nome do atributo federado pode se dar de duas formas: (1) ou escolhe-se um nome genérico que represente o seu significado semântico ou (2) adota-se um nome já existente em algum esquema exportado. Alguns campos i,j da matriz de correspondência podem ficar vazios, informando que não existe relação entre o atributo federado da linha i com o esquema da coluna j .

Construídas a matriz de correspondência de atributos e a tabela semântica, passa-se à construção das tabelas que armazenam as informações dos esquemas exportados (tabelas exportadas). Estas tabelas possuem os seguintes atributos: *Nome*, *Tipo* e *Ponteiro*. Seja T uma tabela exportada para um esquema genérico E . Esta tabela tem o seguinte esquema:

T = (*Nome*, *Tipo*, *Ponteiro*), onde:
T.Nome = nome do atributo no esquema exportado E .
T.Tipo = tipo do atributo: [chave primária, chave estrangeira, não chave].
T.Ponteiro = nome de atributo que permite estabelecer as ligações entre as relações do esquema exportado, da seguinte forma: para um atributo $T.Nome$ de uma relação r do esquema exportado E ,
 SE $T.Tipo$ = [chave primária] ENTÃO
 $T.Ponteiro$ = nome da chave estrangeira da relação r .
 SE $T.Tipo$ = [não chave] ENTÃO
 $T.Ponteiro$ = nome da chave primária da relação r .
 SE $T.Tipo$ = [chave estrangeira] ENTÃO
 $T.Ponteiro$ = nome da chave primária da relação referenciada por este atributo.

A figura 3.7 sintetiza a lei de formação das tabelas exportadas tal que, para quaisquer relações distintas R_i e R_j de um esquema exportado E , temos: $R_i.K$ e $R_j.K$ são chaves primárias das relações i e j , respectivamente; $R_i.E$ é uma chave estrangeira qualquer da relação R_i e $R_i.A$ é um atributo não chave qualquer da relação R_i .

Com a matriz de correspondência de atributos, a tabela semântica e as tabelas exportadas, é possível processar consultas envolvendo todos os componentes da federação, sem se preocupar com a heterogeneidade de nomes entre eles. Para isto, Zhao propôs uma pequena extensão na linguagem de consulta SQL permitindo que o usuário informe o nome dos esquemas dos bancos de dados dos quais ele deseja extrair informações.

Devido às próprias características da federação fracamente acoplada, as extensões consideradas por Zhao contemplam apenas as consultas dos dados, desconsiderando os procedimentos de atualização. A alteração de dados neste tipo de federação não é aconselhável devido à autonomia dos bancos de dados componentes. Permitir atualizações neste tipo de ambiente deixaria o sistema integrado vulnerável a problemas de integridade de dados [SL90].

Assim, a extensão de Zhao ao SQL é realizada modificando-se apenas a cláusula *FROM* da expressão SQL da seguinte forma:

```
SELECT atributos federados
FROM nomes dos bancos de dados componentes
WHERE condições de seleção e junção entre os esquemas exportados
```

A tradução da consulta criada pelos usuários para subconsultas a serem processadas por cada banco de dados componente fica a cargo do SBDF. Em outras palavras, o SBDF fica responsável pelo tratamento da heterogeneidade estrutural de nomes entre os componentes, liberando os usuários desta tarefa.

O processo de tradução da consulta ocorre em 4 fases:

1. **Divisão da consulta:** Nesta fase é feita a divisão da consulta em subconsultas, de acordo com as condições de junção entre os bancos de dados especificados na cláusula *FROM*. Após esta fase, esta cláusula torna-se vazia.
2. **Conversão de atributos:** Para cada subconsulta criada, é feita a conversão dos atributos federados presentes nas cláusulas *SELECT* e *WHERE* para os atributos dos bancos de dados componentes, de acordo com a matriz de correspondência de atributos.
3. **Especificação de junções:** Nesta fase são especificadas todas as condições necessárias para se estabelecer as junções para acessar os atributos descritos na cláusula *SELECT*. As chaves estrangeiras são identificadas através dos ponteiros armazenados na terceira coluna das tabelas exportadas.
4. **Identificação dos objetos originais:** Finalmente, os nomes das relações envolvidas nas subconsultas são extraídos dos atributos descritos nas cláusulas *SELECT* e *WHERE* e armazenados na cláusula *FROM*.

O conjunto de passos a seguir representa o algoritmo proposto por Zhao que implementa as 4 fases do processo de tradução de consulta. Este algoritmo está ligeiramente modificado em relação ao original para facilitar o entendimento e evitar ambigüidades.

- 1) Represente a consulta federada através do vetor $Q(D, A, C)$, onde D é o conjunto de bancos de dados componentes (BD), A é o conjunto de atributos federados da cláusula SELECT e C é o conjunto de restrições e condições de inter-relacionamentos (junções).
- 2) Derive subconsultas $q_i(d_i, A, C)$ para $i = 1..n$, onde n é o número de subconsultas;
 - SE D não contém inter-relacionamentos ENTÃO,
 - d_i é o i ésimo banco de dados componente em D ;
 - SENÃO,
 - d_i é o conjunto de bancos de dados componentes dependentes em D , considerando os inter-relacionamentos necessários para responder Q , ou seja, para cada dependência em D ,
 - $d_i = \text{par de componentes dependentes}$.
- 3) PARA CADA subconsulta q_i , faça:
 - a) PARA CADA Atributo-Federado em A , faça:
 - PARA CADA banco de dados em d_i , faça:
 - SE Atributo-Federado existe na Matriz de Correspondência de Atributos ENTÃO,
 - {
 - converta Atributo-Federado para o BD.Relação.Atributo correspondente;
 - armazene o resultado em a_i ;
 - }
 - SENÃO,
 - ignore Atributo-Federado;
 - b) PARA CADA BD.Atributo-Federado em C , faça:
 - SE Atributo-Federado existe na Matriz de Correspondência de Atributos ENTÃO,
 - {
 - converta Atributo-Federado para o BD.Relação.Atributo correspondente;
 - armazene o resultado em c_i ;
 - }
 - SENÃO,
 - remova q_i , pois suas restrições não podem ser avaliadas.
- 4) PARA CADA subconsulta q_i , faça:
 - PARA CADA atributo em $d_i.a_i \cup d_i.c_i$ tal que $d_i.c_i$ é parte de uma condição de inter-relacionamento, percorra os ponteiros na tabela exportada para identificar as chaves estrangeiras dos inter-relacionamentos;
 - junte todas as condições de inter-relacionamento a c_i .
- 5) PARA CADA subconsulta q_i , faça:
 - PARA CADA BD.Relação em $a_i \cup c_i$,
 - junte BD.Relação à cláusula FROM de q_i .

O trabalho de Zhao é concluído com a indicação de como formular consultas. Neste aspecto, peca por não mostrar como a consulta direcionada ao SBDF é processada e retornada aos usuários. O processamento da consulta apresenta um problema, pois é dividida em subconsultas, que posteriormente precisarão ser integradas para obtenção da resposta. Esta dissertação analisa o problema e apresenta uma breve descrição de como resolver a questão.

Basicamente, três operações são suficientes para se chegar à resposta final do usuário: união, interseção e junção (dos resultados das subconsultas). As respostas retornadas pelas subconsultas são chamadas de subrespostas.

A operação de união é utilizada quando os resultados das subconsultas têm o mesmo esquema e são complementares, ou seja, cada subconsulta é responsável por obter os dados

requeridos pelo usuário armazenados em alguns componentes. A união das subrespostas forma o conjunto total dos dados que respondem à consulta. Por outro lado, a operação de interseção é utilizada quando uma subconsulta restringe o resultado da outra e ambas têm o mesmo esquema na subresposta. Por fim, a operação de junção é utilizada quando uma parte da informação (alguns atributos) requerida pelo usuário é respondida por uma subconsulta e o restante pela outra subconsulta. Assim, a resposta final é a junção destas informações segundo os atributos que elas têm em comum.

A seguir é apresentado um exemplo que constrói a matriz de correspondência de atributos, a tabela semântica e as tabelas exportadas de três esquemas, além de mostrar, passo a passo, o processamento de uma consulta utilizando o algoritmo de Zhao.

3.3.2 Exemplo de Aplicação do Algoritmo de Zhao

Para exemplificar o algoritmo de Zhao na criação do esquema coordenado, considere os esquemas exportados de três bancos de dados B_1 , B_2 e B_3 da figura 3.8.

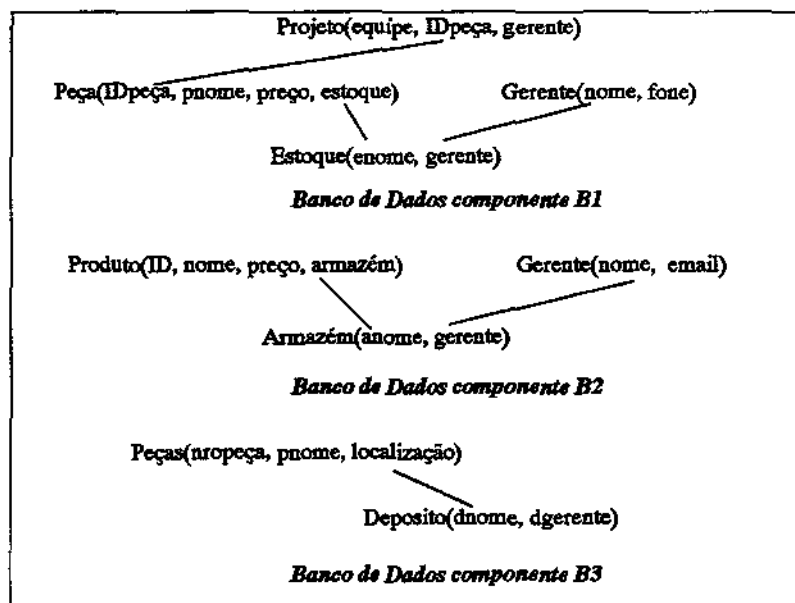


Figura 3.8: Esquemas exportados de três bancos de dados componentes.

O algoritmo de Zhao fornece para este exemplo, a tabela semântica 3.3 e a matriz de correspondência 3.4.

A primeira linha da matriz 3.4 mostra, por exemplo, que os atributos $Peça.IDpeça$ do componente B_1 , $Produto.ID$ do componente B_2 e $Peças.nropeça$ do componente B_3 são referenciados no esquema coordenado pelo atributo federado $IDPeça$, cujo significado

<i>Semântica</i>	<i>Atributo-Federado</i>
Identificador da peça da máquina	IDPeça
Nome de fábrica da peça	NomePeça
Preço em dólares da peça	PreçoPeça
Local de armazenamento da peça	EstoquePeça
Nome do estoque da peça	NomeEstoque
Identificador do gerente do estoque	GerenteEstoque
Nome da equipe projetista da peça	EquipeProjetista
Identificador da peça no projeto	IDPeçaProjeto
Nome do gerente da equipe projetista	GerenteProjetista
Nome do gerente do estoque	NomeGerente
Telefone do gerente	FoneGerente
Endereço eletrônico do gerente	E-mail

Tabela 3.3: Tabela semântica.

<i>Atributo-Federado</i>	<i>Componente-B₁</i>	<i>Componente-B₂</i>	<i>Componente-B₃</i>
IDPeça	Peça.IDpeça	Produto.ID	Peças.nropeça
NomePeça	Peça.pnome	Produto.nome	Peças.pnome
PreçoPeça	Peça.preço	Produto.preço	
EstoquePeça	Peça.estoque	Produto.armazém	Peças.localização
NomeEstoque	Estoque.enome	Armazém.anome	Depósito.dnome
GerenteEstoque	Estoque.gerente	Armazém.gerente	Depósito.dgerente
EquipeProjetista	Projeto.equipe		
IDPeçaProjeto	Projeto.IDpeça		
GerenteProjetista	Projeto.gerente		
NomeGerente	Gerente.nome	Gerente.nome	
FoneGerente	Gerente.fone		
E-mail		Gerente.email	

Tabela 3.4: Matriz de correspondência de atributos.

$B_1.Nome$	$B_1.Tipo$	$B_1.Ponteiro$
Projeto.equipe	chave primária	Projeto.IDpeça
Projeto.IDpeça	chave estrangeira	Peça.IDpeça
Projeto.gerente	não chave	Projeto.equipe
Peça.IDpeça	chave primária	Peça.estoque
Peça.pnome	não chave	Peça.IDpeça
Peça.preço	não chave	Peça.IDpeça
Peça.estoque	chave estrangeira	Estoque.enome
Estoque.enome	chave primária	Estoque.gerente
Estoque.gerente	chave estrangeira	Gerente.nome
Gerente.nome	chave primária	
Gerente.fone	não chave	Gerente.nome

Tabela 3.5: Tabela exportada referente ao componente de banco de dados B_1 .

semântico está armazenado na primeira linha da tabela semântica 3.3 (Identificador da peça da máquina). Já a terceira linha da matriz 3.4 mostra que os atributos $Peça.preço$ do componente B_1 e $Produto.preço$ do componente B_2 são referenciados pelo atributo federado $PreçoPeça$ e possuem o significado semântico que está armazenado na terceira linha da tabela 3.3 (Preço em dólares da peça). Como o componente B_3 não possui um atributo com este mesmo significado semântico, o valor da terceira coluna da matriz 3.4 para o atributo federado $PreçoPeça$ é nulo.

Seguindo o algoritmo de Zhao, passa-se para a construção das tabelas exportadas. Para o exemplo da figura 3.8, a tabela 3.5 refere-se ao componente B_1 , a tabela 3.6 ao componente B_2 e a tabela 3.7 ao componente B_3 .

Aplicando a lei de formação das tabelas exportadas de forma detalhada para o componente B_3 da figura 3.8, chega-se à tabela 3.7 da seguinte forma: o componente B_3 tem duas relações, "Peças" ($nropeça$, $pnome$, $localização$) e "Depósito" ($dnome$, $dgerente$). Os atributos destas relações estão na 1a. coluna da tabela 3.7.

A primeira linha da tabela 3.7 armazena a chave primária da relação "Peças". O valor do atributo $B_3.Ponteiro$ nesta linha deve ser então, o nome do atributo que é chave estrangeira na relação "Peças" (neste caso, o atributo $Peças.localização$). Este atributo, por sua vez, armazenado na terceira linha da tabela 3.7, possui como ponteiro o nome da chave primária da relação à qual ele se refere, ou seja, $Depósito.dnome$. A relação "Depósito" não possui atributo que seja chave estrangeira, portanto o valor de $B_3.Ponteiro$ da quarta linha da tabela 3.7 é vazio. Por fim, os atributos $Peças.pnome$ e $Depósito.dgerente$ são do tipo *não chave*, o que faz com que os valores dos ponteiros nestas linhas sejam as chaves

<i>B₂.Nome</i>	<i>B₂.Tipo</i>	<i>B₂.Ponteiro</i>
Produto.ID	chave primária	Produto.armazém
Produto.nome	não chave	Produto.ID
Produto.preço	não chave	Produto.ID
Produto.armazém	chave estrangeira	Armazém.anome
Armazém.anome	chave primária	Armazém.gerente
Armazém.gerente	chave estrangeira	Gerente.nome
Gerente.nome	chave primária	
Gerente.email	não chave	Gerente.nome

Tabela 3.6: Tabela exportada referente ao componente de banco de dados *B₂*.

<i>B₃.Nome</i>	<i>B₃.Tipo</i>	<i>B₃.Ponteiro</i>
Peças.nropeça	chave primária	Peças.localização
Peças.pnome	não chave	Peças.nropeça
Peças.localização	chave estrangeira	Depósito.dnome
Depósito.dnome	chave primária	
Depósito.dgerente	não chave	Depósito.dnome

Tabela 3.7: Tabela exportada referente ao componente de banco de dados *B₃*.

primárias de suas relações, ou seja, *Peças.nropeça* e *Depósito.dnome*, respectivamente.

De posse da matriz 3.4, da tabela semântica 3.3 e das tabelas exportadas 3.5, 3.6 e 3.7, é possível processar consultas sobre os bancos de dados da figura 3.8 utilizando apenas os nomes dos atributos federados. Assim, seja a seguinte consulta:

“Ache o preço das peças dos bancos de dados B_1 e B_2 que possuem os mesmos nomes que as peças armazenadas no armazém “d1” do banco de dados B_3 e imprima os nomes e emails dos gerentes dos estoques” [Zha97].

Utilizando a linguagem SQL estendida de Zhao, esta consulta pode ser expressa pela seguinte expressão:

```
SELECT [NomePeça], [PreçoPeça], [NomeGerente], [E-mail]
FROM  $B_1, B_2, B_3$ 
WHERE  $B_3$ . [NomePeça] = ( $B_1, B_2$ ). [NomePeça]
AND  $B_3$ . [NomeEstoque] = "d1"
```

O termo (B_1, B_2) na cláusula WHERE significa que junções são realizadas entre os bancos de dados B_3 e B_1 e também entre os bancos de dados B_3 e B_2 .

O primeiro passo para responder a esta consulta é dividi-la em subconsultas, de acordo com os relacionamentos entre os bancos de dados componentes, produzindo as seguintes subconsultas:

- Bancos de Dados B_3, B_1

```
SELECT [NomePeça], [PreçoPeça], [NomeGerente], [E-mail]
FROM
WHERE  $B_3$ . [NomePeça] =  $B_1$ . [NomePeça]
AND  $B_3$ . [NomeEstoque] = "d1"
```

- Bancos de Dados B_3, B_2

```
SELECT [NomePeça], [PreçoPeça], [NomeGerente], [E-mail]
FROM
WHERE  $B_3$ . [NomePeça] =  $B_2$ . [NomePeça]
AND  $B_3$ . [NomeEstoque] = "d1"
```

As figuras 3.9 e 3.10 apresentam o processamento destas subconsultas, seguindo o algoritmo proposto por Zhao [Zha97].

A seguir são apresentados alguns passos para a elaboração da figura 3.10:

Subconsulta criada	SELECT FROM WHERE AND	[NomePeça], [PreçoPeça], [NomeGerente], [E-mail] $B_3.[\text{NomePeça}] = B_1.[\text{NomePeça}]$ $B_3.[\text{NomeEstoque}] = \text{"d1"}$
Conversão de atributos	SELECT FROM WHERE AND	$B_3.\text{Peças.pnome}, B_1.\text{Peça.preço}, B_1.\text{Gerente.nome}$ $B_3.\text{Peças.pnome} = B_1.\text{Peça.pnome}$ $B_3.\text{Depósito.dnome} = \text{"d1"}$
Especificação de junções	SELECT FROM WHERE AND AND AND	$B_3.\text{Peças.pnome}, B_1.\text{Peça.preço}, B_1.\text{Gerente.nome}$ $B_3.\text{Peças.pnome} = B_1.\text{Peça.pnome}$ $B_3.\text{Depósito.dnome} = \text{"d1"}$ $B_1.\text{Peça.estoque} = B_1.\text{Estoque.enome}$ $B_1.\text{Estoque.gerente} = B_1.\text{Gerente.nome}$
Identificação dos objetos originais	SELECT FROM WHERE AND AND AND	$B_3.\text{Peças.pnome}, B_1.\text{Peça.preço}, B_1.\text{Gerente.nome}$ $B_3.\text{Peças}, B_1.\text{Peça}, B_1.\text{Estoque}, B_1.\text{Gerente}, B_3.\text{Depósito}$ $B_3.\text{Peças.pnome} = B_1.\text{Peça.pnome}$ $B_3.\text{Depósito.dnome} = \text{"d1"}$ $B_1.\text{Peça.estoque} = B_1.\text{Estoque.enome}$ $B_1.\text{Estoque.gerente} = B_1.\text{Gerente.nome}$

Figura 3.9: Processo de tradução da subconsulta envolvendo os componentes B_3 e B_1 .

Subconsulta criada	SELECT FROM WHERE AND	[NomePeça], [PreçoPeça], [NomeGerente], [E-mail] $B_3.[\text{NomePeça}] = B_2.[\text{NomePeça}]$ $B_3.[\text{NomeEstoque}] = \text{"d1"}$
Conversão de atributos	SELECT FROM WHERE AND	$B_3.\text{Peças.pnome}, B_2.\text{Produto.preço}, B_2.\text{Gerente.nome}, B_2.\text{Gerente.email}$ $B_3.\text{Peças.pnome} = B_2.\text{Produto.nome}$ $B_3.\text{Depósito.dnome} = \text{"d1"}$
Especificação de junções	SELECT FROM WHERE AND AND AND	$B_3.\text{Peças.pnome}, B_2.\text{Produto.preço}, B_2.\text{Gerente.nome}, B_2.\text{Gerente.email}$ $B_3.\text{Peças.pnome} = B_2.\text{Produto.nome}$ $B_3.\text{Depósito.dnome} = \text{"d1"}$ $B_2.\text{Produto.Armazém} = B_2.\text{Armazém.anome}$ $B_2.\text{Armazém.gerente} = B_2.\text{Gerente.nome}$
Identificação dos objetos originais	SELECT FROM WHERE AND AND AND	$B_3.\text{Peças.pnome}, B_2.\text{Produto.preço}, B_2.\text{Gerente.nome}, B_2.\text{Gerente.email}$ $B_3.\text{Peças}, B_2.\text{Produto}, B_2.\text{Armazém}, B_2.\text{Gerente}, B_3.\text{Depósito}$ $B_3.\text{Peças.pnome} = B_2.\text{Produto.nome}$ $B_3.\text{Depósito.dnome} = \text{"d1"}$ $B_2.\text{Produto.Armazém} = B_2.\text{Armazém.anome}$ $B_2.\text{Armazém.gerente} = B_2.\text{Gerente.nome}$

Figura 3.10: Processo de tradução da subconsulta envolvendo os componentes B_3 e B_2 .

Divisão da consulta: Na primeira fase do algoritmo são criadas as subconsultas que, neste caso, já estão representadas pelas figuras 3.9 e 3.10.

Conversão de atributos: Os atributos federados das cláusulas SELECT e WHERE são convertidos de acordo com a Matriz de Correspondência de Atributos. Por exemplo, o atributo federado *NomePeça* pode ser convertido tanto para $B_2.Produto.nome$ quanto para $B_3.Pecas.pnome$. Como nesta consulta o valor dos dois atributos é o mesmo, escolhe-se apenas um deles. A seção 3.3.3 comenta os problemas desta simplificação e apresenta alternativas para os casos em que os valores dos atributos convertidos são diferentes.

Especificação de junções: Após a conversão de todos os atributos federados, passa-se para a especificação das junções. Neste caso, para se chegar ao valor do atributo $B_1.Gerente.nome$, por exemplo, é necessário identificar as condições de junção existentes entre as relações do banco de dados B_1 , utilizando a tabela exportada 3.5. *Gerente.nome* é referenciado pela chave estrangeira *Estoque.gerente* da relação "Estoque". Portanto, a primeira condição de junção que deve ser colocada na cláusula WHERE da subconsulta é: $B_1.Estoque.gerente = B_1.Gerente.nome$. Já a relação "Estoque" possui o atributo *Estoque.enome* que é referenciado pela chave estrangeira *Peça.estoque* da relação "Peça". Assim, mais uma condição de junção é adicionada à cláusula WHERE: $B_1.Peca.estoque = B_1.Estoque.enome$.

Identificação dos objetos originais: Por fim, os nomes das relações originais e seus respectivos bancos de dados são colocados na cláusula FROM da subconsulta.

Depois de processadas as subconsultas, as subrespostas são unidas para formar a resposta final do usuário, já que estas consultas são complementares.

3.3.3 Modificações Propostas ao Algoritmo de Zhao

O trabalho de Zhao [Zha97], apesar de ser o único encontrado na literatura que apresenta um algoritmo para processar a integração de bancos de dados heterogêneos, possui algumas limitações, como por exemplo:

1. O usuário precisa saber os nomes dos bancos de dados onde estão as informações desejadas.
2. Algumas consultas causam respostas ambíguas porque, embora o usuário precise saber o nome do banco de dados onde está a informação desejada, este nome não é usado na cláusula SELECT.

3. A construção do esquema coordenado não considera a existência de chaves estrangeiras compostas, nem tampouco de atributos que são, ao mesmo tempo, chaves estrangeiras e chaves primárias.
4. Zhao não armazena informações relativas aos domínios dos atributos a serem integrados. Isto impossibilita um futuro tratamento da heterogeneidade entre domínios.
5. Outras ambigüidades podem ocorrer durante o processamento de consultas causadas por caminhos diferentes que podem ser seguidos para se obter uma informação.

Zhao cita em seu trabalho que a necessidade de se saber os nomes dos bancos de dados não chega a ser um problema quando se trata de bancos de dados que ele denomina competitivos (*competing databases*), ou seja, aqueles que armazenam informações semelhantes, mas para necessidades diferentes. Usando este conceito, o exemplo mostrado na seção 3.3.2 é um conjunto de três bancos de dados competitivos. No entanto, isto dificulta o processamento de consultas mais genéricas, como por exemplo: "Quais os nomes das peças cujos gerentes de estoque se chamam "José" ?

Para os objetivos desta dissertação, a necessidade imposta aos usuários de informar os nomes dos bancos de dados (problema 1) apenas restringe os tipos de consultas que podem ser processadas e não compromete a metodologia proposta. No entanto, seria de grande valia dotar o esquema coordenado de ferramentas que conseguissem processar consultas mais genéricas, sem se restringir a bancos de dados componentes específicos. Devido ao não comprometimento do objetivo central desta dissertação, que é a metodologia de integração, a construção destas ferramentas é proposta como extensão a este trabalho.

Embora o algoritmo de Zhao necessite que os nomes dos bancos de dados sejam especificados, estes nomes não são usados na cláusula SELECT (problema 2). Isto dificulta o processamento de consultas do tipo:

"Quais os preços das peças do banco de dados B_1 que possuem os mesmos nomes das peças do banco de dados B_2 ?"

Se fosse usado na cláusula SELECT apenas o nome do atributo *PreçoPeça*, o valor retornando poderia ser tanto o preço da peça no banco de dados B_1 quanto no banco de dados B_2 .

Para responder a esta consulta, é necessário informar o nome do banco de dados B_1 na cláusula SELECT, da seguinte forma:

```
SELECT  $B_1$ .[NomePeça],  $B_1$ .[PreçoPeça]  
FROM  $B_1$ ,  $B_2$ 
```

WHERE $B_1.[\text{NomePeça}] = B_2.[\text{NomePeça}]$

Isto gera uma simplificação no item (a) do terceiro passo no algoritmo para o processamento de consultas mostrado na seção 3.3.1. Com a presença dos nomes dos bancos de dados na cláusula SELECT, não é mais necessário realizar o item (a) para todos os bancos de dados armazenados em d_i . Assim, este item do algoritmo é substituído por:

- ```

3) PARA CADA subconsulta q_i , faça:
 a) PARA CADA BD.Atributo-Federado em A, faça:
 SE Atributo-Federado existe na Matriz de Correspondência de Atributos ENTÃO,
 {
 converta Atributo-Federado para o BD.Relação.Atributo correspondente;
 armazene o resultado em a_i ;
 }
 SENÃO,
 ignore Atributo-Federado;

```

O terceiro problema citado é a dificuldade de representação de chaves estrangeiras compostas nas tabelas exportadas. A figura 3.11 apresenta o esquema de um banco de dados componente  $B_2'$ , baseado no componente  $B_2$  do exemplo da seção 3.3.2. No componente  $B_2'$ , as relações "Produto" e "Armazém" possuem um novo atributo chamado "seção" e a composição dos atributos *Produto.armazém* e *Produto.seção* passam a ser a chave estrangeira para a relação "Armazém". O algoritmo de Zhao gera, para o componente  $B_2'$ , a tabela exportada 3.8.

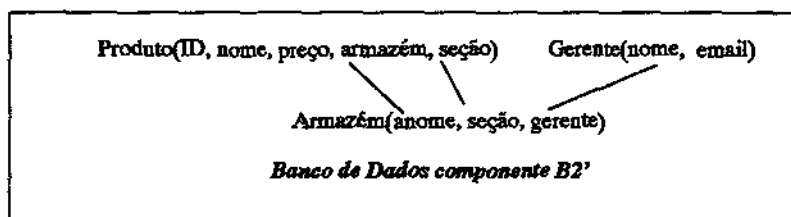


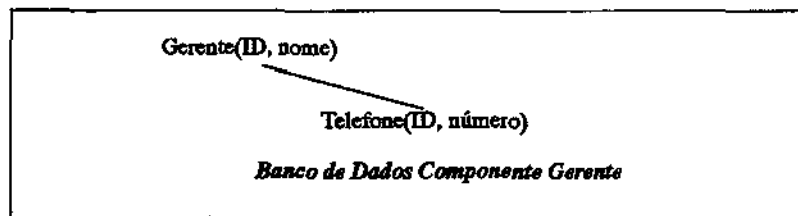
Figura 3.11: Esquema exportado do banco de dados componente  $B_2'$ , com chave estrangeira composta.

Considerando a mesma consulta usada no exemplo da seção 3.3.2 e o algoritmo de Zhao apresentado na seção 3.3, não seria possível processar as junções entre as relações "Produto" e "Armazém", já que faltaria a seguinte condição na cláusula WHERE da subconsulta referente aos componentes  $B_2$  e  $B_3$ :  $Produto.seção = Armazém.seção$  (denotando chave estrangeira  $\langle anome, seção \rangle$ ). A existência de chaves estrangeiras compostas leva ao surgimento de chaves primárias compostas. Neste caso, como pode ser visto na tabela 3.8, existe uma redundância de dados no campo  $B_2'.Ponteiro$  dos atributos que compõem a chave primária (6a. e 7a. linhas).

| $B_2'.Nome$     | $B_2'.Tipo$       | $B_2'.Ponteiro$ |
|-----------------|-------------------|-----------------|
| Produto.ID      | chave primária    | Produto.armazém |
| Produto.nome    | não chave         | Produto.ID      |
| Produto.preço   | não chave         | Produto.ID      |
| Produto.armazém | chave estrangeira | Armazém.anome   |
| Produto.seção   | chave estrangeira | Armazém.seção   |
| Armazém.anome   | chave primária    | Armazém.gerente |
| Armazém.seção   | chave primária    | Armazém.gerente |
| Armazém.gerente | chave estrangeira | Gerente.nome    |
| Gerente.nome    | chave primária    |                 |
| Gerente.email   | não chave         | Gerente.nome    |

Tabela 3.8: Tabela exportada referente ao componente de banco de dados  $B_2'$ .

Além deste problema, Zhao não considera a situação em que uma relação possui uma chave primária que também é chave estrangeira para uma outra relação, o que normalmente acontece na prática pela própria definição de chave estrangeira. A figura 3.12 exemplifica este fato.

Figura 3.12: Esquema exportado do banco de dados componente *Gerente*.

O atributo *Gerente.ID* é, ao mesmo tempo, chave primária da relação "Gerente" e chave estrangeira para a relação "Telefone". Na tabela exportada deste banco de dados, não seria possível determinar as dependências entre estas duas relações.

As soluções aqui propostas para estes dois problemas são as seguintes. Em relação aos problemas relacionados com chaves estrangeiras compostas, sugere-se alterar o campo *Ponteiro* das tabelas exportadas para que este possa armazenar uma lista de atributos e não apenas o nome de um único atributo. Assim, a tabela exportada para o banco de dados da figura 3.11 seria a tabela 3.9. O campo  $B_2'.Ponteiro$  do atributo *Produto.ID* armazena uma lista com todos os nomes das chaves estrangeiras da relação "Produto". Os demais atributos que compõem a chave primária possuem o campo  $B_2'.Ponteiro$  vazio, como é o caso do atributo *Armazém.seção*.

| $B_2'.Nome$     | $B_2'.Tipo$       | $B_2'.Ponteiro$                |
|-----------------|-------------------|--------------------------------|
| Produto.ID      | chave primária    | Produto.armazém, Produto.seção |
| Produto.nome    | não chave         | Produto.ID                     |
| Produto.preço   | não chave         | Produto.ID                     |
| Produto.armazém | chave estrangeira | Armazém.anome                  |
| Produto.seção   | chave estrangeira | Armazém.seção                  |
| Armazém.anome   | chave primária    | Armazém.gerente                |
| Armazém.seção   | chave primária    |                                |
| Armazém.gerente | chave estrangeira | Gerente.nome                   |
| Gerente.nome    | chave primária    |                                |
| Gerente.email   | não chave         | Gerente.nome                   |

Tabela 3.9: Tabela exportada referente ao componente de banco de dados  $B_2'$ , com suporte para chaves estrangeiras compostas.

Também é necessário tratar os casos em que uma relação possua atributos que são, ao mesmo tempo, chaves primárias e estrangeiras, como o esquema da figura 3.12. Para isto, basta considerar um novo valor para o campo *Tipo* das tabelas exportadas que é [primária-estrangeira].

Desta forma, os esquemas das tabelas exportadas passam por modificações. Seja  $T'$  uma tabela exportada para um esquema  $E$  de um banco de dados genérico. Com as modificações propostas, a tabela  $T'$  passa a ter o seguinte esquema:

$T'$  = (*Nome*, *Tipo*, *Ponteiro*), onde:

$T'.Nome$  = nome do atributo no esquema exportado  $E$ .

$T'.Tipo$  = tipo do atributo: [chave primária, chave estrangeira, primária-estrangeira, não chave].

$T'.Ponteiro$  = Lista de atributos que permite estabelecer as ligações entre as relações do esquema exportado, da seguinte forma: para um atributo  $T'.Nome$  de uma relação  $r$  do esquema exportado  $E$ ,

SE  $T'.Tipo$  = [primária-estrangeira] e  $T'.Nome$  = primeiro atributo que compõe a chave primária ENTÃO

$T'.Ponteiro$  = lista de todos os nomes das chaves estrangeiras da relação  $r - T'.Nome +$  nome do atributo da chave primária da relação referenciada por  $T'.Nome$ .

SE  $T'.Tipo$  = [primária-estrangeira] e  $T'.Nome \neq$  primeiro atributo que compõe a chave primária ENTÃO

$T'.Ponteiro$  = nome do atributo da chave primária da relação referenciada por  $T'.Nome$ .

SE  $T'.Tipo$  = [chave primária] e  $T'.Nome$  = primeiro atributo que compõe a chave primária ENTÃO

| <i>Gerente.Nome</i> | <i>Gerente.Tipo</i>  | <i>Gerente.Ponteiro</i> |
|---------------------|----------------------|-------------------------|
| Gerente.ID          | primária-estrangeira | Telefone.ID             |
| Gerente.nome        | não chave            | Gerente.ID              |
| Telefone.ID         | chave primária       |                         |
| Telefone.número     | não chave            | Telefone.ID             |

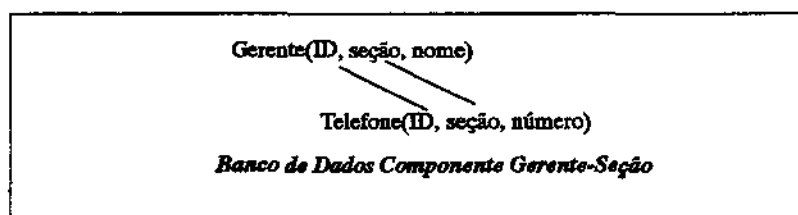
Tabela 3.10: Tabela exportada referente ao componente de banco de dados *Gerente*.

$T'.Ponteiro$  = lista de todos os nomes das chaves estrangeiras da relação  $r$ .  
 SE  $T'.Tipo$  = [chave primária] e  $T'.Nome \neq$  primeiro atributo que compõe a chave primária ENTÃO  
 $T'.Ponteiro$  = vazio.  
 SE  $T'.Tipo$  = [chave estrangeira] ENTÃO  
 $T'.Ponteiro$  = nome da chave primária da relação referenciada por  $T'.Nome$ .  
 SE  $T'.Tipo$  = [não chave] ENTÃO  
 $T'.Ponteiro$  = nome do primeiro atributo que compõe a chave primária da relação  $r$ .

Desta forma, a redundância de dados gerada pela presença de chaves primárias compostas é eliminada e todas as condições necessárias para se fazer as junções de relações com chaves estrangeiras compostas são obtidas a partir da lista de chaves estrangeiras armazenadas no campo *Ponteiro* das tabelas exportadas.

A tabela exportada 3.10 é referente ao banco de dados da figura 3.12, de acordo com as modificações propostas nesta seção.

A figura 3.13 apresenta um exemplo de um esquema de banco de dados, chamado *Gerente-Seção*, que possui chave estrangeira composta e atributos que são, ao mesmo tempo, chaves primárias e chaves estrangeiras. A tabela 3.11 é a tabela exportada para este exemplo.

Figura 3.13: Esquema exportado do banco de dados componente *Gerente-Seção*.

| <i>Gerente-Seção.Nome</i> | <i>Gerente-Seção.Tipo</i> | <i>Gerente-Seção.Ponteiro</i> |
|---------------------------|---------------------------|-------------------------------|
| Gerente.ID                | primária-estrangeira      | Gerente.seção, Telefone.ID    |
| Gerente.seção             | primária-estrangeira      | Telefone.seção                |
| Gerente.nome              | não chave                 | Gerente.ID                    |
| Telefone.ID               | chave primária            |                               |
| Telefone.seção            | chave primária            |                               |
| Telefone.número           | não chave                 | Telefone.ID                   |

Tabela 3.11: Tabela exportada referente ao componente de banco de dados *Gerente-Seção*.

A primeira linha da tabela 3.11 está armazenando a chave primária da relação "Gerente", que também é chave estrangeira desta relação. O valor do atributo *Ponteiro* nesta linha deve ser então, uma lista dos nomes dos outros atributos que também são chaves estrangeiras, neste caso, *Gerente.seção*, além do nome do atributo referenciado por *Gerente.ID*, neste caso, *Telefone.ID*. Já o atributo *Gerente.seção* possui como ponteiro o nome do atributo que compõe a chave primária da relação a qual ele se refere, ou seja, *Telefone.seção*. Os atributos "não chave" apontam para o primeiro atributo que compõe a chave primária de sua relação.

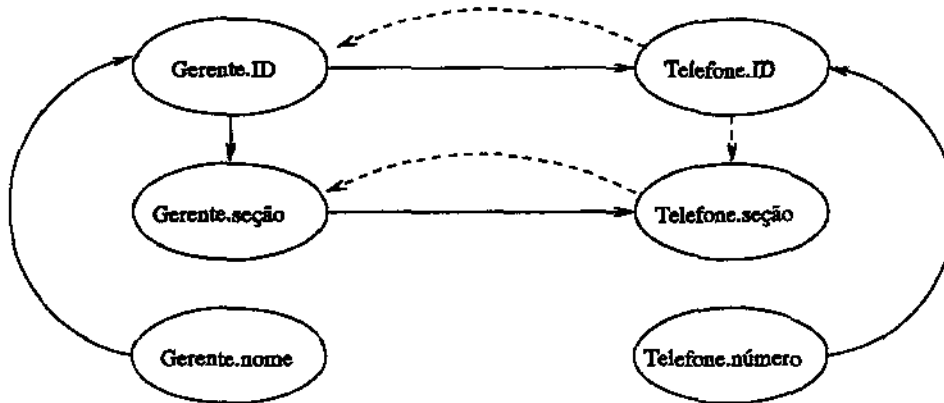


Figura 3.14: Ligações explícitas e implícitas entre as relações de cada banco de dados componente.

É importante salientar que os atributos *Telefone.ID* e *Telefone.seção* da tabela 3.11 também são chaves estrangeiras para a relação "Gerente". No entanto, o algoritmo de Zhao não considera este fato, mantendo-se assim uma relação hierárquica entre as relações de cada banco de dados componente. Desta forma, a figura 3.14, relacionada com a tabela 3.11, apresenta as ligações que existem entre as relações e que são explicitamente



| $B_3$ .Nome       | $B_3$ .Tipo       | $B_3$ .Ponteiro   | $B_3$ .Domínio |
|-------------------|-------------------|-------------------|----------------|
| Peças.nropeça     | chave primária    | Peças.localização | [0..9]         |
| Peças.pnome       | não-chave         | Peças.nropeça     | [A..Z]         |
| Peças.localização | chave-estrangeira | Depósito.dnome    | alfanumérico   |
| Depósito.dnome    | chave primária    |                   | alfanumérico   |
| Depósito.dgerente | não-chave         | Depósito.dnome    | [A..Z]         |

Tabela 3.12: Tabela exportada referente ao componente de banco de dados  $B_3$ , com informações dos domínios dos atributos.

determinadas pelas tabelas exportadas (linhas contínuas), além das ligações implícitas que não são determinadas pelas tabelas exportadas (linhas tracejadas).

O quarto problema do trabalho de Zhao é a dificuldade de se tratar os conflitos de domínio entre os atributos. Conflitos de domínio acontecem quando dois ou mais atributos com o mesmo significado semântico possuem formas diferentes de representar os seus dados. As diferenças podem ser tanto de granularidade quanto de tipo. Por exemplo, os atributos *Peça.IDpeça* e *Peças.nropeça* dos bancos de dados  $B_1$  e  $B_3$ , respectivamente, apresentados na seção 3.3.2, armazenam um código para identificar uma peça. O atributo *Peça.IDpeça* pode utilizar, por exemplo, letras (A, B, C etc.) para gerar estes códigos ao passo que o atributo *Peças.nropeça* pode utilizar números (1, 2, 3 etc.) para o mesmo fim. Através das tabelas geradas pelo algoritmo de Zhao não é possível recuperar estas informações para que sejam devidamente tratadas.

Para facilitar o tratamento desta heterogeneidade é necessário armazenar os domínios referentes a cada atributo. Os conflitos de domínio ocorrem durante o processamento de junções entre relações e durante as verificações das restrições contidas nas consultas, ações que utilizam com frequência as tabelas exportadas. Com isto, é importante que as informações sobre os domínios dos atributos também fiquem armazenadas nestas tabelas. Isto pode ser feito com a inclusão de mais um campo nas tabelas exportadas chamado *Domínio*. Considerando valores fictícios de domínio, a tabela exportada para o banco de dados componente  $B_3$  do exemplo da seção 3.3.2 passa a ser a tabela 3.12.

A partir do campo *Domínio* das tabelas exportadas é possível identificar os conflitos em tempo de processamento das consultas. Estes conflitos devem ser apresentados ao usuário através de uma interface, como a proposta em [MRS<sup>+</sup>87].

Por fim, o quinto problema refere-se às ambigüidades que podem existir decorrentes da existência de caminhos diferentes para se chegar a uma dada informação. No caso do exemplo da figura 3.8, página 45, se o atributo *Projeto.gerente* do componente  $B_1$  fosse chave estrangeira na relação "Projeto", referente ao atributo *Gerente.nome* da relação "Gerente", seria possível determinar a existência de dois caminhos entre as relações "Peça"

e "Gerente":

1. Peça.estoque ↔ Estoque.enome ↔ Estoque.gerente ↔ Gerente.nome
2. Peça.IDpeça ↔ Projeto.IDpeça ↔ Projeto.gerente ↔ Gerente.nome

Para cada caminho, o significado do atributo *Gerente.nome* é diferente. O primeiro caminho informa o nome do gerente que supervisiona o estoque onde a peça está armazenada. Já o segundo informa o nome do gerente da equipe que projetou a peça.

Estas ambigüidades devem ser tratadas diretamente pelos usuários, também através de uma interface que pode ser baseada no trabalho de Mendonça et al. [MLRN98]. Este trabalho propõe uma interface para tratar consultas incompletas em bancos de dados relacionais, ou seja, consultas que não possuem informações suficientes para se chegar a um único resultado. A partir das interpretações (caminhos) existentes, determina-se as semânticas de cada resultado possível de ser gerado e o usuário escolhe qual interpretação melhor representa suas necessidades.

### 3.4 Síntese da Metodologia Proposta

Esta seção apresenta uma síntese de todos os passos da metodologia proposta nesta dissertação, permitindo uma análise geral de todo o trabalho necessário para a integração de bancos de dados heterogêneos e sistemas legados através da criação de um banco de dados federado. Os passos estão resumidos nos itens abaixo:

#### Fase 1: Padronização

*Entradas:* Sistemas legados e esquemas de bancos de dados heterogêneos.

*Saídas:* Esquemas dos dados de entrada representados através do *modelo canônico*.

1. Identificar os bancos de dados e sistemas legados a serem integrados e as prioridades desta integração.
2. Padronizar os bancos de dados heterogêneos: Traduzir os modelos dos esquemas originais para esquemas no modelo canônico através da aplicação de algoritmos de tradução de modelos.
3. Padronizar os sistemas legados:
  - (a) Para sistemas com arquitetura não decomposta e/ou sistemas fechados:
    - i. Construir um novo sistema a partir do sistema legado.

- ii. Migrar os dados do sistema legado para o novo sistema.
  - iii. Utilizar a metodologia *Chicken Little* para proceder a migração de dados.
- (b) Para sistemas com arquitetura parcial/totalmente decomposta, sistemas de arquivos, SGBDs rudimentares e/ou sistemas com pouca documentação:
- i. 1a. opção:
    - A. Manter o sistema legado.
    - B. Utilizar um mediador para acessar os dados do sistema legado.
  - ii. 2a. opção:
    - A. Substituir o sistema legado por outro sistema mais robusto e flexível.
    - B. Migrar os dados do sistema legado para o novo sistema.
    - C. Utilizar a metodologia *Chicken Little* para proceder a migração de dados.

### Fase 2: Exportação

*Entradas:* Esquemas de dados representados no modelo canônico.

*Saídas:* Subconjuntos de dados de cada esquema de entrada, chamado *esquemas exportados*, representando os dados que realmente serão integrados.

1. Filtrar os dados a serem integrados de acordo com as necessidades dos usuários.
2. Definir os esquemas exportados.

### Fase 3: Construção da federação

*Entradas:* Esquemas exportados.

*Saída:* Sistema de bancos de dados federados.

1. Construir o esquema coordenado, usando o algoritmo de Zhao [Zha97] estendido, através da criação das seguintes estruturas:
  - (a) Matriz de correspondência de atributos;
  - (b) Tabela semântica;
  - (c) Tabelas exportadas para cada componente da federação.

## 3.5 Resumo do Capítulo

Este capítulo apresentou a metodologia proposta por esta dissertação para integração de bancos de dados heterogêneos e sistemas legados através da criação de um sistema de bancos de dados federados fracamente acoplado. Esta integração é considerada virtual e é composta por três fases principais: (1) Padronização; (2) Exportação; e (3) Criação do Sistema Integrado (federação).

A fase de Padronização é responsável por representar os dados a serem integrados através de um modelo de dados canônico (MDC). O MDC é composto pelo modelo relacional e mais informações semânticas adicionais, já que o modelo relacional não possui as mesmas facilidades de representação semânticas que outros modelos como entidade-relacionamento, OO etc.

A fase de Exportação é responsável por analisar os dados representados no MDC e filtrar aqueles que realmente farão parte da federação. Os dados filtrados formam os esquemas exportados.

A fase de Construção do Sistema Integrado é responsável por construir o sistema federado propriamente dito, baseado no trabalho de Zhao [Zha97] estendido. O sistema federado é composto por uma *matriz de correspondência de atributos*, uma *tabela semântica* e várias *tabelas exportadas*, uma para cada esquema exportado.

Além de criar o sistema federado, é necessária uma extensão da linguagem de consulta a ser utilizada para suportar o acesso aos componentes.

Embora o algoritmo de Zhao seja bem elaborado e capaz de realizar a construção do sistema federado, existem algumas limitações que foram citadas e tratadas neste capítulo através de propostas de soluções.

## Capítulo 4

# Aplicação da Metodologia Baseada em Bancos de Dados Reais

Para se estabelecer um projeto de integração de bancos de dados em uma organização, é importante a criação de um projeto piloto como ponto de partida, visando identificar e corrigir possíveis erros com minimização de custos.

Este capítulo mostra como aplicar a metodologia utilizando o exemplo do projeto SIGGER/Paulínia como estudo de caso. Este estudo foi realizado a partir de apenas três componentes heterogêneos, que precisariam ser integrados durante o desenvolvimento do projeto SIGGER. Estes componentes foram integrados através da construção de um SBDF de acordo com o apresentado na seção 3.3, levando-se em conta as modificações propostas na seção 3.3.3. Esta integração pode ser vista como um projeto piloto para o SIGGER.

Os três componentes foram especificados a partir dos bancos de dados das secretarias de Educação, de Esportes e do SAE (Serviço de Apoio ao Empregado), um serviço responsável por auxiliar pessoas desempregadas da cidade. A Secretaria de Educação possui um banco de dados gerenciado pelo *dBASE*, relativo a matrículas de alunos no sistema municipal de ensino (primeiro e segundo graus). Já a Secretaria de Esportes armazena dados das pessoas que praticam algum tipo de esporte oferecido pela prefeitura, os locais e os responsáveis pelo esporte praticado. Estes dados estão armazenados em tabelas da planilha eletrônica *EXCEL*. Por fim, o SAE mantém arquivos com extensão *DBF* gerenciados pelo programa *CLIPPER* que armazena dados pessoais e informações dos últimos empregos dos desempregados da cidade.

Várias informações armazenadas nestes três componentes são semelhantes, como por exemplo o nome, o endereço, o sexo e a naturalidade das pessoas. A partir das informações destas secretarias e do SAE a prefeitura gostaria de realizar consultas do tipo:

- Qual a naturalidade das pessoas que utilizam os serviços da prefeitura;

- Quais as pessoas que praticam esporte mas não estudam;
- Quais os desempregados que estudam.

Além disto, cada secretaria necessita de outras informações, como por exemplo:

- Quais os bairros da cidade de Paulínia que possuem menos pessoas nas escolas;
- Quais os locais de prática de esporte mais solicitados;
- Qual o tempo médio que as pessoas desempregadas permanecem nos empregos.

O SBDF criado permite executar estes e outros tipos de consultas envolvendo todos ou alguns de seus componentes.

Ressalte-se que, embora os componentes Esporte e SAE apresentarem algumas características de sistemas legados, como por exemplo, falta de documentação, falta de flexibilidade, difícil modularização, dentre outras, estes componentes não são sistemas legados complexos. O ideal para validação desta metodologia no que tange mais especificamente a parte de sistemas legados seria ter disponível sistemas mais antigos e com mais volume de dados, o que possibilitaria inclusive detectar mais problemas na fase de padronização.

A seguir é aplicada a metodologia de integração sobre os componentes escolhidos para se estabelecer a especificação do SBDF. Os componentes são referenciados pelos nomes Educação, Esporte e SAE.

## 4.1 Padronização e Exportação

O componente Educação pode ser considerado um banco de dados relacional pelo fato dos dados estarem sendo gerenciados pelo *dBASE* e por existir uma documentação que permita a representação dos dados no modelo canônico.

Já o componente Esporte se enquadra no pior caso de sistema legado, pois seus dados estão armazenados em um formato que não permite acessos a partir de mediadores, são pouco flexíveis, difíceis de gerenciar e, por isto, precisam ser migrados para um novo sistema.

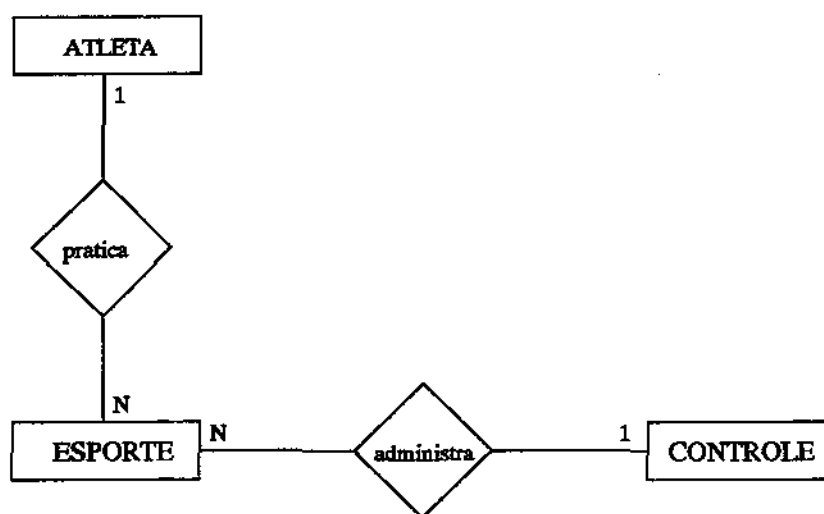
O componente SAE também é considerado um sistema legado. No entanto, os dados deste componente possuem uma organização interna maior do que o componente Esporte, já utilizam o modelo relacional e podem continuar utilizando o sistema atual feito em *CLIPPER*.

Devido a estas características, a fase de padronização deve ser diferente em cada caso. O componente Educação possui em sua documentação a descrição das tabelas relacionais dos dados armazenados e, portanto, o modelo de representação dos dados usado é o

próprio modelo canônico. Desta forma, a fase de padronização é desnecessária para este componente. O mesmo se aplica ao componente SAE.

Já o componente Esporte necessita inicialmente de um modelo de representação para seus dados, já que este componente não possui documentação. Assim, foi construído um diagrama entidade-relacionamento (ER) analisando os próprios dados armazenados nas tabelas *EXCEL*. A partir deste diagrama, é possível determinar o modelo canônico através dos algoritmos de tradução de modelos apresentados na seção 3.2.1. A figura 4.1 apresenta o diagrama ER para este componente. Os atributos de cada entidade foram ocultados nesta figura para simplificar o entendimento, mas serão apresentados mais adiante.

No caso destes componentes, não foi necessário criar tabelas adicionais para armazenar restrições ou informações semânticas dos dados. Desta forma, o modelo canônico, neste caso, é composto apenas do modelo relacional. Além disto, a heterogeneidade de domínio não foi considerada neste estudo de caso e, portanto, o campo *Domínio* foi ocultado das tabelas exportadas de cada componente.



*Diagrama Entidade-Relacionamento*

Figura 4.1: Diagrama Entidade-Relacionamento do componente legado Esporte.

Aplicando o algoritmo de tradução de modelos proposto em [Oli93], são criadas relações para cada entidade do diagrama ER (Atleta, Controle e Esporte). As chaves primárias destas relações são os atributos chaves de suas respectivas entidades. Como os relacionamentos presentes na figura 4.1 são do tipo 1:N, não são necessárias outras relações.

O diagrama ER e as relações criadas também auxiliam o processo de migração de dados, já que o componente Esporte necessita de um outro sistema para gerenciar seus dados. Esta migração deve ser realizada seguindo a metodologia *Chicken Little* citada

anteriormente, para que esta operação seja feita de forma mais controlada. O SGBD candidato a gerenciar estes dados seria o *ORACLE*, por estar sendo cogitado como o SGBD a ser usado futuramente. Como ainda não foi definido nem o SIG nem o SGBD a ser adquirido pela prefeitura de Paulínia, a escolha do SGBD para estes dados fica adiada. No entanto, isto não compromete a criação da federação, desde que escolhido um SGBD relacional.

As relações dos componente Esporte, Educação e SAE estão representadas na figura 4.2

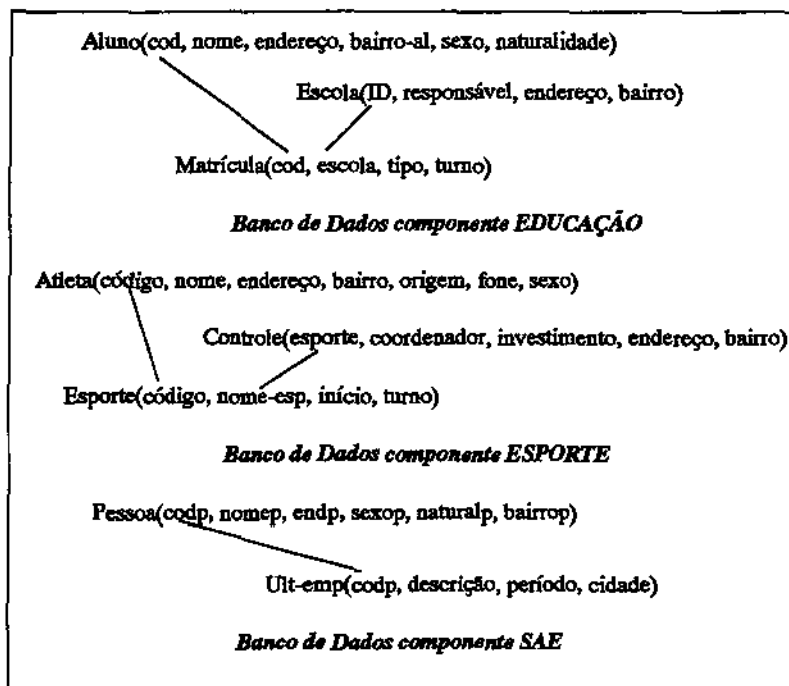


Figura 4.2: Esquemas dos componentes *Educação*, *Esporte* e *SAE*.

No componente Educação, a relação "Aluno" armazena os dados pessoais dos alunos que estão ou já foram matriculados em algum curso oferecido pela prefeitura. A matrícula e os dados dos cursos estão armazenados na relação "Matricula". O atributo *tipo* em "Matricula" indica o curso em que a matrícula foi realizada (por exemplo, "série 1 grau 1" ou "informática"). Já o atributo identificador da escola, o responsável e o endereço de cada escola estão armazenados na relação "Escola".

O componente Esporte armazena na relação "Atleta" os dados das pessoas que praticam ou já praticaram algum esporte oferecido pela secretaria de esportes. As matrículas realizadas pelos atletas são armazenadas na relação "Esporte". O atributo *início* em "Esporte" armazena a data em que o atleta começou a praticar um determinado esporte.



Os nomes dos esportes disponíveis, os responsáveis, o valor investido e os locais onde se pratica cada esporte ficam armazenados na relação "Controle".

Já o componente SAE armazena os dados das pessoas desempregadas na relação "Pessoa". Além disto, o SAE armazena na relação "Ult-emp" as informações relativas aos últimos empregos em que a pessoa trabalhou. O atributo *descrição* em "Ult-emp" armazena o cargo ou função que a pessoa já exerceu e o atributo *período* informa as datas iniciais e finais que a pessoa trabalhou em cada emprego.

Terminada a fase de padronização dos componentes, passa-se para a fase de exportação. No entanto, neste caso, todos os dados dos componentes serão federados, o que dispensa a fase de exportação. Desta forma, os esquemas exportados e os esquemas resultantes da fase de padronização são os mesmos.

## 4.2 Construção da Federação

A partir dos dados padronizados e representados através do modelo canônico, é possível proceder à construção da federação composta pelos componentes Educação, Esporte e SAE, como mostra a figura 4.3.

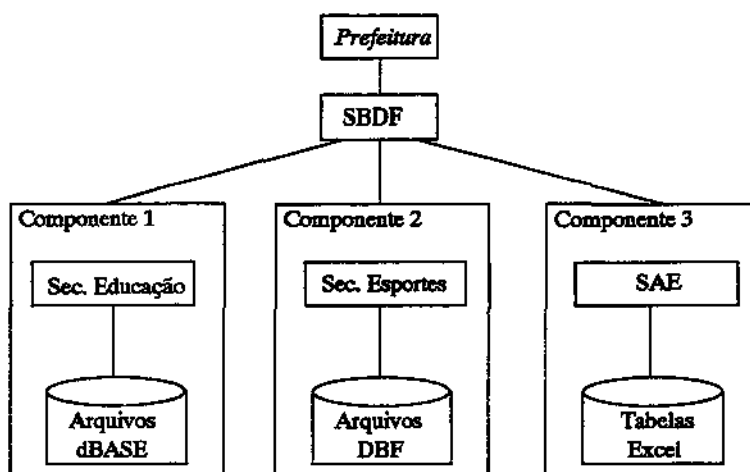


Figura 4.3: Sistema Federado com componentes *Educação*, *Esporte* e *SAE*.

Aplicando o algoritmo de Zhao estendido, são criadas as tabelas exportadas de cada componente, sendo a tabela 4.1 referente ao componente Educação, a tabela 4.2 referente ao componente Esporte e a tabela 4.3 referente ao componente SAE.

Por fim, são criadas a tabela semântica 4.4 e a matriz de correspondência de atributos 4.5 relativas a estes componentes.

| <i>Educação.Nome</i> | <i>Educação.Tipo</i> | <i>Educação.Ponteiro</i> |
|----------------------|----------------------|--------------------------|
| Aluno.cod            | primária-estrangeira | Matrícula.cod            |
| Aluno.nome           | não chave            | Aluno.cod                |
| Aluno.endereço       | não chave            | Aluno.cod                |
| Aluno.bairro-al      | não chave            | Aluno.cod                |
| Aluno.sexo           | não chave            | Aluno.cod                |
| Aluno.naturalidade   | não chave            | Aluno.cod                |
| Escola.ID            | primária-estrangeira | Matrícula.escola         |
| Escola.responsável   | não chave            | Escola.ID                |
| Escola.endereço      | não chave            | Escola.ID                |
| Escola.bairro        | não chave            | Escola.ID                |
| Matrícula.cod        | primária             |                          |
| Matrícula.escola     | primária             |                          |
| Matrícula.tipo       | primária             |                          |
| Matrícula.turno      | não chave            | Matrícula.cod            |

Tabela 4.1: Tabela exportada referente ao componente de banco de dados Educação.

| <i>Esporte.Nome</i>   | <i>Esporte.Tipo</i>  | <i>Esporte.Ponteiro</i> |
|-----------------------|----------------------|-------------------------|
| Atleta.código         | primária-estrangeira | Esporte.código          |
| Atleta.nome           | não chave            | Atleta.código           |
| Atleta.endereço       | não chave            | Atleta.código           |
| Atleta.bairro         | não chave            | Atleta.código           |
| Atleta.origem         | não chave            | Atleta.código           |
| Atleta.fone           | não chave            | Atleta.código           |
| Atleta.sexo           | não chave            | Atleta.código           |
| Controle.esporte      | primária-estrangeira | Esporte.nome-esp        |
| Controle.coordenador  | não chave            | Controle.esporte        |
| Controle.investimento | não chave            | Controle.esporte        |
| Controle.endereço     | não chave            | Controle.esporte        |
| Controle.bairro       | não chave            | Controle.esporte        |
| Esporte.código        | primária             |                         |
| Esporte.nome-esp      | primária             |                         |
| Esporte.início        | primária             |                         |
| Esporte.turno         | não chave            | Esporte.código          |

Tabela 4.2: Tabela exportada referente ao componente de banco de dados Esporte.

| <i>SAE.Nome</i>   | <i>SAE.Tipo</i>      | <i>SAE.Ponteiro</i> |
|-------------------|----------------------|---------------------|
| Pessoa.codp       | primária-estrangeira | Ult-emp.codp        |
| Pessoa.nomep      | não chave            | Pessoa.codp         |
| Pessoa.endp       | não chave            | Pessoa.codp         |
| Pessoa.sexop      | não chave            | Pessoa.codp         |
| Pessoa.naturalp   | não chave            | Pessoa.codp         |
| Pessoa.bairrop    | não chave            | Pessoa.codp         |
| Ult-emp.codp      | primária             |                     |
| Ult-emp.descrição | primária             |                     |
| Ult-emp.período   | primária             |                     |
| Ult-emp.cidade    | não chave            | Ult-emp.codp        |

Tabela 4.3: Tabela exportada referente ao componente de banco de dados SAE.

| <i>Semântica</i>                           | <i>Atributo-Federado</i> |
|--------------------------------------------|--------------------------|
| Identificação da pessoa                    | IDPessoa                 |
| Nome da pessoa                             | NomePessoa               |
| Endereço da pessoa                         | EndereçoPessoa           |
| Bairro da pessoa                           | BairroPessoa             |
| Naturalidade da pessoa                     | NaturalPessoa            |
| Telefone da pessoa                         | FonePessoa               |
| Sexo da pessoa                             | SexoPessoa               |
| Identificação do participante da atividade | PessoaAtividade          |
| Nome da atividade                          | NomeAtividade            |
| Turno da atividade                         | TurnoAtividade           |
| Data de início da atividade                | InícioAtividade          |
| Classificação da atividade                 | TipoAtividade            |
| Identificação da atividade                 | IDAtividade              |
| Recursos destinados à atividade            | GastosAtividade          |
| Responsável pela atividade                 | ResponsávelAtividade     |
| Endereço onde se realiza a atividade       | EndereçoAtividade        |
| Bairro onde se realiza a atividade         | BairroAtividade          |
| Identificação da pessoa desempregada       | PessoaEmprego            |
| Descrição do último emprego da pessoa      | DescriçãoEmprego         |
| Intervalo de tempo de trabalho no emprego  | PeríodoEmprego           |
| Cidade do último emprego da pessoa         | CidadeEmprego            |

Tabela 4.4: Tabela semântica.

| <i>Atributo-Federado</i> | <i>Educação</i>    | <i>Esporte</i>        | <i>SAE</i>        |
|--------------------------|--------------------|-----------------------|-------------------|
| IDPessoa                 | Aluno.cod          | Atleta.código         | Pessoa.codp       |
| NomePessoa               | Aluno.nome         | Atleta.nome           | Pessoa.nomep      |
| EndereçoPessoa           | Aluno.endereço     | Atleta.endereço       | Pessoa.endp       |
| BairroPessoa             | Aluno.bairro-al    | Atleta.bairro         | Pessoa.bairrop    |
| NaturalPessoa            | Aluno.naturalidade | Atleta.origem         | Pessoa.naturalp   |
| FonePessoa               |                    | Atleta.fone           |                   |
| SexoPessoa               | Aluno.sexo         | Atleta.sexo           | Pessoa.sexop      |
| PessoaAtividade          | Matrícula.cod      | Esporte.código        |                   |
| NomeAtividade            | Matrícula.escola   | Esporte.nome-esp      |                   |
| TurnoAtividade           | Matrícula.turno    | Esporte.turno         |                   |
| InícioAtividade          |                    | Esporte.início        |                   |
| TipoAtividade            | Matrícula.tipo     |                       |                   |
| IDAtividade              | Escola.ID          | Controle.esporte      |                   |
| GastosAtividade          |                    | Controle.investimento |                   |
| ResponsávelAtividade     | Escola.responsável | Controle.coordenador  |                   |
| EndereçoAtividade        | Escola.endereço    | Controle.endereço     |                   |
| BairroAtividade          | Escola.bairro      | Controle.bairro       |                   |
| PessoaEmprego            |                    |                       | Ult-emp.codp      |
| DescriçãoEmprego         |                    |                       | Ult-emp.descrição |
| PeríodoEmprego           |                    |                       | Ult-emp.período   |
| CidadeEmprego            |                    |                       | Ult-emp.cidade    |

Tabela 4.5: Matriz de correspondência de atributos.

Através da federação criada, várias consultas podem ser processadas, tanto em nível local quanto em nível integrado. A seguir são apresentadas algumas consultas que os usuários desta federação gostariam de processar, obtidas através de entrevistas com estes usuários:

- Qual a naturalidade das pessoas que estão desempregadas?
- Qual o bairro que possui mais pessoas estudando e praticando esportes?
- Quais os responsáveis pelos cursos e os esportes praticados por "José"?
- Quais as pessoas que praticam esporte e que estão desempregadas?
- Qual o nome e a naturalidade das pessoas que não são naturais de Paulínia, estão matriculadas em algum curso, praticam esporte e estão desempregadas?

O primeiro exemplo de consulta utiliza dados de apenas um componente . Já as três consultas seguintes utilizam dados de dois componentes da federação. Por fim, a última consulta utiliza dados de todos os componentes federados e será usada para exemplificar o processamento de consulta nesta federação.

Utilizando a linguagem SQL estendida de Zhao, esta consulta pode ser expressa pela seguinte expressão:

```
SELECT Educação.[NomePessoa], Educação.[NaturalPessoa]
FROM Educação, Esporte, SAE
WHERE Educação.[NomePessoa] = (Esporte, SAE).[NomePessoa]
AND Educação.[NaturalPessoa] ≠ "Paulínia"
```

O primeiro passo para responder a esta consulta é dividi-la em subconsultas, de acordo com os relacionamentos entre os componentes, produzindo as seguintes subconsultas:

- Junção usando os componentes Educação e Esporte
 

```
SELECT Educação.[NomePessoa], Educação.[NaturalPessoa]
FROM
WHERE Educação.[NomePessoa] = Esporte.[NomePessoa]
AND Educação.[NaturalPessoa] ≠ "Paulínia"
```
- Junção usando os componentes Educação e SAE
 

```
SELECT Educação.[NomePessoa], Educação.[NaturalPessoa]
FROM
```

|                                     |        |                                                                     |
|-------------------------------------|--------|---------------------------------------------------------------------|
| Subconsulta criada                  | SELECT | Educação.[NomePessoa], Educação.[NaturalPessoa]                     |
|                                     | FROM   |                                                                     |
|                                     | WHERE  | Educação.[NomePessoa] = Esporte.[NomePessoa]                        |
|                                     | AND    | Educação.[NaturalPessoa] ≠ "Paulínia"                               |
| Conversão de atributos              | SELECT | Educação.Aluno.nome, Educação.Aluno.naturalidade                    |
|                                     | FROM   |                                                                     |
|                                     | WHERE  | Educação.Aluno.nome = Esporte.Atleta.nome                           |
|                                     | AND    | Educação.Aluno.naturalidade ≠ "Paulínia"                            |
| Especificação de junções            | SELECT | Educação.Aluno.nome, Educação.Aluno.naturalidade                    |
|                                     | FROM   |                                                                     |
|                                     | WHERE  | Educação.Aluno.nome = Esporte.Atleta.nome                           |
|                                     | AND    | Educação.Aluno.naturalidade ≠ "Paulínia"                            |
|                                     | AND    | Educação.Aluno.cod = Educação.Matricula.cod                         |
|                                     | AND    | Esporte.Atleta.código = Esporte.Esporte.código                      |
| Identificação dos objetos originais | SELECT | Educação.Aluno.nome, Educação.Aluno.naturalidade                    |
|                                     | FROM   | Educação.Aluno, Educação.Matricula, Esporte.Atleta, Esporte.Esporte |
|                                     | WHERE  | Educação.Aluno.nome = Esporte.Atleta.nome                           |
|                                     | AND    | Educação.Aluno.naturalidade ≠ "Paulínia"                            |
|                                     | AND    | Educação.Aluno.cod = Educação.Matricula.cod                         |
|                                     | AND    | Esporte.Atleta.código = Esporte.Esporte.código                      |

Figura 4.4: Processo de tradução da subconsulta envolvendo os componentes Educação e Esporte.

```
WHERE Educação.[NomePessoa] = SAE.[NomePessoa]
AND Educação.[NaturalPessoa] ≠ "Paulínia"
```

As figuras 4.4 e 4.5 apresentam o processamento destas subconsultas, seguindo o algoritmo de Zhao estendido.

Cada subconsulta retorna uma tabela contendo as informações de cada componente requeridas pelo usuário. Sobre as tabelas resultantes é aplicada uma operação de união, pois os dados de cada subconsulta são complementares e têm o mesmo esquema.

A partir desta federação criada, é possível inserir outros componentes de bancos de dados, criando outras tabelas exportadas e adicionando os novos atributos à matriz de correspondência de atributos e, se necessário, à tabela semântica. Exemplos de entidades que podem constituir um componente nesta federação são: hospitais, creches, postos de saúde, secretarias de segurança, transporte dentre outras.

Além das secretarias e da prefeitura que pode utilizar o SBDF criado, este sistema pode fornecer dados ao SIG a ser implantado. Um SIG pode ser inserido neste contexto, basicamente, de três formas. Os conceitos básicos de SIG bem como a sua utilização no SBDF serão vistos no próximo capítulo.

Um protótipo do sistema que permite processar consultas sobre a federação está sendo implementado. Este protótipo está sendo desenvolvido utilizando o *Microsoft Visual Fox-Pro 5.0* como SGBD e o aplicativo *Borland Delphi 3.0* para desenvolver as suas funcionalidades. Estas escolhas foram feitas, basicamente, por serem ferramentas de fácil

|                                     |                                |                                                                                                                                                                      |
|-------------------------------------|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Subconsulta criada                  | SELECT<br>FROM<br>WHERE<br>AND | Educação.[NomePessoa], Educação.[NaturalPessoa]<br><br>Educação.[NomePessoa] = SAE.[NomePessoa]<br>Educação.[NaturalPessoa] ≠ "Paulínia"                             |
| Conversão de atributos              | SELECT<br>FROM<br>WHERE<br>AND | Educação.Aluno.nome, Educação.Aluno.naturalidade<br><br>Educação.Aluno.nome = SAE.Pessoa.nomep<br>Educação.Aluno.naturalidade ≠ "Paulínia"                           |
| Especificação de junções            | SELECT<br>FROM<br>WHERE<br>AND | Educação.Aluno.nome, Educação.Aluno.naturalidade<br><br>Educação.Aluno.nome = SAE.Pessoa.nomep<br>Educação.Aluno.naturalidade ≠ "Paulínia"                           |
| Identificação dos objetos originais | SELECT<br>FROM<br>WHERE<br>AND | Educação.Aluno.nome, Educação.Aluno.naturalidade<br>Educação.Aluno, SAE.Pessoa<br>Educação.Aluno.nome = SAE.Pessoa.nomep<br>Educação.Aluno.naturalidade ≠ "Paulínia" |

Figura 4.5: Processo de tradução da subconsulta envolvendo os componentes Educação e SAE.

entendimento e, portanto, passíveis de utilização imediata. Além disto, o *Borland Delphi* apresenta um conjunto de campos de busca, filtros de campos e outras funcionalidades úteis e de fácil implementação. A herança visual de formulários, que é um recurso de programação orientada a objetos, auxilia a reutilização de código e torna o seu desenvolvimento mais rápido.

Neste protótipo, o usuário terá à disposição listas com os nomes dos atributos federados, os operadores disponíveis para formação de predicados (=, ≠, <, > e outros), os nomes dos bancos de dados componentes e uma interface que lhe permitirá representar a consulta na linguagem SQL. A partir da consulta fornecida pelo usuário, o protótipo irá decompor a consulta em subconsultas e enviá-las aos componentes responsáveis pelas respostas. Os resultados das subconsultas serão processados e apresentados aos usuários sub a forma de tabelas. A figura 4.6 apresenta uma tela do protótipo em que o usuário formula suas consultas e envia para o banco de dados federado.

### 4.3 Resumo do Capítulo

Este capítulo apresentou uma aplicação da metodologia proposta nesta dissertação para a criação de uma federação, baseada em um caso real da cidade de Paulínia-SP. O SBDF criado possui três componentes sendo, duas secretarias (Educação e Esportes) e uma repartição da prefeitura que presta serviços às pessoas desempregadas da cidade (SAE). A organização dos dados armazenados em cada componente foi obtida através de entrevistas realizadas com usuários e pessoas que necessitam utilizar as informações já existentes de uma forma integrada.

| Nome do Campo    | Descrição                         |
|------------------|-----------------------------------|
| Endereço         | Endereço eletrônico do gerente    |
| EquipeProjetista | Nome da equipe projetista da peça |
| EstoquePeça      | Local de armazenamento da peça    |
| FoneGerente      | Telefone do gerente               |

Below the table, there is a grid of input fields for query formulation. The fields are arranged in a grid with labels on the left: 'SELECÇÃO', 'TABELA', 'OPERADOR', 'E', 'E', 'E', 'E', 'E'. Each label is followed by a series of input boxes for constructing the query.

Figura 4.6: Tela para formulação de consultas no sistema federado.

Como o projeto SIGGER não chegou ao nível de desenvolvimento esperado durante o desenvolvimento desta dissertação, não foi possível descrever na íntegra os nomes reais dos atributos que armazenam as informações de cada componente. No entanto, a partir dos dados coletados nas entrevistas, foi possível identificar os atributos que necessariamente cada componente deveria conter.

A federação criada permite processar consultas que levam a uma melhor administração dos recursos destinados a cada setor da prefeitura e auxiliam o gerenciamento das escolas e quadras de esporte. Além disto, é possível identificar, por exemplo, a naturalidade das pessoas que estão fazendo uso dos serviços públicos.

Outros componentes podem ser adicionados na federação sem comprometer o SBDF criado. Para isto é necessário criar as tabelas exportadas para os novos componentes e alterar a matriz de correspondência de atributos e a tabela semântica de acordo com as necessidades.

O protótipo em desenvolvimento possibilitará testes mais ostensivos desta metodologia e poderá ser usado pela prefeitura de Paulínia como um projeto piloto para o SIGGER, caso haja interesse das partes envolvidas.



## Capítulo 5

# Metodologia Aplicada a Sistemas de Informações Geográficas

A introdução de *Sistema de Informações Geográficas* (SIG) na administração (municipal, estadual, regional etc.) está se expandindo exponencialmente. No entanto, esta nova tecnologia vem sendo adotada de forma desordenada e pouco integrada. Além do mais, os responsáveis pelo gerenciamento de dados da administração raramente se preocupam com o problema de integrar dados/sistemas legados aos novos sistemas geográficos. Desta forma, a metodologia proposta nesta dissertação pode ser usada em ambientes que utilizam *bancos de dados geográficos* visando sua utilização por SIGs. Bancos de dados geográficos são aqueles que, além de armazenarem dados convencionais (nome, endereço, telefone etc.), armazenam tabelas que relacionam os dados convencionais a dados espaciais. Isto permite que os dados sejam georeferenciados e possam ser utilizados pelos SIGs.

A metodologia de integração proposta pode ser aplicada na integração de dados que sejam manipulados por SIGs. No entanto, a metodologia definida nesta dissertação é capaz de tratar apenas a parte convencional dos dados, já que os dados espaciais necessitam de cuidados especiais. As seções subseqüentes apresentam os conceitos básicos de SIGs, apontam os problemas principais da integração de dados espaciais e mostram como SIGs podem ser inseridos no contexto desta dissertação através de pequenas modificações na metodologia proposta.

### 5.1 Sistemas de Informações Geográficas

Falta ainda, dentro da comunidade científica, uma definição padrão para SIGs. Basicamente, um SIG pode ser visto como um *software* composto de vários subsistemas integrados voltados para geração de mapas e extração de informações sobre os objetos geográficos representados nestes mapas [Cif95]. SIGs são sistemas dependentes de bancos de dados

que manipulam dados sobre fenômenos geográficos, associados à sua localização física e relacionamentos espaciais [Cer96]. Para Câmara et al. [CCH+96], SIGs são sistemas automatizados usados para armazenar, analisar e manipular dados geográficos, ou seja, dados que representam objetos e fenômenos em que a localização geográfica é uma característica inerente à informação e indispensável para analisá-la. Para isto, é necessário que o SIG possua como componente um sistema gerenciador de banco de dados (SGBD) não convencional. Este SGBD é responsável por permitir o uso conjunto de uma enorme quantidade de dados espaciais e convencionais, através de estruturas de armazenamento de dados, linguagens e otimizadores de consultas específicos.

Entende-se por objeto geográfico uma representação de uma entidade do mundo real, possuindo uma localização em relação à Terra e uma geometria, ou seja, esta entidade deve ser *georeferenciada* [Cif95].

O *mapa* é um conceito chave no contexto de SIG. Um *mapa* é uma representação em superfície plana e em escala menor de objetos geográficos como um país, um estado, uma cidade, um terreno ou uma residência, dentre outros [Agu95].

Usuários de SIG enxergam o mundo de acordo com duas visões: *visão de objetos* e *visão de campos*. Na visão de campos, cada localização está relacionada a um valor ou conjunto de valores descritivos do fenômeno geográfico, formando um conjunto de regiões disjuntas cuja união determina o campo. Este modelo normalmente é empregado para representar fenômenos cuja natureza geográfica não é precisa como, por exemplo, tipos de solos, temperatura, altitude e vegetação. Por outro lado, na visão de objetos, o mundo é encarado como um conjunto de entidades georeferenciadas bem definidas, cuja existência independe do fenômeno geográfico em estudo.

### 5.1.1 Modelo de Dados para Aplicações Geográficas

Um fator importante do ponto de vista de SIG é o modelo de dados utilizado. Modelos de dados são abstrações conceituais que permitem a representação de entidades do mundo real [Cer96]. Um modelo de dados deve fornecer ferramentas para descrever a organização lógica de bancos de dados, bem como definir as operações de manipulação de dados permitidas [CCH+96].

Estudos mais recentes recomendam o uso de orientação para a modelagem de dados em SIG porque, além de flexíveis, estes modelos facilitam a especificação incremental de aplicações, característica importante em SIG [CCH+96]. O modelo de dados SAIF (Spatial Archive and Interchange Format) [Ogi97] é o modelo orientado a objetos considerado padrão pelos desenvolvedores de produtos comerciais devido à facilidade de mapeamento entre o SAIF e outros modelos. O SAIF incorpora, dentre outros, conceitos de identidade, generalização, agregação, herança e associações simples entre objetos, além de oferecer

algumas classes construtoras geográficas básicas, a partir das quais outras classes podem ser criadas.

O trabalho de Shekhar et al. [SCG<sup>+</sup>97] propõe um novo modelo chamado *Geographic Information System Entity Relational Model* que é uma extensão do modelo Entidade-Relacionamento em que é possível o uso de campos contínuos, integrando visão de campos e de objetos. Desta forma, vários aspectos de gerenciamento de dados de entrada, modelagem, consultas e apresentação de resultados podem ser feitos por um único modelo integrado.

Pires [Pir97] também propõe um modelo para sistemas de informações geográficas baseado em orientação a objetos chamado GMOD que contempla não apenas a modelagem de dados, mas também a modelagem dinâmica e de processos. Este modelo permite a especificação de regras para a implementação de restrições e funções de monitoramento e simulação, possibilitando retratar a dinâmica do mundo real. Além disto, o GMOD é independente de qualquer modelo de dados utilizado por SIGs comerciais, permitindo representar dados convencionais e georeferenciados em um nível mais alto de abstração.

### 5.1.2 Formato dos Dados

O formato dos dados a serem armazenados em um SIG pode assumir dois tipos: *varredura* e *vetorial* [Agu95]. A opção por um dos formatos depende do tipo de consulta que se queira processar.

O formato varredura é conhecido como formato baseado em posição, já que ele considera como principal componente para a representação de um objeto sua própria posição. Neste formato, cada posição é caracterizada por possuir um conjunto de propriedades do objeto que está representando. Uma estrutura matricial é utilizada para o armazenamento dos dados e cada posição da matriz corresponde a uma célula [Agu95, Cif95]. O principal problema desta representação está na falta de precisão.

Já o formato vetorial considera que o principal componente para a representação de um objeto é sua característica geográfica/cartográfica e, por isto, este formato é conhecido como baseado em característica. As demais propriedades dos objetos, incluindo posição, são armazenadas como propriedades adicionais.

A escolha do modelo de dados bem como do formato dos dados de um SIG depende do tipo de aplicação desejada e se o SIG será usado para fins urbanos e rurais ou para fins ambientais.

### 5.1.3 Caracterização das Aplicações

Aplicações voltadas para fins urbanos e rurais envolvem informações sobre limites de ruas, avenidas, praças, lotes, propriedades particulares, rios e outros dados que podem ser úteis

para cálculos de taxas e impostos, controle de natalidade, serviços de entregas (correios, revistas, jornais), controle de nível de desemprego, analfabetismo e vários outros fins que necessitam de informações agrupadas por regiões ou setores e que podem ser facilitados com a utilização de mapas. Este tipo de aplicação utiliza em geral dados em formato vetorial e visão de objetos.

Aplicações voltadas para administração e planejamento ambiental cuidam das informações sobre recursos naturais, como por exemplo florestas e parques ecológicos, com o objetivo de controlar a utilização destes recursos e planejar a renovação dos mesmos. Este tipo de domínio costuma usar dados em formato varredura, inclusive pelo tipo de informação utilizada (imagens de satélite ou radar).

## 5.2 SIGs e a Metodologia Proposta

Nesta dissertação, o conceito de SIG considerado é o mesmo adotado em [CCH<sup>+</sup>96], voltado para aplicações urbanas.

Quando se deseja considerar SIGs no processo de integração de dados, deve-se considerar dois tipos de gerenciamento: (1) gerenciamento de dados convencionais e (2) gerenciamento de dados espaciais. A ligação entre estes tipos de dados é feita através de um mecanismo semelhante ao de integridade referencial em bancos de dados relacionais. Quando se considera este tipo de paradigma, além de realizar a integração dos dados convencionais como foi apresentado no capítulo 3, é necessário integrar os dados espaciais. Este último é um problema em aberto e detalhes escapam ao escopo desta dissertação. No entanto, aqui são discutidos alguns problemas que causam heterogeneidade de dados espaciais, como por exemplo:

1. Representação (varredura/vetorial);
2. Escala;
3. Projeção;
4. Qualidade e heterogeneidade temporal e espacial dos dados.

O primeiro problema acontece quando alguns dados espaciais utilizam a representação varredura e outros utilizam a representação vetorial. O segundo problema está relacionado com dados armazenados em diferentes escalas, o que impede o processamento de algumas operações como por exemplo a sobreposição de imagens ou operações topológicas. Já o terceiro problema ocorre quando a projeção dos dados espaciais é feita através de diferentes técnicas (por exemplo, projeção cilíndrica e cônica), inviabilizando o uso integrado destes dados. Por fim, o quarto problema está relacionado com a fase de coleta de dados. Alguns

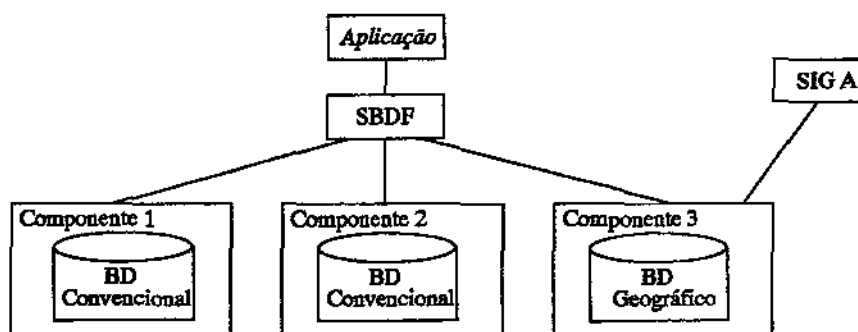


Figura 5.1: Primeiro caso de integração de dados envolvendo SIG.

dados podem ter sido coletados há muito tempo, utilizando equipamentos que permitiam um nível de qualidade e de detalhes diferentes dos dados recém coletados. Além disto, os dados espaciais de uma região podem se alterar com o tempo, seja pelo desenvolvimento da região, desmatamentos ou pelos próprios fatores ambientais.

Para resolver os três primeiros problemas, é necessário uma padronização dos dados espaciais, ou seja, é preciso escolher uma representação, uma escala e uma projeção padrão e, posteriormente, converter todos os dados para este padrão. Vale ressaltar que a padronização neste caso só é possível se os dados estiverem sendo usados por SIGs iguais, pois cada SIG depende totalmente de formatos quase que proprietários dos dados espaciais. Quando os SIGs são diferentes é preciso migrar os dados de cada SIG para um banco de dados independente, o que deixa de ser uma integração de dados.

O quarto problema, na maioria dos casos, não permite a realização de algum tipo de padronização e dificilmente os dados podem ser usados de forma integrada.

Existem, basicamente, três casos em que SIGs são inseridos no contexto desta dissertação. Nestes casos, a metodologia precisa considerar os problemas aqui citados e que envolvem integração de dados espaciais. Os casos são:

- SIG utilizando um dos bancos de dados componente da federação;
- SIG acessando diretamente a federação;
- SIG acessando alguns dos componentes da federação.

No primeiro caso, um dos componentes da federação é um banco de dados geográfico. No entanto, apenas os dados convencionais deste componente são utilizados pela federação, enquanto os dados espaciais são processados apenas localmente. A obtenção dos dados convencionais para uso na federação é feita, extraíndo estes dados do banco de dados geográfico através de mediadores e disponibilizando-os para a federação. Desta

forma, a metodologia proposta não precisa ser alterada, já que apenas os dados convencionais são exportados. Como a federação mantém a autonomia dos seus componentes, o SIG pode continuar acessando o banco de dados geográfico normalmente. A figura 5.1 ilustra esta situação, em que o **SIG A** utiliza o componente 3 da federação sem interferir na integração dos dados convencionais.

No segundo caso, exemplificado pela figura 5.2, a federação integra tanto dados convencionais quanto espaciais. Cada componente, neste caso, é um banco de dados geográfico que está sendo acessado por algum SIG. Para que isto aconteça, a metodologia proposta no capítulo anterior deve ser acrescida, em cada fase, dos seguintes passos:

### **Fase 1: Padronização**

#### 1. Padronizar os bancos de dados geográficos:

- Determinar representação básica;
- Determinar escala padrão;
- Verificar problemas de diferenças de projeção;
- Executar procedimentos básicos de transformação dos mapas para unificar a representação.

### **Fase 2: Exportação**

1. Exportar o esquema referente à parte dos dados convencionais;
2. Exportar as definições dos arquivos espaciais.

### **Fase 3: Construção da federação**

- As tabelas convencionais devem ser construídas como anteriormente. A única diferença é que, em vários SIGs, existem atributos com nomes específicos que denotam ponteiros para mapas. Estes nomes de atributos são particulares para cada SIG. Em especial, dificilmente um SIG consegue acessar arquivos definidos por um SIG diferente, sem que haja um pré-processamento.

O terceiro caso considera uma união dos dois primeiros casos através da construção de duas federações, uma sobre os dados convencionais e outra sobre os dados espaciais de cada componente. A figura 5.3 mostra uma instanciamento deste caso. A federação *Y* integra o componente 3 com os dados convencionais dos componentes 1 e 2. Já a federação *X* integra os dados convencionais e espaciais dos componentes 1 e 2. A partir

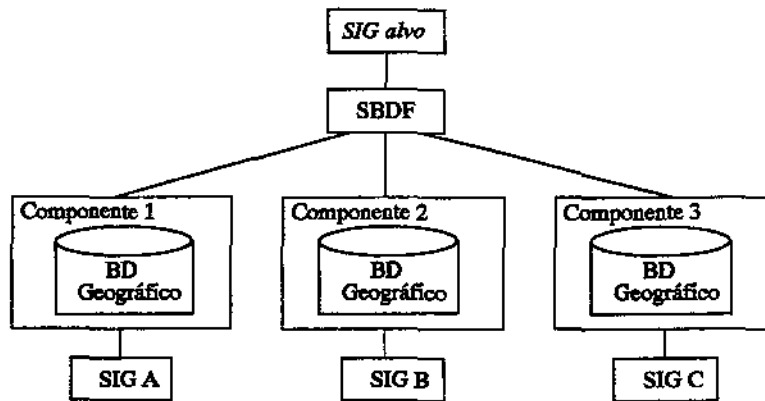


Figura 5.2: Segundo caso de integração de dados envolvendo SIG.

daí, SIGs e aplicações podem utilizar o sistema federado para acessar dados integrados. Para a federação *Y* também não são necessárias modificações na metodologia. No entanto, para a federação *X*, considera-se as mesmas modificações propostas anteriormente para o segundo caso.

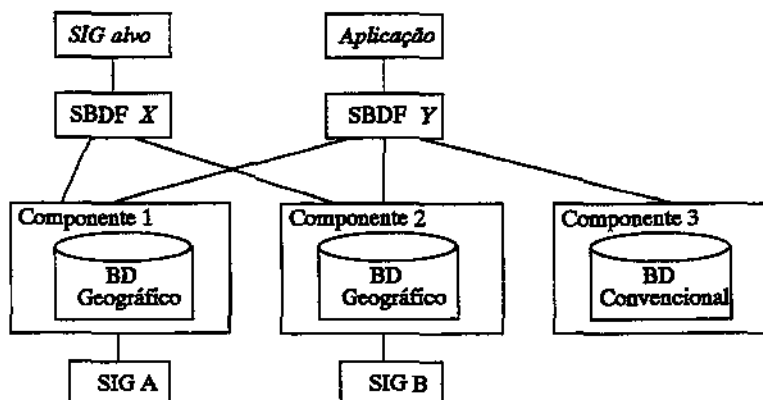


Figura 5.3: Terceiro caso de integração de dados envolvendo SIG.

Voltando ao exemplo apresentado no capítulo 4 sobre o SIGGER/Paulínia, a federação especificada para integrar os componentes Educação, Esporte e SAE pode ser usada por um SIG para auxiliar a administração urbana. Neste sentido, o SIG pode acessar a federação de duas formas através dos segundo e terceiro casos descritos anteriormente, ou seja, acessando diretamente a federação ou acessando alguns dos componentes federados.

Em ambos os casos, é necessário que exista pelo menos um atributo georeferenciado em cada componente utilizado pelo SIG para que se estabeleça a ligação entre os dados textuais e os mapas da cidade de Paulínia. No caso dos componentes em questão, os

atributos que armazenam endereços devem passar a ser georeferenciados. Estes atributos são: *Aluno.endereço*, *Atleta.endereço* e *Pessoa.endp* que armazenam os endereços das pessoas cadastradas nas secretarias de Educação, Esportes e no SAE, respectivamente e os atributos *Escola.endereço* e *Controle.endereço* que armazenam os endereços das escolas e dos locais de prática de esportes da prefeitura, respectivamente.

Os atributos federados que estão relacionados com endereço são *EndereçoPessoa* e *EndereçoAtividade*. No entanto, a ligação entre os dados convencionais e os mapas exigirá uma padronização dos endereços armazenados. Um mesmo endereço pode estar atualmente escrito de várias formas em cada componente e isto torna impossível georeferenciar um atributo federado, ou seja, estabelecer uma ligação entre um atributo federado e uma região no mapa.

Para que a solução federada efetivamente funcione, é preciso antes de mais nada, padronizar os endereços em cada componente. Isto é feito através de um banco de dados único de endereços que sirva como modelo para os componentes. Assim, cada endereço armazenado nos componentes deve passar por um processo de validação e, se necessário, uma redigitação utilizando os padrões estabelecidos.

### 5.3 Resumo do Capítulo

Este capítulo apresentou uma revisão dos principais conceitos relativos a sistemas de informações geográficas (SIGs) e como a metodologia de integração pode ser aplicada em ambientes que utilizam SIGs. A definição adotada nesta dissertação foi a mesma proposta em [CCH<sup>+</sup>96]: SIGs são sistemas automatizados usados para armazenar, analisar e manipular dados geográficos.

Os dados dos SIGs podem estar no formato *varredura* ou *vetorial*. A diferença principal entre eles é a precisão dos dados, maior no formato vetorial. Assim, aplicações voltadas para fins urbanos e rurais geralmente utilizam o formato vetorial enquanto que as aplicações voltadas para fins ambientais utilizam o formato varredura.

As organizações que passam pelo processo de implantação de SIG são bons alvos para a aplicação da metodologia proposta nesta dissertação, já que muitos sistemas precisam ser integrados ao SIG a ser implantado. A presença de bancos de dados heterogêneos e sistemas legados nestas organizações é freqüente e a metodologia proposta pode ser aplicada para resolver tanto a integração dos dados convencionais quanto a integração dos dados espaciais. No entanto, os dados espaciais precisam de um tratamento à parte para resolver outros tipos de heterogeneidade como: diferenças de escala, representação de dados, projeção, qualidade e aspectos temporais dos dados. Na maioria dos casos, esta heterogeneidade é resolvida pelo próprio usuário do SIG, através de procedimentos de padronização.



Os SIGs foram considerados dentro do contexto desta dissertação através de três casos diferentes. O primeiro caso considera SIGs acessando bancos de dados componentes da federação mas apenas os dados convencionais de cada componente são federados. O segundo caso considera a federação composta por bancos de dados geográficos, sendo que a integração é feita tanto dos dados convencionais como dos espaciais. O terceiro e último caso considera a utilização de duas federações, uma contendo apenas dados convencionais e a outra contendo dados espaciais e convencionais. Apenas o segundo e o terceiro casos necessitam de passos adicionais na metodologia de integração para permitir a padronização dos dados espaciais.

Por fim, foi mostrado como um SIG pode utilizar o sistema federado especificado no capítulo 4, enfatizando a falta de padronização no armazenamento de endereços pelos componentes da federação.

# Capítulo 6

## Conclusões e Extensões

O principal objetivo desta dissertação foi apresentar um conjunto de passos para se estabelecer um sistema integrado de dados através da criação de um sistema de bancos de dados federados que possui como componentes, bancos de dados heterogêneos e sistemas legados.

A metodologia de integração proposta nesta dissertação fornece linhas gerais para a integração de quaisquer bancos de dados e sistemas legados de uma organização. Evidentemente, pode haver algumas variações nos passos desta metodologia, dependendo das particularidades dos ambientes computacionais de cada organização. No entanto, ela apresenta um avanço quanto à situação atual, estendendo um conjunto básico de passos a serem seguidos. Isto facilita a documentação e execução de cada fase do processo de integração. Foi constatado também que em muitos casos não é possível realizar a integração de dados e, nestes casos, a reconstrução dos sistemas passa a ser inevitável.

A metodologia também foi inserida no contexto de sistemas de informações geográficas, apresentando as facilidades e limitações existente ao se considerar SIGs em sistemas de bancos de dados federados. Esta parte serve de alerta às tentativas de realizar este tipo de integração.

O estudo de caso realizado na cidade de Paulínia possibilitou aplicar a metodologia utilizando especificações de bancos de dados reais, o que auxiliou o processo de refinamento da metodologia e permitiu associar problemas de ordem prática à teoria aqui apresentada.

As principais contribuições deste trabalho são:

- Análise das opções de integração e migração de dados de sistemas legados visando sua utilização em sistemas de bancos de dados federados;
- Proposta de uma metodologia de integração de bancos de dados heterogêneos e sistemas legados através da construção de um sistema de bancos de dados federados.

- Aplicação da metodologia a um caso real composto por sistemas legados e sistemas de bancos de dados da cidade de Paulínia-SP;
- Linhas gerais para adaptação da metodologia, buscando suportar a inserção de SIGs ao sistema de bancos de dados federados;
- Validação através de um protótipo para um sistema de bancos de dados federados fracamente acoplado.

A atenção especial dedicada aos sistemas legados nesta dissertação transcende o problema de integração de dados, permitindo uma melhor compreensão destes sistemas. Assim, as opções apresentadas para o tratamento destes sistemas podem ser aplicadas de forma isolada para torná-los mais flexíveis e administráveis, já que sistemas legados são, por si só, um problema em potencial.

A metodologia de integração de bancos de dados heterogêneos e sistemas legados encapsula todos os conceitos aqui apresentados de uma forma organizada, permitindo um direcionamento mais embasado e seguro para os projetistas de SBDFs. A aplicação da metodologia aos problemas enfrentados pela cidade de Paulínia possibilitou acompanhar como a metodologia se comporta na prática.

Embora a dissertação tenha apresentado os problemas relacionados com a integração de dados espaciais, esta abordagem precisa ser melhor pesquisada e testada, estendendo as soluções aqui apresentadas.

Por fim, o protótipo do sistema de bancos de dados federados, além de permitir a utilização prática da metodologia proposta, contribui para constatar as limitações do trabalho de Zhao [Zha97] que, até o presente, não implementou de fato sua proposta.

Como extensões a esta dissertação, são propostos os seguintes tópicos:

- Realização de mais testes sobre cada passo da metodologia, em especial utilizando sistemas legados mais complexos e um maior número de componentes;
- Otimização do processamento de consulta no SBDF;
- Conversão do esquema relacional do SBDF para um modelo orientado a objetos, possivelmente o GMOD [Pir97], para que possa ser usado por sistemas orientados a objetos, sem a necessidade de conversões;
- Implementação de um SBDF mais robusto, baseado no protótipo especificado, oferecendo uma interface amigável e que permita aos usuários solucionar as ambigüidades geradas durante os processamentos de consultas;

- Extensão da metodologia considerando os aspectos de distribuição de dados, acesso concorrente e recuperação de informação. O trabalho não considerou estes aspectos porque a distribuição nem sempre é encontrada em sistemas federados;
- Estudo mais aprofundado dos problemas e soluções envolvendo integração de SIGs.

Testes exaustivos sobre cada passo da metodologia são importantes para identificar casos mais específicos ou patológicos que requerem adaptações da metodologia, tornando-a mais robusta. Otimizações no processamento de consultas realizadas sobre o SBDF também são necessárias para federações com muitos componentes. Já a possibilidade de se ter uma federação orientada a objetos permite contemplar os SIGs que seguem a tendência de modelar dados geográficos através de modelos orientados a objetos. O protótipo especificado serve de núcleo para implantar um SBDF que pode ser a base para futuros sistemas comerciais, além de consolidar ainda mais a metodologia proposta. Uma extensão importante do ponto de vista comercial é suportar distribuição de dados e, mais importante, facilitar os processos de controle de concorrência e recuperação de dados em caso de falhas, tanto no SBDF quanto nos bancos de dados componentes. Por fim, os problemas relacionados com integração de SIGs precisam ser melhor analisados, já que este assunto possui um grande potencial para pesquisas.

# Bibliografia

- [Agu95] Aguiar, C. D. Integração de sistemas de banco de dados heterogêneos em aplicações de planejamento urbano. Tese de mestrado, Universidade Estadual de Campinas - UNICAMP, 1995.
- [AMR94] Aiken, P., Muntz, A., e Richards, R. Dod legacy systems - reverse engineering data requirements. *Communications of the ACM*, 37(5):26–41, 1994.
- [AP87] Azar, N. e Pichat, G. Translation of an extended entity-relationship model into the universal relation with inclusion formalism. In Spaccapietra, S., editor, *Entity-Relationship Approach*, páginas 253–268, North-Holland, 1987.
- [BDK92] Buneman, P., Davidson, S., e Kosky, A. Theoretical aspects of schema merging. In *Proc. Extending Database Technology*, páginas 152–167, 1992.
- [Ber96] Bernstein, P. A. Middleware. *Communications of the ACM*, 39(2):86–98, 1996.
- [BFK95] Breche, P., Ferrandina, F., e Kuklok, M. Simulation of schema change using views. *Springer Verlag Lecture Notes in Computer Science, LNCS 978*, páginas 247–258, 1995.
- [BHP94] Bright, M. W., Hurson, A. R., e Pakzad, S. Automated resolution of semantic heterogeneity in multidatabases. *ACM Transactions on Database Systems*, 19(2):212–253, 1994.
- [BL86] Batini, C. e Lenzerini, M. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
- [BLF97] Batini, C., Longobardi, G., e Fornasiero, S. An experience of integration of conceptual schemas in the italian public administration. *Conceptual Proceedings ER' 97*, páginas 313–322, 1997.
- [Bre90] Breitbart, Y. Multidatabases interoperability. *SIGMOD RECORD*, 19(3):53–60, 1990.

- [BS81] Bancilhon, F. e Spyrtatos, N. Update semantics of relational views. *ACM Transactions on Database Systems*, 6(4):557-575, 1981.
- [BS95] Brodie, M. L. e Stonebraker, M. *Migrating Legacy Systems - Gateways, Interfaces and the Incremental Approach*. Morgan Kaufmann Publishers, Inc., 1995.
- [CA97] Castano, S. e Antonellis, V. Semantic dictionary design for database interoperability. *Thirteenth International Conference on Data Engineering*, páginas 43-54, 1997.
- [CCH<sup>+</sup>96] Câmara, G., Casanova, M. A., Hemerly, A. S., Magalhaes, G. C., e Medeiros, C. M. B. Anatomia de sistemas de informação geográfica. *Escola de Computação*, 1996.
- [Cer96] Cereja, N. Visões em sistemas de informações geográficas - modelo e mecanismos. Tese de mestrado, Universidade Estadual de Campinas - UNICAMP, 1996.
- [Che76] Chen, P. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9-36, 1976.
- [Cif95] Ciferri, R. R. Um benchmark voltado para a análise de desempenho de sistemas de informações geográficas. Tese de mestrado, Universidade Estadual de Campinas - UNICAMP, 1995.
- [CNC83] Chung, I., Nakamura, F., e Chen, P. A decomposition of relations using the entity-relationship approach. In Chen, P. P., editor, *Entity-Relationship Approach to Information Modeling and Analysis*, páginas 149-171, North-Holland, 1983.
- [Cod70] Codd, E. F. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377-387, 1970.
- [DA83] Dumpala, S. e Arora, S. Schema translation using the entity-relationship approach. In Chen, P. P., editor, *Entity-Relationship Approach to Information Modeling and Analysis*, páginas 337-356, North-Holland, 1983.
- [DC83] Dogac, A. e Chen, P. Entity-relationship model in the ansi/sparc framework. In Chen, P. P., editor, *Entity-Relationship Approach to Information Modeling and Analysis*, páginas 357-374, North-Holland, 1983.

- [DK97] Davidson, S. B. e Kosky, A. S. Wol: A language for database transformations and constraints. *Thirteenth International Conference on Data Engineering*, páginas 55–65, 1997.
- [EN89] Elmasri, R. e Navathe, S. *Fundamentals of Database Systems*. The Benjamin/Cummings Publishing Company Inc., 1989.
- [EWH85] Elmasri, R., Weeldreyer, J., e Hevner, A. The category concept: An extension to entity-relationship model. In Schneider, H. J., editor, *Data and Knowledge Engineering*, páginas 75–146, North-Holland, 1985.
- [FMU82] Fagin, R., Mendelzon, A. O., e Ullman, J. D. A simplified universal relation assumption and its properties. *ACM Transactions on Database Systems*, 7(3):343–360, 1982.
- [Fon92] Fong, J. S. Methodology for schema translation from hierarchical or network into relational. *Information and Software Technology*, 19(3):159–174, 1992.
- [Geo91] Georgakopoulos, D. Multidatabase recoverability and recovery. *First International Workshop on Interoperability in Multidatabase Systems*, páginas 348–355, 1991.
- [GQ94] Gong, L. e Qian, X. Computational issues in secure interoperation. *IEEE Symposium on Research in Security and Privacy*, páginas 1–10, 1994.
- [GRS91] Georgakopoulos, D., Rusinkiewicz, M., e Sheth, A. On serializability of multidatabase transactions through forced local conflicts. *Seventh International Conference on Data Engineering*, páginas 314–323, 1991.
- [HB91] Hurson, A. R. e Bright, M. W. Multidatabase systems: An advanced concept in handling distributed data. *Advances in Computers*, 32:147–200, 1991.
- [HBRY91] Hsu, C., Bouziane, M., Rattner, L., e Yee, L. Information resources management in heterogeneous, distributed environments: A metadatabase approach. *IEEE Transactions on Software Engineering*, 17(6):604–624, 1991.
- [HM85] Heimbigner, D. e McLeod, D. A federated architecture for information management. *Integration of Information Systems: Bridging Heterogeneous Databases*, 3(3):46–71, 1985.
- [HMN91] Hornick, M. F., Morrison, J. D., e Nayeri, F. Integrating heterogeneous, autonomous, distributed applications using the dom prototype. Relatório técnico, GTE Laboratories, 1991.

- [HMZ90] Heiler, S., Manola, F., e Zdonik, S. An object-oriented database approach to federated systems, 1990.
- [Hul97] Hull, R. Managing semantic heterogeneity in databases: A theoretical perspective. *PODS97*, 1997.
- [Jr95] Jr, G. S. Novak. Creation of views for reuse of software with different data representations. *IEEE Transactions on Software Engineering*, 21(12):993–1005, 1995.
- [LH90] Lerner, B. S. e Habermann, A. N. Beyond schema evolution to database reorganization. *ECOOP/OOPSLA '90 Proceedings*, páginas 67–77, 1990.
- [LL97] Lee, M. L. e Ling, T. W. Resolving constraint conflicts in the integration of entity-relationship schemas. *Conceptual Proceedings ER' 97*, páginas 394–407, 1997.
- [LS97] Le, P. D. e Srinivasan, B. A migration tool to support resource and load sharing in heterogeneous computing environments. *Computer Communications*, 20:361–374, 1997.
- [MHG+92] Manola, F., Heiler, S., Georgakopoulos, D., Hornick, M., e Brodie, M. Distributed object management. *International Journal of Intelligent and Cooperative Information Systems*, 1(1):5–42, 1992.
- [MLRN98] Mendonca, L. M. L., Laender, A. H. F., e Ribeiro-Neto, B. A. Especificação de consultas incompletas em bancos de dados relacionais. *Anais XIII Simpósio Brasileiro de Banco de Dados*, páginas 185–200, 1998.
- [MM91] Mamou, J. e Medeiros, C. M. B. Interactive manipulation of object-oriented views. *Seventh International Conference on Data Engineering*, páginas 60–69, 1991.
- [MNB+94] Markosian, L., Newcomb, P., Brand, R., Burson, S., e Kitzmiller, T. Using an enabling technology to reengineer legacy systems. *Communications of the ACM*, 37(5):58–70, 1994.
- [Mot98] Motz, R. Instanciating integrated schemas. *Anais XIII Simpósio Brasileiro de Banco de Dados*, páginas 53–68, 1998.
- [MRS+87] Maier, D., Rozenshtein, D., Salveter, S., Stein, J., e Warren, D. S. Pique: a relation query language without relations. *Information Systems*, 12(3):317–335, 1987.



- [MYW<sup>+</sup>93] Meng, W., Yu, C., Wang, G., Pham, T., e Dao, S. Construction of a relational front-end for object-oriented database systems. *IEEE Data Engineering Conference*, páginas 476–483, 1993.
- [NEK94] Ning, J. Q., Engberts, A., e Kozaczynski, W. Automated support for legacy code understanding. *Communications of the ACM*, 37(5):50–57, 1994.
- [Ogi97] Ogis, . Saif. <http://www.widarsey.com/infosafe/saif/saifhome.html>, 1997.
- [Oli93] Oliveira, R. L. Transparência de modelos em sistemas de bancos de dados homogêneos. Tese de mestrado, Universidade Estadual de Campinas - UNICAMP, 1993.
- [OM93] Oliveira, R. L. e Magalhaes, G. C. Metodologias para conversão de esquemas em sistemas de bancos de dados heterogêneos. Relatório Técnico DCC-17/93, Universidade Estadual de Campinas - UNICAMP, 1993.
- [PH95] Pernul, G. e Hasenauer, H. Combining reverse with forward database engineering - a step forward to solve the legacy system dilemma. *Springer Verlag Lecture Notes in Computer Science - 978 DEXA 95*, páginas 177–186, 1995.
- [Pir97] Pires, F. *Um Ambiente Computacional para Modelagem de Aplicações Ambientais*. Tese de doutorado, Universidade Estadual de Campinas - UNICAMP, 1997.
- [Pit96] Pitrik, R. M. Requirements and comparison of view mechanisms for object-oriented databases. *Information Systems*, 21(3):229–251, 1996.
- [Qia93] Qian, X. Semantic interoperation via intelligent mediation. *Proceedings of the 1993 International Workshop on Research Issues in Data Engineering*, páginas 1–4, 1993.
- [Qia96] Qian, X. Correct schema transformations. *International Conference on Extending Database Technology - EDBT*, 1996.
- [QL94] Qian, X. e Lunt, T. F. Semantic interoperation: A query mediation approach. Relatório Técnico SRI-CSL-94-02, Computer Science Laboratory, SRI International, 1994.
- [QR95] Qian, X. e Raschid, L. Query interoperation among object-oriented and relational databases. *Proceedings of the 1995 International Conference on Data Engineering*, páginas 1–8, 1995.

- [Rob97] Robertson, P. Integrating legacy systems with modern corporate applications. *Communications of the ACM*, 40(5):39–46, 1997.
- [SCG<sup>+</sup>97] Shekhar, S., Coyle, M., Goyal, B., Liu, D., e Sarkar, S. Data models in geographic information systems. *Communications of the ACM*, 40(4):103–111, 1997.
- [SD94] Smith, D. J. e Dearnley, P. A. Overview and description of circe database federation. *School of Information Systems, University of East Anglia*, 1994.
- [SL90] Sheth, A. P. e Larson, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–235, 1990.
- [SL94] Sacramento, E. R. e Laender, A. H. R. Uma abordagem orientada a objetos para o projeto lógico de bancos de dados relacionais. *Anais IX Simpósio Brasileiro de Banco de Dados*, páginas 110–126, 1994.
- [SLL81] Su, S. Y. W., Lam, H., e Lo, D. H. Transformation of data traversals and operations in application programs to account for semantic changes of databases. *ACM Transactions on Database Systems*, 6(2):255–294, 1981.
- [SM97] Soares, H. R. e Medeiros, C. M. B. Integração de sistemas legados a bancos de dados para desenvolvimento de aplicações sig urbanas. *Anais GIS BRASIL97 - Congresso e Feira para Usuários de Geoprocessamento*, 1997.
- [SSR94] Sciore, E., Siegel, M., e Rosenthal, A. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2):254–290, 1994.
- [TK78] Tsichritzis, D. e Klug, A. The ansi/x3/sparc dbms framework. *Information Systems*, 3,4, 1978.
- [Tri96] Triantafillou, P. Independent recovery in large-scale distributed systems. *IEEE Transactions on Software Engineering*, 22(11):812–826, 1996.
- [VL97] Vidal, V. M. P. e Lóscio, B. F. Especificação de mediadores para acesso e atualização de múltiplas bases de dados. *Anais XII Simpósio Brasileiro de Banco de Dados*, 1997.
- [Zha97] Zhao, J. L. Schema coordination in federated database management: a comparison with schema integration. *Decision Support Systems*, 20:243–257, 1997.

- [ZWM97] Zhuge, Y., Wiener, J. L., e Molina, H. G. Multiple view consistency for data warehousing. *Thirteenth International Conference on Data Engineering*, páginas 289–300, 1997.