**Metamodelo para Controle de
Estratégias Assíncronas de
Replicação de Dados**

Carlos Henrique Maia Braga

Dissertação para o Programa de Mestrado
submetida em cumprimento parcial dos requisitos
para obtenção do grau de
Mestre em Ciência da Computação

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Banco de Dados
Orientador: Prof. Dr. João Eduardo Ferreira

Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação
- São Paulo, novembro de 2001 -

Metamodelo para Controle de Estratégias Assíncronas de Replicação de Dados

Este exemplar corresponde à redação
final da dissertação apresentada por
Carlos Henrique Maia Braga e
aprovada pela comissão julgadora.

São Paulo, novembro de 2001.

Banca Examinadora:

Prof. Dr. João Eduardo Ferreira (orientador)
Prof. Dr. Caetano Traina Júnior
Prof. Dr. Francisco Carlos da Rocha Reverbel

IME - USP
ICMC - USP
IME - USP

Dedico este trabalho
fundamentalmente a Deus, por
permitir a continuidade desta
caminhada e à minha namorada,
que soube me entender nos
momentos mais difíceis.

Agradecimentos

A meu orientador, Prof. João Eduardo Ferreira, pelo apoio e incentivo, pela confiança, amizade e consideração.

A todo o Departamento de Informática da Reitoria da USP, onde trabalho; principalmente, aos meus Diretores, que permitiram a realização deste trabalho, através do estudo de caso realizado, a utilização do ambiente e dos recursos disponíveis, e também pela compreensão de minhas ausências em horários de aula.

Aos ex-colegas de serviço que sempre me animaram; e a todos que ainda estão ao meu lado, apoiando-me e incentivando-me, principalmente à Marga, pelos comentários e sugestões do texto.

A todos do Instituto de Matemática e Estatística da USP que, direta ou indiretamente, me ajudaram em todo este período de estudo.

Aos professores Cristina, Dilma, Hernan, Finger, Reverbel e Siang pelo aprendizado em suas aulas, e ao professor Routo Terada pelas sugestões como orientador de programa.

Aos meus colegas, da velha guarda, Cibele, Euler, Isabel, Maria do Carmo, Myrthes e Vera.

E aos mais recentes, Eduardo, Luciano, Márcio, Marcelo e Uirá que sempre estiveram do meu lado.

Finalmente, aos meus pais e minhas irmãs, pelo amor e carinho que sempre tiveram, pelas cobranças e incentivos, que me fazem sempre olhar mais e mais para frente, e simplesmente por sempre “*estarem*” comigo.

Resumo

Esta dissertação trata das formas e técnicas de distribuição de dados, através da replicação assíncrona, integrado com o projeto de banco de dados com modularização, criando um projeto único chamado de Projeto de Banco de Dados com Modularização e Replicação.

A elaboração do projeto tem em vista a utilização de um metamodelo de dados, capaz de manter todas as informações necessárias para se fazer o controle no uso de diversas estratégias de replicação, conforme o Projeto de Modularização e Replicação de Dados, realizado em um modelo corporativo de dados.

Foi desenvolvida uma ferramenta que permite que o metamodelo seja alimentado, possibilitando o gerenciamento e manutenção do projeto de replicação e particionamento de dados, após o cadastro lógico do modelo conceitual global de dados.

Para garantir a utilização correta de diversas estratégias de replicação do metamodelo em questão, foi implementado um verificador para analisar as consistências dos projetos de replicação definidos para o banco de dados.

Tal verificador, por sua vez, foi incorporado à ferramenta desenvolvida pelo Departamento de Informática da Reitoria da USP, para garantir a consistência de suas bases de dados.

Abstract

This dissertation is about the forms and techniques of distributed data, through the asynchronous replication, integrated with the Database Modularization Design, creating one single project named Database Modularization and Replication Design.

The creation of the project aims the use of a data meta-model, capable of maintaining all necessary information to control the use of several replication strategies, according to the Modularization and Data Replication Design, accomplished in a corporate data model.

A tool was developed to allow the feeding of the meta-model, making easy the management and the maintenance of the Modularization and Data Replication Design, after the logical input of the global conceptual data model.

To guarantee the correct use of several replication strategies of the meta-model, a checker to analyze the consistencies of the Replication Design defined for the database was implemented.

Such checker, was incorporated to the tool developed by the Departamento de Informática da Reitoria da USP, to guarantee the consistency of its databases.

Sumário

1.	INTRODUÇÃO	1
2.	DISTRIBUIÇÃO DE DADOS.....	3
2.1.	CONCEITOS DE BANCOS DE DADOS DISTRIBUÍDOS.....	3
2.1.1.	<i>Vantagens.....</i>	3
2.1.2.	<i>Funções Adicionais dos Bancos de Dados Distribuídos.....</i>	5
2.2.	FORMAS DE DISTRIBUIÇÃO DE DADOS	6
2.2.1.	<i>Fragmentação de Dados</i>	6
2.2.2.	<i>Replicação de Dados</i>	8
3.	MODULARIZAÇÃO DO BANCO DE DADOS	10
3.1.	PROJETO DE BANCO DE DADOS COM MODULARIZAÇÃO	10
3.1.1.	<i>Coleta e Análise de Requisitos de Modularização.....</i>	12
3.1.2.	<i>Projeto de Modularização</i>	12
4.	ASPECTOS DA REPLICAÇÃO DE DADOS.....	17
4.1.	MODELO DE SISTEMA DE BANCO DE DADOS DISTRIBUÍDO.....	17
4.1.1.	<i>Controle de Concorrência.....</i>	18
4.1.2.	<i>Controle de Atomicidade</i>	19
4.1.3.	<i>Consistência Mútua em Bancos de Dados Replicados.....</i>	19
4.2.	FORMAS DE CONSISTÊNCIA DA DISTRIBUIÇÃO DE DADOS.....	20
4.2.1.	<i>Síncrona.....</i>	20
4.2.2.	<i>Assíncrona.....</i>	20
4.3.	ESTRATÉGIAS PARA GARANTIR A CONSISTÊNCIA DA REPLICAÇÃO DE DADOS.....	21
4.3.1.	<i>Consistência Forte</i>	22
4.3.2.	<i>Consistência Fraca</i>	23
4.3.3.	<i>Consistência Relaxada.....</i>	24
4.3.4.	<i>Manual</i>	25
4.3.5.	<i>Com um Gerenciador de Replicação</i>	26
5.	FERRAMENTA GERENCIADORA PARA CONTROLE ASSÍNCRONO DE REPLICAÇÃO DE DADOS.....	27
5.1.	COMPONENTES	27
5.2.	ARQUITETURA.....	28
5.3.	EXEMPLOS DE ESTRATÉGIAS DE CONSISTÊNCIA ASSÍNCRONA DE REPLICAÇÃO DE DADOS	30
5.3.1.	<i>Sites Replicados somente para leitura</i>	30
5.3.2.	<i>Atualização remota do Site Primário por conexões do cliente</i>	31
5.3.3.	<i>Site Remoto requisita atualização do Site Primário</i>	32
5.3.4.	<i>Site Remoto requisita atualização do Site Primário e mantém atualização temporária no local</i>	33
5.3.5.	<i>Fragmentos Primários Distribuídos</i>	35
5.3.6.	<i>Site Corporativo recebe dados dos outros Sites (Corporate Roll-Up).....</i>	37

5.3.7.	<i>Sites Corporativos enviando Dados para Sites Replicados</i>	39
5.3.8.	<i>Sites Primários Redundantes – esperando queda do primário (Warm Standby)</i>	40
5.3.9.	<i>Sites Primários Redundantes – trocando os primários (Switching Over)</i>	41
5.3.10.	<i>Ponto a Ponto (Peer-to-Peer)</i>	42
5.3.11.	<i>Site Primário Dinâmico (Dynamic Site Ownership)</i>	43
5.4.	CONFLITOS NA REPLICAÇÃO DE DADOS.....	44
6.	MODULARIZAÇÃO E REPLICAÇÃO DE DADOS	46
6.1.	ADAPTAÇÃO DO PROJETO DE REPLICAÇÃO NAS FASES DA MODULARIZAÇÃO	46
6.1.1.	<i>Coleta e Análise de Tipos de Replicação</i>	47
6.1.2.	<i>Projeto de Modularização e Replicação de Dados</i>	49
7.	REPLICAÇÃO DE DADOS E UM ESTUDO DE CASO	64
7.1.	CENÁRIO DA USP.....	64
7.2.	TIPOS DE TABELAS.....	66
7.3.	ESTRATÉGIAS DE REPLICAÇÃO UTILIZADAS NO AMBIENTE USP	67
7.4.	FORMAS DE REMOÇÃO	77
7.5.	METAMODELO DE APOIO À REPLICAÇÃO	79
7.6.	VERIFICADOR DE CONSISTÊNCIA DE REPLICAÇÃO EM UM BANCO DE DADOS	82
7.7.	IMPLEMENTAÇÃO FÍSICA DA BASE DE DADOS E FERRAMENTA DE APOIO AO METAMODELO.....	83
8.	CONCLUSÃO	85
8.1.	CONSIDERAÇÕES INICIAIS	85
8.2.	CONTRIBUIÇÕES DIRETAS	85
8.3.	CONTRIBUIÇÕES DECORRENTES.....	86
8.3.1.	<i>Validação do Trabalho com a Utilização do Estudo de Caso</i>	86
8.3.2.	<i>Delimitação e Diminuição do Grau de Heterogeneidade dos Sistemas</i>	87
8.3.3.	<i>Contribuição para Projetos e Técnicas de Fragmentação e Distribuição de Dados mais Eficientes</i>	87
8.4.	FUTURAS PESQUISAS.....	88
8.4.1.	<i>Transformar as Diretrizes para Controle das Formas de Réplicas em Metodologia</i>	88
8.4.2.	<i>Incorporar o Controle em uma Ferramenta CASE</i>	88
8.4.3.	<i>Alterar o Catálogo do Gerenciador de Replicação</i>	88
8.4.4.	<i>Desenvolver Regras Ativas para Controle de Consistências Complexas</i>	89
8.4.5.	<i>Apoio à Data-Mining Distribuído</i>	89
	APÊNDICE A	90
A.1.	GERENCIADOR LÓGICO DE REPLICAÇÃO DE DADOS.....	90
	APÊNDICE B	100
B.1.	DETALHES DO METAMODELO	100
	APÊNDICE C	104
C.1.	EXEMPLO DA VERIFICAÇÃO NO AMBIENTE USP	104
	BIBLIOGRAFIA	115

Lista de Figuras

FIGURA 1 – FASES DO PROJETO DE BASE DE DADOS COM MODULARIZAÇÃO	11
FIGURA 2 – REPRESENTAÇÃO DE UM MÓDULO DE DADOS.....	16
FIGURA 3 – ARQUITETURA DE REPLICAÇÃO DO REPLICATION SERVER SYBASE	28
FIGURA 4 – SITES REPLICADOS SOMENTE PARA LEITURA	30
FIGURA 5 – ATUALIZAÇÃO REMOTA DO SITE PRIMÁRIO POR CONEXÕES DO CLIENTE	31
FIGURA 6 – SITE REMOTO REQUISITA ATUALIZAÇÃO DO SITE PRIMÁRIO	32
FIGURA 7 – SITE REMOTO REQUISITA ATUALIZAÇÃO DO SITE PRIMÁRIO E MANTÉM ATUALIZAÇÃO TEMPORÁRIA NO LOCAL	33
FIGURA 8 – FRAGMENTOS PRIMÁRIOS DISTRIBUÍDOS	35
FIGURA 9 – SITE CORPORATIVO RECEBE DADOS DOS OUTROS SITES (CORPORATE ROLL-UP)	37
FIGURA 10 – SITES CORPORATIVOS ENVIANDO DADOS PARA SITES REPLICADOS	39
FIGURA 11 – SITES PRIMÁRIOS REDUNDANTES – ESPERANDO QUEDA DO PRIMÁRIO	40
FIGURA 12 – SITES PRIMÁRIOS REDUNDANTES – TROCANDO OS PRIMÁRIOS	41
FIGURA 13 – PONTO A PONTO (PEER-TO-PEER)	42
FIGURA 14 – SITE PRIMÁRIO DINÂMICO (DYNAMIC SITE OWNERSHIP)	43
FIGURA 15 – CONFLITOS NA REPLICAÇÃO DE DADOS.....	44
FIGURA 16 – FASES DO PROJETO DE BASE DE DADOS COM MODULARIZAÇÃO E REPLICAÇÃO DE DADOS..	47
FIGURA 17 – SUBESQUEMAS OBTIDOS A PARTIR DO ESQUEMA CONCEITUAL GLOBAL DE DADOS.....	51
FIGURA 18 – ESQUEMA DE DADOS DO MÓDULO M_1	59
FIGURA 19 – ESQUEMA DE DADOS DO MÓDULO M_2	60
FIGURA 20 – ESQUEMA DE DADOS DO MÓDULO M_3	61
FIGURA 21 – ESQUEMA DE DADOS DO MÓDULO M_4	62
FIGURA 22 – MODELO DE REPRESENTAÇÃO DA TABELA EM UM BANCO DE DADOS	68
FIGURA 23 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 1	68
FIGURA 24 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 2	68
FIGURA 25 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 3	69
FIGURA 26 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 4	69
FIGURA 27 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 5	70
FIGURA 28 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 6	70
FIGURA 29 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 7	71
FIGURA 30 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 8	71
FIGURA 31 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 9	72
FIGURA 32 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 10	72
FIGURA 33 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 11	73
FIGURA 34 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 12	73
FIGURA 35 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 13	74
FIGURA 36 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 14	74

<i>FIGURA 37 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 15</i>	75
<i>FIGURA 38 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 16</i>	75
<i>FIGURA 39 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 17</i>	76
<i>FIGURA 40 – REPRESENTAÇÃO DA REPLICAÇÃO COM A ESTRATÉGIA 18</i>	76
<i>FIGURA 41 – MODELO RELACIONAL DO METAMODELO DE APOIO À REPLICAÇÃO</i>	80
<i>FIGURA 42 – TELA DE SERVIDORES DE BANCO DE DADOS</i>	91
<i>FIGURA 43 – TELA DE BANCO DE DADOS</i>	92
<i>FIGURA 44 – TELA DE ESTRATÉGIAS DE REPLICAÇÃO</i>	93
<i>FIGURA 45 – TELA DE REPRESENTAÇÃO DO MODELO FÍSICO DE REPLICAÇÃO DE DADOS</i>	94
<i>FIGURA 46 – TELA DE PUBLICAÇÕES</i>	95
<i>FIGURA 47 – TELA DE SUBSCRIÇÕES</i>	96
<i>FIGURA 48 – TELA DE RELACIONAMENTOS</i>	97
<i>FIGURA 49 – TELA DE CONSOLIDAÇÃO DOS PROCEDIMENTOS DE DISTRIBUIÇÃO</i>	98
<i>FIGURA 50 – TELA DE CONSISTÊNCIA DO BANCO DE DADOS</i>	99
<i>FIGURA 51 – DETALHAMENTO DO METAMODELO DE APOIO À REPLICAÇÃO</i>	103

Lista de Tabelas

<i>TABELA 1 – TIPO DE COMPARTILHAMENTO DE CADA ELEMENTO DO ESQUEMA CONCEITUAL GLOBAL DE DADOS.....</i>	<i>52</i>
<i>TABELA 2 – DEFINIÇÃO DOS MÓDULOS DA BASE DE DADOS.....</i>	<i>54</i>
<i>TABELA 3 – DEFINIÇÃO DAS ESTRATÉGIAS DE REPLICAÇÃO PARA CADA ELEMENTO DOS MÓDULOS DA BASE DE DADOS.....</i>	<i>58</i>

1. Introdução

Técnicas de distribuição de dados estão sendo aplicadas para solucionar problemas como autonomia de site¹, desempenho, disponibilidade e confiabilidade dos dados. Porém, há dificuldades em se planejar, manter e gerenciar um ambiente de distribuição de dados mais complexo. Com o intuito de esclarecer alguns aspectos desta complexidade, serão ilustrados nos próximos capítulos, as características do ambiente de distribuição de dados, o modo de projetar um banco de dados através da modularização e também alguns aspectos e estratégias de replicação de dados.

Em bancos de dados corporativos, encontram-se muitos sistemas aplicativos que compartilham informações entre si, por isso, é necessário um estudo sobre como e qual a melhor forma de efetuar a distribuição de banco de dados (capítulo 2).

A modularização de banco de dados (capítulo 3) cria um nível de abstração do modelo conceitual global por módulos de negócio, que podem ser utilizados por vários sistemas.

A elaboração de um projeto de replicação de dados (capítulo 4), como forma de distribuição das informações, tem como um dos benefícios o aumento na autonomia dos sistemas aplicativos.

Nesse sentido, para melhorar a autonomia dos sistemas de forma controlada e para facilitar a manutenção dos mesmos, viu-se a necessidade de integrar a modularização à replicação de dados em um único projeto (capítulo 6).

Contudo, é necessária que essa interação seja feita de forma coordenada e que seja facilmente gerenciada, por parte dos administradores de dados e dos administradores de gerenciadores de banco de dados, porque, na fase de projeto de modularização, pode-se encontrar a interseção entre dois ou mais subesquemas de sistemas diferentes, oriundos do mesmo modelo conceitual global. Nesse momento, a utilização da replicação de dados vem complementar o projeto de alocação dos dados, conforme as necessidades de cada sistema e sua modularização.

Sabendo como uma ferramenta responsável pelo gerenciamento assíncrono de replicação de dados funciona (capítulo 5), e suas estratégias de replicação, será

¹ Nós de rede com recursos computacionais (hardware e software).

demonstrado, através de um estudo de caso (capítulo 7), como elaborar um projeto de replicação de dados, conforme as estratégias implementadas por esse gerenciador.

É possível fazer o controle de como os dados devem ser distribuídos, de acordo com cada sistema que utiliza o modelo corporativo, através da utilização de um metamodelo de dados (item 7.5). Este metamodelo é responsável por armazenar todas as informações necessárias, como tabelas, colunas, relacionamentos, estratégias de replicação e o próprio projeto de distribuição de dados dessas tabelas, nos diversos bancos de dados, conforme as particularidades e aspectos da replicação de dados (capítulo 4).

Assim, o metamodelo deve atender às necessidades das principais estratégias disponíveis pelo gerenciador de replicação (capítulo 5) e pelas definições obtidas no Projeto de Modularização e Replicação de Dados (capítulo 6).

Após o projeto de modularização e replicação de dados, todas as informações são cadastradas no metamodelo. Tais informações são: a estratégia que se deve utilizar para cada tabela, em quais bancos de dados as tabelas devem ficar, e se elas estão replicando ou recebendo informações.

No Apêndice A, é ilustrada a ferramenta responsável por controlar as definições do projeto de distribuição de dados na USP, através do uso do metamodelo.

A construção de um verificador (item 7.6) permite a análise de possíveis inconsistências, na escolha da estratégia de replicação individualizada, no projeto de replicação. Este verificador incrementou essas funcionalidades à ferramenta (Apêndice A) desenvolvida pela USP.

2. Distribuição de Dados

Com o grande aumento na utilização de redes de computadores, na área de informática, para melhorar o compartilhamento de recursos, aumentou-se também a necessidade de se compartilhar informações contidas nos bancos de dados de cada nó da rede.

Desta forma, surgiram os sistemas de bancos de dados distribuídos, que acessam informações de diversos nós da rede (servidores).

Cada servidor atua como no sistema cliente-servidor [24], onde diversas aplicações compartilham os dados gerenciados por ele, observando-se, porém, que as consultas não são feitas especificamente para um determinado servidor, pois as informações estão espalhadas entre vários servidores de banco de dados.

2.1. Conceitos de Bancos de Dados Distribuídos

Segundo Elmasri e Navathe [22], os bancos de dados distribuídos trazem as vantagens da computação distribuída para o domínio do gerenciador do banco de dados. Um sistema de computação distribuída consiste em vários processos, não necessariamente homogêneos, que são interconectados por uma rede de computadores e que cooperam no cumprimento de tarefas partilhadas. De uma maneira geral, os sistemas de computação distribuída dividem um grande problema em pequenos pedaços e os resolvem eficientemente e de maneira coordenada.

Özsu e Valduriez [14][15] definem bancos de dados distribuídos (BDD) como uma coleção de múltiplos bancos de dados logicamente interrelacionados sobre uma rede de computadores, e um sistema gerenciador de banco de dados (SGBD) como sendo um sistema que permite o gerenciamento dos sistemas de bancos de dados distribuídos (SBDD) e que faz a distribuição de uma forma transparente para o usuário.

2.1.1. Vantagens

Elmasri e Navathe [22] consideram como principais vantagens da distribuição de dados:

- **Gerenciamento dos dados distribuídos com diferentes níveis de transparência:** idealmente, um SGBD deve ter transparência na distribuição, isto é, não deve informar detalhes de onde cada informação está fisicamente armazenada.
 - transparência da rede ou distribuição: o usuário não precisa saber dos detalhes operacionais da rede;
 - transparência na replicação: cópias dos dados podem ser armazenadas em vários sites para melhor disponibilidade, desempenho e confiabilidade. A transparência na replicação deixa o usuário sem conhecimento da existência da cópia;
 - transparência na fragmentação: existem dois tipos de fragmentação mais usuais. A fragmentação horizontal, que distribui a relação em conjunto de tuplas (linhas), e a fragmentação vertical, que distribui a relação em sub-relações, onde cada sub-relação é definida como sendo um subconjunto das colunas da relação original. A transparência na fragmentação deixa o usuário sem conhecimento da existência destes fragmentos.
- **Aumento da confiabilidade e disponibilidade:** são as maiores vantagens na distribuição de dados. Em [4], confiabilidade é definida como sendo a frequência com a qual uma ligação ou um local apresenta falhas, e disponibilidade é o grau no qual o dado pode ser acessado, ainda que algumas ligações apresentem falhas. Quando os dados e o SGBD são distribuídos, um site pode falhar, enquanto os outros continuam em operação. Somente os dados e os programas que estão no site que falhou estão inacessíveis. Em sistemas centralizados, a falha em um simples site faz com que todos os sistemas fiquem indisponíveis para todos os usuários.
- **Melhora no desempenho:** fragmentos do banco de dados são mantidos mais próximos de onde eles são mais acessados. A localização dos dados reduz a contenção de CPU, o serviço de I/O e simultaneamente diminui o tempo de acesso envolvido.
- **Possibilidade de expansão:** em um ambiente distribuído, a capacidade de escalar um sistema, em termos de aumento no tamanho do banco de dados ou de adição de mais processadores, se torna viável.

2.1.2. Funções Adicionais dos Bancos de Dados Distribuídos

Sob o ponto de vista físico, os Sistemas Gerenciadores de Banco de Dados Distribuídos (SGBDD) diferem dos SGBD centralizados por possuírem múltiplos computadores, que podem ser chamados de sites ou nós. Estes sites são conectados por uma rede de comunicação para transmitir dados e comandos entre eles.

No ambiente distribuído, os problemas encontrados em sistemas de banco de dados têm uma complexidade adicional, se comparada com sistemas centralizados. Esta complexidade adicional gera novos problemas, influenciados principalmente por três fatores.

Primeiramente, os dados no ambiente distribuído podem estar replicados. Assim, um banco de dados distribuído pode ser projetado de forma que o banco de dados inteiro, ou parte dele, resida em diferentes sites em uma rede de computadores. No entanto, não é necessário que todo site contenha o banco de dados, mas é importante que haja mais de um site com o banco de dados. Esta possível duplicação de dados está principalmente ligada às considerações de confiabilidade e de eficiência vistas anteriormente. Logo, o SGBDD é responsável por escolher, em caso de leitura, qual a cópia acessar e, em caso de atualizações, estas devem refletir-se em todas as cópias.

O segundo fator é que, se algum site (software ou hardware) ou a rede de comunicação falhar, enquanto uma atualização estiver sendo executada, o sistema tem que ter certeza de que os efeitos serão refletidos nos dados residentes no site que falhou ou nos sites inacessíveis, assim que o sistema se recuperar da falha.

O terceiro ponto é que o site não tem a informação instantânea sobre a ação que está sendo levada para os outros sites, por isso a sincronização das transações em vários sites é mais complexa que em sistemas centralizados.

A distribuição de dados, portanto, aumenta a complexidade do projeto e a implementação dos sistemas apoiados em banco de dados. Desta forma, o SGBDD [12], para alcançar as vantagens anteriormente descritas, deve ter ainda as seguintes características:

- **Rastreamento dos dados:** o catálogo do SGBDD deve manter um controle sobre os dados distribuídos, fragmentados e replicados;

- **Processamento de consultas distribuídas:** deve acessar os sites remotos, transmitindo consultas e dados para vários sites através da rede;
- **Gerenciamento das transações distribuídas:** precisa criar estratégias de execução para as consultas e transações que acessam dados de vários sites e para sincronizar os acessos aos dados distribuídos. Além disso, também deve manter a integridade de todos os bancos de dados;
- **Gerenciamento dos dados replicados:** deve decidir qual cópia dos dados replicados deve acessar e manter a consistência entre estas cópias;
- **Recuperação do banco de dados distribuído:** tem que estar apto a se recuperar de falhas ocorridas nos sites e na comunicação entre estes sites;
- **Segurança:** transações distribuídas devem ser executadas com gerenciamento da segurança dos dados, de acordo com a autorização de acesso dos usuários;
- **Gerenciamento do catálogo de informações:** o catálogo possui informações (metadados) sobre os dados no banco de dados e pode ser único para todo o BDD, ou local para cada site.

2.2. Formas de Distribuição de Dados

Em banco de dados distribuído pode-se particionar estes dados em várias unidades lógicas que são chamadas de fragmentos. Estes fragmentos, por sua vez, podem estar espalhados por vários sites.

Date [5] define que os dados em um sistema distribuído, de acordo com seu armazenamento físico, isto é, sua alocação, podem ser particionados ou replicados. Portanto, deve-se esclarecer que o particionamento é a alocação dos fragmentos de forma não-redundante, e replicação é a alocação de um mesmo fragmento em mais de um local, isto é, de forma redundante.

Todos estes temas, como fragmentação, particionamento, replicação e alocação são tratados no projeto de distribuição de dados [6][13].

2.2.1. Fragmentação de Dados

No processo de distribuição de dados, é necessário inicialmente determinar as unidades lógicas do banco de dados que serão utilizados por vários usuários e nós da

rede. Estas unidades lógicas, como vistas anteriormente, são chamadas de fragmentos.

Existem três tipos fundamentais de fragmentação [11][14][22]:

- **Fragmentação horizontal:** particiona a relação em tuplas (linhas). Assim, cada fragmento tem um subconjunto de linhas de uma relação. Há duas versões de particionamento horizontal: primária e derivada. A *fragmentação horizontal primária* de uma relação é criada a partir de predicados definidos sobre a própria relação. Já a *fragmentação horizontal derivada* é o particionamento da relação de acordo com os predicados envolvendo outras relações;
- **Fragmentação vertical:** particiona a relação em colunas, isto é, distribui a relação em sub-relações, onde cada sub-relação é definida como um subconjunto das colunas da relação original. Portanto, uma relação R produz os fragmentos $R_1, R_2, \dots, R_r$, cada um contendo um subconjunto de atributos de R como também a chave primária de R . O principal objetivo da fragmentação vertical é dividir uma relação em pequenas relações, agrupando os atributos, que são freqüentemente utilizados pelos usuários. Em formas gerais, uma fragmentação ótima é aquela que produz um esquema de fragmentação que minimiza o tempo de execução das aplicações dos usuários que acessam estes fragmentos;
- **Fragmentação mista ou híbrida:** é a junção dos dois modos de fragmentação vistos anteriormente, isto é, a combinação da fragmentação horizontal com a fragmentação vertical.

Silberschatz, Korth e Sudarshan [4] dividem uma relação mista R em um número de relações fragmentadas $R_1, R_2, \dots, R_r$. Cada fragmento é obtido a partir da aplicação ou do esquema de fragmentação horizontal ou fragmentação vertical na relação R , ou de um fragmento de R , o qual foi obtido previamente.

Os fragmentos podem ser alocados de forma *não-redundante*, onde cada fragmento é mapeado para apenas um local; ou de forma *redundante*, onde cada fragmento pode ser mapeado para um ou mais locais, dependendo do grau de replicação definido pelo projetista.

2.2.2. Replicação de Dados

A replicação de dados é utilizada para aumentar a disponibilidade de dados. O caso mais extremo está em replicar todo o banco de dados em todos os sites do sistema distribuído, criando um banco de dados distribuído completamente replicado para cada site. Isso pode melhorar a disponibilidade, porque o sistema pode continuar operando, enquanto pelo menos um site esteja funcionando. Também melhora o desempenho na recuperação de dados globais, porque o resultado de uma consulta pode ser obtida localmente de qualquer site. A desvantagem de replicar todos os dados é que se pode reduzir drasticamente a velocidade nas operações de atualização, pois uma única atualização lógica deve ser executada em todos os sites que possuem uma réplica do banco de dados, mantendo assim as cópias consistentes.

No outro extremo da replicação total está a não-replicação ou a também chamada alocação não-redundante, isto é, cada fragmento é armazenado em apenas um site. Neste caso todos os fragmentos são disjuntos, com exceção da repetição da chave primária dos fragmentos verticais ou híbridos.

Entre estes dois extremos, replicação ou particionamento, existem situações intermediárias na qual é possível considerar a replicação parcial dos dados. Ou seja, alguns fragmentos do banco de dados podem ser replicados, enquanto outros não. O número de cópias de cada fragmento pode variar até o número total de sites no sistema distribuído. Um caso especial de replicação parcial está acontecendo com mais frequência em aplicações que envolvem computação móvel, pois o banco de dados é parcialmente replicado para computadores móveis que possuem programas encarregados de sincronizá-los periodicamente com o servidor de banco de dados. A descrição dos fragmentos de replicação são chamados de esquemas de replicação.

Todo fragmento ou cada cópia de um fragmento deve estar associado a um site específico no sistema distribuído. Esse processo é chamado de distribuição dos dados ou alocação de dados. A escolha do site e o grau de replicação depende do desempenho e disponibilidade desejada para o sistema e dos tipos e frequências de transações submetidas em cada site.

Por exemplo, se é necessária alta disponibilidade, se as transações podem ser submetidas em qualquer site, e a maior parte das transações é só de leitura, então um banco de dados totalmente replicado é uma boa escolha.

Porém, se certas transações precisam ter acesso a partes específicas do banco de dados, então essas transações devem ser submetidas em um site particular, logo o conjunto de fragmentos correspondentes precisam estar alocados nesse site. Já os dados que devem ser acessados em vários locais podem ser replicados para todos esses sites. Se muitas atualizações são executadas, então é melhor limitar o uso da replicação.

3. Modularização do Banco de Dados

Conforme Ferreira e Busichia [10], a fase de projeto de um sistema é dividida em projeto funcional e projeto do banco de dados. A modularização é tratada pelo projeto funcional, de onde são gerados módulos compostos por transações, que irão manipular o banco de dados. O projeto do banco de dados considera o modelo global de dados [7], utilizando um repositório de dados comum a todos os módulos do sistema.

No entanto, quando uma aplicação é composta por vários subsistemas, a existência de um repositório comum acaba diminuindo a autonomia dos subsistemas envolvidos, pois apesar de um subsistema acessar apenas parte do banco de dados, ele possui acesso a todo o repositório de dados.

Para que os subsistemas obtenham maior autonomia no gerenciamento de dados [1][30], é necessária a modularização do banco de dados, criando um repositório próprio, apenas com os dados relevantes às transações de cada módulo gerado.

3.1. Projeto de Banco de Dados com Modularização

As duas últimas fases citadas em Ferreira e Busichia [10] foram incorporadas ao modelo original encontrados em Ceri [28][29] e em Navathe [22][27], para implementar o processo de modularização do banco de dados, conforme mostra a Figura 1.

- **Coleta e Análise de Requisitos:** essa fase consiste em se levantar e entender as necessidades dos dados e requisitos funcionais da aplicação. Dessa forma, os elementos relevantes no projeto de dados e as transações que manipulam estes dados podem ser definidos sem ambigüidade;
- **Projeto Conceitual:** é composto por duas atividades que podem ser realizadas paralelamente. A primeira é a elaboração do esquema conceitual dos dados, a partir das necessidades dos dados obtidos na fase anterior, não se levando em conta detalhes da implementação. A outra é a criação de especificações para as transações definidas, a partir dos requisitos funcionais, obtidos na fase anterior, sem se importar com o SGBD utilizado;

Figura 1 – Fases do Projeto de Base de Dados com Modularização

- **Projeto Lógico:** nesta fase, é criado o esquema lógico de dados, com base no esquema conceitual, originado na fase anterior, e de acordo com o SGBD escolhido;
- **Projeto Físico:** consiste em escolher, de acordo com o SGBD utilizado, as melhores estruturas de armazenamento e acesso para que os requisitos de desempenho dos sistemas que formam a aplicação sejam atendidas;
- **Coleta e Análise dos Requisitos de Modularização:** consiste em levantar e entender as necessidades da modularização da aplicação, definindo e associando cada subsistema às transações definidas na fase de Coleta e Análise de Requisitos, conforme as suas funções;

- **Projeto de Modularização:** é nessa fase que ocorre a modularização do banco de dados, associando os dados aos subsistemas, de acordo com as transações executadas por esses subsistemas. Essa fase utiliza o esquema conceitual global de dados, decompondo-o para gerar subesquemas correspondentes às necessidades de cada subsistema.

Maiores detalhes sobre as fases que englobam a modularização estão nas próximas seções.

3.1.1. Coleta e Análise de Requisitos de Modularização

Nessa fase, deve-se definir quais os subsistemas que compõem a aplicação e de quais subsistemas cada unidade operacional da empresa necessitará.

Assim, com base nas informações contidas no próprio domínio da aplicação e nos requisitos funcionais [25] obtidos na fase de Coleta e Análise de Requisitos, podem-se identificar os subsistemas e associar a cada um deles as transações que atendam às suas necessidades operacionais.

3.1.2. Projeto de Modularização

Objetiva identificar os módulos de dados acessados pelos subsistemas que compõem a aplicação e criar os subesquemas conceituais de cada um desses módulos, contemplando o compartilhamento e a interoperabilidade da informação. Essa fase pode ser subdividida em quatro etapas, segundo [10][11]:

- **Particionamento do esquema conceitual global de dados:** tem como objetivo decompor o esquema conceitual global de dados da aplicação construído na fase do Projeto Conceitual, de acordo com as especificações dos subsistemas e das transações globais, obtidas nas fases de Coleta e Análise de Requisitos de Modularização e Projeto Conceitual, respectivamente. Durante o particionamento, os elementos do esquema conceitual global de dados devem ser agrupados, de acordo com as necessidades operacionais de cada subsistema; e, através de uma análise cíclica entre os dados e as transações que os operam, obtêm-se os subesquemas correspondentes aos subsistemas. Para que um esquema conceitual de dados possa ser decomposto é importante que as suas

informações estejam representadas da forma mais atômica possível. No caso da utilização de mecanismos de abstração, no processo de modelagem, a atomicidade de um esquema de dados é obtida no nível mais baixo da hierarquia de abstrações, onde a informação se encontra em sua forma mais detalhada. Formalmente, a decomposição do esquema [2] pode ser representada por:

$$DER = \cup_{i=1..n} DER_i,$$

onde *DER* representa o diagrama entidade-relacionamento

- **Tratamento do compartilhamento da informação:** é nessa etapa que se trata do compartilhamento de porções do esquema conceitual global de dados, que surge devido à sua decomposição. Esse compartilhamento é caracterizado pela interseção dos subesquemas, identificados na etapa anterior, ou seja, existem elementos do esquema conceitual global de dados que pertencem a mais de um subesquema e, portanto, mais de um subsistema necessita de seus dados.

Os subesquemas que não possuem nenhum elemento em interseção, originam um módulo do banco de dados com autonomia total, em relação ao gerenciamento de seus dados, não necessitando assim de tratamento especial para seus elementos. Já os casos em que há compartilhamento, isto é, interseção entre os subesquemas, deve-se analisar separadamente cada um dos elementos que estão sobrepostos. Primeiramente, verificam-se as operações executadas, em cada elemento, por parte dos subsistemas. Essas operações podem ser classificadas em: operações de leitura, ou seja, apenas consulta-se o elemento; e operações de escrita, ou seja, criação, atualização e remoção do elemento. Em seguida, de acordo com o número de subesquemas aos quais cada elemento pertence, e com as operações executadas em cada um, atribui-se a cada elemento o seu tipo de compartilhamento. Assim, quanto ao tipo de compartilhamento, pode-se ter:

- *não-compartilhado*: o elemento pertence a um único subesquema e, portanto, um único subsistema executa operações, tanto de leitura quanto de escrita no elemento;

- *compartilhamento unidirecional*: o elemento pertence a dois ou mais subesquemas, mas apenas um subsistema executa operações de escrita no elemento, sendo que os outros executam apenas operações de leitura;
- *compartilhamento multidirecional*: o elemento pertence a dois ou mais subesquemas, e mais de um subsistema executa operações de escrita no elemento.

Formalmente, para cada subsistema S_i tem-se a utilização de um subesquema DER_i . Assim:

$$S_1 \text{ utiliza } DER_1 = \{ E_1, A_1, R_1, HG_1, AG_1, C_1 \}$$

$$S_2 \text{ utiliza } DER_2 = \{ E_2, A_2, R_2, HG_2, AG_2, C_2 \}$$

...

$$S_n \text{ utiliza } DER_n = \{ E_n, A_n, R_n, HG_n, AG_n, C_n \},$$

onde E , A , R , HG , AG e C representam respectivamente: Entidade, Atributo, Relação, Hierarquia de Generalização, Agregação e Composição.

Definem-se Gs_n como sendo os grupos de elementos dos respectivos subesquemas e operações para sintetizar cada um dos DER_i utilizados por S_n .

- **Definição dos módulos do banco de dados:** nessa etapa são definidos os módulos do banco de dados, conforme as informações de compartilhamento informadas na etapa anterior. É nessa etapa, portanto, que se geram os esquemas conceituais de cada módulo, determinando a responsabilidade pela manutenção de seus elementos. Esta manutenção está relacionada com as operações de escrita executadas no elemento, isto é, os subsistemas que efetuam operações de escrita têm a responsabilidade de fazer a manutenção. Portanto, para se definir os módulos, primeiramente, devem-se agrupar os elementos de acordo com os subsistemas que os mantêm, resultando em um grupo de elementos mantidos para cada subsistema. A seguir, fazem-se todas as combinações de interseção entre grupos Gs_n obtidos, de modo que:
 - Grupos, cuja interseção é nula, originam um módulo. Assim, se para cada combinação de interseção:

$$Gs_1 \cap Gs_2 = \emptyset \text{ e}$$

$$Gs_1 \cap Gs_3 = \emptyset \text{ e}$$

...

$G_{s_1} \cap G_{s_2} \cap \dots G_{s_n} = \emptyset$, então cria-se um módulo M_i para cada G_{s_i} .

- Grupos, cuja interseção não é nula, devem ter seus elementos unidos, isto é, os elementos que estão em mais de um grupo originam um novo módulo. Assim, se para cada combinação de interseção:

$G_{s_1} \cap G_{s_2} \neq \emptyset$, então cria-se um módulo $M_i = G_{s_1} \cup G_{s_2}$ ou

$G_{s_1} \cap G_{s_3} \neq \emptyset$, então cria-se um módulo $M_i = G_{s_1} \cup G_{s_3}$ ou

...

$G_{s_1} \cap G_{s_2} \cap \dots G_{s_n} \neq \emptyset$, então cria-se um módulo $M_i = G_{s_1} \cup G_{s_2} \cup \dots G_{s_n}$.

Quando a maioria dos elementos de uma interseção não pertence a nenhuma hierarquia de generalização, abstração de agregação ou a qualquer outra hierarquia complexa [9][18], podem-se criar até outros módulos. Por exemplo:

Se $G_{s_1} \cap G_{s_2} = G_x$, então

$G_{s_1} - G_x \Rightarrow M_1$

$G_{s_2} - G_x \Rightarrow M_2$

$G_x \Rightarrow M_3$

M_1 é acessado somente por S_1 ; M_2 é acessado somente por S_2 e M_3 é acessado por S_1 e S_2 .

- **Definição da interface dos módulos do banco de dados:** nessa etapa é definida uma interface de acesso aos dados para cada módulo do banco de dados, especificado na etapa anterior. Nesse caso, é importante a criação de procedimentos que encapsulem o acesso aos dados de cada módulo do banco de dados. Esses procedimentos devem ser criados conforme as necessidades operacionais que cada subsistema tem em relação aos dados de cada módulo. Assim, cada subsistema que utiliza dados de um determinado módulo deve fazê-lo somente através dos procedimentos daquele módulo. Pode-se classificar, ainda, os procedimentos de cada módulo como *privados* e *públicos*. Os procedimentos privados são os responsáveis pela execução de operações de escrita nos dados, sendo, portanto, disponibilizados apenas para os subsistemas responsáveis pela manutenção dos dados do módulo. Já os procedimentos públicos realizam apenas operações de leitura nos dados,

podendo ser disponibilizados para todos os subsistemas que acessam o módulo.

Assim, o esquema de dados de cada módulo deve ter a representação de seus elementos e a especificação de seus procedimentos privados e públicos.

Figura 2 – Representação de um Módulo de Dados

Para eliminar possíveis confusões existentes em projetos de bancos de dados corporativos, foram estabelecidas diretrizes, baseadas nas duas hipóteses seguintes:

- para determinar o grau de compartilhamento dos módulos dos bancos de dados é necessário uma análise cíclica entre os dados e as transações envolvidas na construção dos sistemas de informação;
- para a decomposição do esquema conceitual global do esquema de dados é necessária a utilização de um grau detalhado das abstrações de dados.

Com a adição de mais estas duas fases no processo de projeto de banco de dados e a criação dos critérios para geração dos esquemas conceituais de dados de cada módulo, é possível construir sistemas de informação para atender as necessidades específicas de cada unidade operacional de grandes corporações.

Maiores detalhes e um exemplo de modularização, através de um estudo de caso, também podem ser encontrados em [11].

4. Aspectos da Replicação de Dados

O projeto de replicação de dados envolve outros parâmetros como métodos transacionais, topologia da rede de computadores, modelos de replicação de dados e o próprio programa encarregado de gerenciar esta replicação. Estes fatores em conjunto auxiliam na elaboração de um bom projeto de replicação de dados, pois é através destes parâmetros que, entre outras características, pode-se levantar qual a necessidade dos dados, isto é, se os dados são acessados somente para leitura ou também para escrita, se as réplicas dos dados precisam estar sincronizadas ou não, se pode haver latência nas atualizações das informações e em quais sites os dados precisam residir.

Neste capítulo serão esclarecidas algumas características da replicação de dados encontrados em [3] e alguns parâmetros, como formas de transacionar os dados, e como replicar estes dados para outros sites. No capítulo seguinte, serão vistas algumas particularidades de um programa gerenciador de replicação.

4.1. Modelo de Sistema de Banco de Dados Distribuído

Um sistema de banco de dados distribuído (SBDD) é um conjunto de itens de dados distribuídos sobre vários pontos de rede e que são potencialmente usados, simultaneamente, por muitos usuários. O sistema de banco de dados administra esses dados para tornar a distribuição, as falhas e a concorrência transparentes para os usuários, assegurando que a execução de leituras e escritas passem despercebidas, como as leituras e escritas executadas em um banco de dados centralizado. A transparência com relação a todos estes fatores pode ser cara e, conseqüentemente, garantida na prática somente a extensões consistentes e com um desempenho aceitável. Será considerado aqui que o sistema de banco de dados é homogêneo e está baseado em um único modelo de dados, um único esquema e uma única linguagem de manipulação.

Cada transação consiste em um conjunto de operações ocorridas durante a execução de um programa. O SGBDD é responsável pela sincronização das

operações ocorridas em diferentes transações. Contudo, as transações necessitam das seguintes propriedades ² básicas [11][14][22]:

- **Atomicidade:** uma transação é inteiramente executada, ou todas as operações não são consideradas;
- **Consistência:** uma transação deve levar o banco de dados de um estado consistente para outro;
- **Isolamento ou independência:** a transação não pode ser afetada por atualizações de transações concorrentes. As atualizações de uma transação somente se tornam visíveis após a sua confirmação;
- **Durabilidade ou persistência:** os efeitos de uma transação confirmada não podem ser desfeitos.

A propriedade de consistência assegura que qualquer execução serial pode ser considerada correta. Uma execução é serial se, para todo par de transações, todas as operações da primeira transação são executadas antes de quaisquer operações da segunda transação. Se o estado inicial do banco de dados é consistente e se cada transação é criada para preservar a consistência de banco de dados, isto é, utilizou a propriedade de isolamento, então essa execução, que é equivalente a uma serial, não contém nenhuma transação que torna o banco de dados inconsistente.

4.1.1. Controle de Concorrência

Uma execução é serializável se produz a mesma saída e tem o mesmo efeito no banco de dados que uma execução serial possível das mesmas transações. A teoria de serializabilidade [17] formalmente define os requisitos para se conseguir execuções serializáveis. O protocolo de controle de concorrência é usado para restringir a execução de transações concorrentes, em sistemas de banco de dados centralizados e distribuídos, somente para execuções que são serializáveis.

O protocolo de controle de concorrência entra em uma das duas categorias, a pessimista ou a otimista. Protocolos pessimistas previnem inconsistências proibindo potenciais execuções não-serializáveis, e asseguram que os efeitos de uma transação

² Também conhecida como propriedades ACID.

*confirmada*³ não sejam restauradas ou anuladas. Tem-se um algoritmo pessimista, que é o bloqueio bifásico⁴ [14][17], implementado em muitos sistemas comerciais. Por outro lado, protocolos otimistas permitem que execuções não-serializáveis ocorram, detectando possíveis anomalias e abortando a transação com problema, durante uma fase de validação, que ocorre antes dos efeitos das transações serem visíveis. A certificação encontrada em [8][23] é um protocolo otimista que executa a validação através da confirmação de tempo.

4.1.2. Controle de Atomicidade

Em Helal, Heddaya e Bhargava [3] um protocolo de controle de atomicidade assegura que cada transação é atômica, isto é, ou todas as operações em uma transação são executadas, ou nenhuma é feita. A transação distribuída é tipicamente iniciada pelo site coordenador. Operações de leitura ou escrita são executadas por sites que contêm os dados envolvidos; esses sites são chamados de participantes. Um protocolo de controle de atomicidade assegura que todos os bancos de dados participantes e concordantes efetivem as operações de uma transação distribuída. Este é o princípio utilizado pelo protocolo de confirmação bifásica⁵.

4.1.3. Consistência Mútua em Bancos de Dados Replicados

Em bancos de dados replicados, é importante que a replicação seja transparente para o usuário. Em outras palavras, a execução simultânea de várias transações deve ser equivalente à execuções consecutivas das mesmas transações em um banco de dados que possui apenas uma cópia de dados. Este critério de correção para os dados replicados é uma extensão da teoria da seriabilidade [17].

O critério de correção deve implementar um mecanismo de controle de replicação que assegure que operações executadas nos dados lógicos sejam refletidas nas cópias físicas. Precisa também assegurar que o sistema replicado sempre esteja com o estado mais recente do banco de dados distribuído, isto é, se algum site ou

³ Será utilizado o termo *confirmado* para representar o termo em inglês commit que indica o término bem-sucedido de uma transação.

⁴ Do inglês: two-phase locking, também abreviado por 2PL.

⁵ Do inglês: two-phase commit, também abreviado por 2PC.

comunicação estiver com problema, seus dados devem ser atualizados, assim que forem solucionados.

4.2. Formas de Consistência da Distribuição de Dados

As formas de consistência servem para identificar o grau da necessidade das atualizações das informações para o SGBDD, que são construídos sobre características de cada método transacional e seus protocolos de comunicação, de segurança e de integridade. As principais formas de controle de consistência da distribuição de dados são mostradas a seguir:

4.2.1. Síncrona

Não há latência sincronizando os dados replicados, isto é, não existe atraso nas mudanças de dados, pois elas acontecem ao mesmo tempo, em todos os locais. A forma de consistência síncrona é usada em operações importantes e de alto risco; e, normalmente, utiliza-se do protocolo de confirmação bifásica *2PC*, para garantir a integridade das transações, em sistemas com bancos de dados distribuídos. Este protocolo garante que as transações são completamente aplicadas em todos os sites, ou nada é feito. Isto é, se houver uma falha em um site, a transação inteira falha, e as operações são desfeitas em todos os sites. O processo de se retornar aos dados originais, isto é, desfazer as operações devido a alguma falha é chamado de *rollback*. Maiores detalhes do protocolo de confirmação bifásica podem ser encontrados em [17]. Contudo, este tipo de consistência acaba sendo muito cara, principalmente quando o número de participantes da distribuição é muito grande, pois o grau de comunicação da rede é alto, e todos os participantes devem estar conectados.

4.2.2. Assíncrona

Nesta forma de consistência da distribuição de dados, as modificações nos dados não acontecem ao mesmo tempo, isto é, existe um certo período de latência na atualização dos dados de cada participante do sistema distribuído. Se uma transação falhar em um site, a transação pode se completar em outros, pois as operações serão desfeitas apenas nos sites que falharam e serão reaplicadas posteriormente. As transações são enviadas pelos sites que estão disponíveis, portanto a espera é

inevitável, se algum site estiver indisponível. Esse tipo de controle na distribuição de dados é usado quando não há necessidade de se obter o dado totalmente atualizado. Apesar de permitir uma maior disponibilidade dos dados, pode haver a necessidade de solucionar possíveis conflitos na sua sincronização.

4.3. Estratégias para Garantir a Consistência da Replicação de Dados

Várias estratégias de consistência de replicação de dados podem ser adotadas. Contudo, a escolha da melhor estratégia depende da disponibilidade de comunicação entre os nós da rede de computadores, do grau de atualização e do volume das informações requisitadas pelos usuários.

A Replicação de Dados é caracterizada como de *consistência forte*, quando o protocolo utilizado na replicação garante que todas as réplicas estão sempre sincronizadas entre si [3]. Neste caso, a sincronização entre as réplicas devem obedecer as mesmas propriedades de uma transação distribuída.

Quando estas informações precisam estar sempre atualizadas, pode-se utilizar um SGBDD que trabalha com a forma de distribuição síncrona e, portanto, implementa consistência forte, normalmente utilizando o protocolo de bloqueio bifásico *2PC* entre os nós da rede de computadores; porém, deve-se lembrar que, nesse modelo é importante que os sites e a rede de comunicação tenham um bom funcionamento.

Consistência fraca ou relaxada são termos utilizados por muitos autores quando é permitido que as réplicas de dados estejam divergentes do dado original, isto é, há um relaxamento nas regras de consistência forte. Nessas situações, os dados são atualizados assincronamente conforme o protocolo adotado.

Caso as informações não precisem estar sempre sincronizadas, isto é, permitem algum grau de latência na atualização, admite-se a replicação de forma assíncrona, utilizando, portanto, consistência fraca ou relaxada.

A seguir serão mostrados, como exemplos, alguns protocolos de replicação de dados com consistência forte, isto é, formas síncronas de replicação de dados e também formas assíncronas, algumas utilizando consistência fraca e outras com consistência relaxada [16]; e algumas formas de replicação manual, além de uma

forma automática de replicação de dados utilizada por um sistema gerenciador de replicação.

4.3.1. Consistência Forte

- **read-one-write-all (ROWA):** é um dos protocolos mais comuns, onde múltiplas cópias são mantidas atualizadas ao mesmo tempo, e os dados podem ser lidos de qualquer cópia. Assim, tem como principal vantagem a alta disponibilidade em operações de leitura, porém necessita de redes de comunicação confiáveis, pois, se alguma partição da rede estiver com problema, nenhuma cópia será atualizada. Uma das variações do ROWA é o **primary copy ROWA**, onde uma cópia é eleita primária, e todas as operações de leitura são submetidas a ela, garantindo assim que a leitura seja a mais atual. As operações de escrita são submetidas primeiramente à cópia primária e posteriormente às demais cópias. A operação somente é tida com sucesso após atualizar todas as cópias. Caso a cópia primária falhe, uma outra cópia assume seu papel.
- **Quorum consensus (QC) ou votação:** nesse protocolo, as operações de escrita podem acontecer em apenas um subconjunto dos sites acessíveis, chamados de quorum de escrita. As operações de leitura também podem acontecer em um subconjunto de sites, chamados de quorum de leitura, desde que, o quorum de leitura, possua uma interseção não-vazia com o quorum de escrita. É essa interseção que garante que as leituras realizadas são obtidas dos dados mais recentemente atualizados, pois estas réplicas divididas em quoruns entram em consenso para responder às operações dirigidas a esse quorum. Diz-se que um site votou quando ele participou com sucesso em uma operação, por isso, o nome votação. As principais vantagens desse protocolo são a capacidade de reconfiguração dinâmica dos quoruns, a tolerância a falhas na comunicação e principalmente o fato de não haver a necessidade de retomar as operações, após uma eventual partição da rede ter sido corrigida. Porém, para que a validação das operações seja realizada, é necessária muita comunicação entre os sites participantes dos quoruns, o que pode ser desvantajoso.

- **híbrido ROWA / QC:** quando a rede é confiável, o protocolo de votação acaba sendo caro demais, devido ao alto volume na comunicação entre os sites participantes ao validar uma operação. Já o protocolo ROWA, apesar de mais eficiente na resposta às operações, precisa de redes mais confiáveis. Por isso, esse modelo propõe a utilização do protocolo ROWA, nos períodos em que a rede está adequada, e o protocolo de votação, nos momentos em que existam falhas nos sites ou na comunicação. Assim, uma transação trabalha sobre o método ROWA, em modo normal, e chaveia para modo de falha, quando percebe algum problema na cópia primária. Ao mudar para o modo de falha, o protocolo utilizado passa a ser o protocolo de votação. Nesse momento, as réplicas possuem uma lista com as atualizações perdidas, bem como suas respectivas versões. Logo que a falha é corrigida, as operações são retomadas a partir desta lista, que por sua vez é eliminada, voltando ao modo normal. No modo de falha, a lista de atualizações perdidas é mantida por todas as réplicas, pois, caso o coordenador falhe, outro coordenador é eleito para dar continuidade às operações.

4.3.2. Consistência Fraca

- **sincronização por marcas de tempo⁶ (timestamped anti-entropy - TSAE):** o TSAE [20] é um dos protocolos de consistência fraca mais conhecidos, pois, apesar de prover consistência eventual, ele garante que todas as réplicas recebam ordenadamente todas as operações em um tempo finito, mas indeterminado [19]. Isso é possível devido ao armazenamento das operações em um registro de mensagens. O TSAE mantém dois vetores de marcas de tempo, o primeiro guarda quais mensagens foram recebidas, e o segundo acompanha os *reconhecimentos*⁷ dos outros processos. Esses vetores mantêm, nas réplicas, uma marca de tempo para cada operação e, de tempos em tempos, uma réplica escolhe outra para dar início a uma sessão de antientropia. Define-se entropia como o grau de desordem em que o sistema se encontra. Neste caso, na sessão de antientropia há uma troca dos vetores de

⁶ Do inglês: timestamp.

⁷ Do inglês: acknowledgments.

mensagens e de reconhecimentos. Assim, cada réplica, através dos marcadores de tempo, contidos nesses vetores, detecta se recebeu mensagens que a outra réplica não recebeu, apenas verificando se a marca de tempo que possui é maior que a marca de tempo da outra réplica. Caso as mensagens não tenham sido recebidas, elas são repassadas para a outra réplica e vice-versa. Ao encerrar a sessão de antientropia, todas as réplicas envolvidas encontram-se com o mesmo conjunto de mensagens. Nesse momento, o vetor de marcas de tempo é ajustado para refletir a atualização e o vetor de reconhecimento é implicitamente atualizado. Porém, se algum dos passos falhar, antes do término da sessão de antientropia, a sessão é cancelada e todas as mudanças são descartadas.

4.3.3. Consistência Relaxada

- **quasi-copies [21]:** nesse protocolo é possível especificar alguns limites de divergência entre as réplicas, o que é garantido devido às operações de escritas serem realizadas apenas na cópia primária. É utilizada uma abordagem *mestre-escravo*⁸ que provê uma especificação de intervalos de inconsistência, pois somente quando há o envio explícito do estado da réplica-mestre (cópia primária) para as réplicas-escravas é que suas informações são atualizadas. O usuário é quem determina uma constante ou um período de tempo em que o valor da réplica pode se diferenciar do valor da cópia primária (período de latência). Este protocolo é muito semelhante ao protocolo de cópia primária, porém a propagação de suas informações é assíncrona e explícita, conforme o critério estipulado pelo usuário.
- **cópia primária virtual:** difere no modelo padrão de cópia primária, por introduzir uma abstração de cópia primária virtual [26], que é o termo utilizado para o conjunto de réplicas que estão consistentes com o estado atual do dado, que, por sua vez, se encontra em um grupo de réplicas chamado de ilha de consistência. Esse modelo não permite a existência de ilhas de consistências vazias. As réplicas podem entrar ou sair da ilha de consistência, de acordo com os requisitos da aplicação, e, assim que uma

⁸ Do inglês: master-slave.

réplica entra na ilha, seus dados são atualizados e as demais réplicas da ilha são informadas do novo membro, que participará das sincronizações das réplicas a partir desse momento. As operações realizadas na ilha de consistência seguem o protocolo *ROWA/QC*, visto anteriormente, possibilitando leituras locais em qualquer membro da ilha, enquanto a ilha estiver livre de falhas. Caso ocorra falha em algum membro da ilha, o algoritmo chaveia para o protocolo de votação. As operações de escrita são realizadas em qualquer membro da ilha, e o protocolo se encarrega de sincronizar os demais membros com a cópia primária. Réplicas inconsistentes podem ser acessadas por aplicativos que também requisitam escritas, porém estas escritas, chamadas de *escritas cegas*⁹, são redirecionadas para a ilha de consistência, onde a operação é realizada como se tivesse sido originada na própria ilha. Caso as operações sejam bem-sucedidas na ilha, elas são propagadas para as réplicas que não pertencem à ilha (réplicas de consistência fraca), conforme a especificação de consistência nessas réplicas. Dessa forma, a ilha de consistência funciona como uma cópia primária das réplicas de consistência fraca. Esse protocolo faz parte de um projeto chamado ASPECT [26], que é responsável pela especificação de consistência orientada à aplicação, que permite reduzir o número de réplicas síncronas e também permite especificar dinamicamente os requisitos de consistência entre as réplicas. Esta flexibilidade é de fundamental importância, quando o conjunto de aplicações e de clientes que acessam um dado replicado variam dinamicamente.

4.3.4. Manual

- **cópia, transferência e carga do banco de dados:** é feita uma cópia de todo o BD para um arquivo que será transferido para os sites replicados somente para leitura. O site replicado não pode ser acessado até que todo o arquivo seja lido, fazendo, portanto, um bloqueio em todo o BD;

⁹ Do inglês: blind writes.

- **cópia, transferência e carga do *log*¹⁰ de dados:** é feita uma cópia do *log* para um arquivo que será transferido para os sites replicados somente para leitura. O site replicado não pode ser acessado até que todo o arquivo seja lido. Este método também bloqueia o BD;
- **cópia completa (*bulk copy*):** os dados de cada tabela são copiados para arquivos, isto é, essa cópia gera um arquivo para cada tabela, o que pode acarretar a perda de integridade dos dados, se não houver bloqueio nas tabelas;
- **imagem (*Snapshot*):** uma imagem de cada tabela é enviada diretamente para os sites replicados, não garantindo também a integridade dos dados, devido ao bloqueio da tabela no momento da imagem;
- **marcação de linha (*row flagging*) e transferência:** nesse método há uma tabela que armazena os dados alterados, e somente esses dados são enviados aos sites replicados. Esse método tem, portanto, como foco principal a linha, e não a transação.

4.3.5. Com um Gerenciador de Replicação

Com a utilização de um gerenciador de replicação, todos os procedimentos de atualização das réplicas tornam-se automáticas, sem a necessidade, na maioria das vezes, da intervenção dos usuários, como nos métodos manuais. No próximo capítulo, serão mostradas as principais características do Replication Server da Sybase, responsável pelo gerenciamento da replicação de dados.

¹⁰ Log pode ser traduzido como registro ou diário, mas é tão usado que será mantido sem tradução.

5. Ferramenta Gerenciadora para Controle Assíncrono de Replicação de Dados

Como exemplo de uma ferramenta gerenciadora de replicação, será utilizado o Replication Server Sybase para ilustração dos vários conceitos e estratégias. Este produto utiliza um método contínuo e de transferência assíncrona de replicação dos registros do *log*. Como todas as transações são armazenadas no *log*, o Replication Server captura estas transações sem bloquear a tabela. As transações são enviadas automaticamente para os sites replicados. Este método tem como foco a transação, que é aplicada consistentemente em todos os bancos de dados com uma pequena latência.

5.1. Componentes

Aplicação do cliente: representa a aplicação instalada no cliente;

Servidor de BD (PDS/RDS): é o responsável pelo gerenciamento dos bancos de dados. Podendo ser o Servidor de Banco de Dados Primário (PDS) ou Servidor de Banco de Dados Replicado (RDS);

Banco de Dados (PDB/RDB): o banco de dados primário (PDB) é responsável pelos dados primários, e o banco de dados replicado (RDB) é responsável pelos dados replicados. Porém, de acordo com a estratégia de replicação utilizada, o banco de dados pode ser tanto primário quanto replicado;

Servidor de Replicação (PRS/RRS): o Servidor de Replicação Primário (PRS) é o responsável por receber as transações do banco de dados primário e enviar a outros Servidores de Replicação. Envia também informações sobre a replicação e subscrição.

O Servidor de Replicação Replicado (RRS) é responsável por receber as transações do Servidor de Replicação Primário e aplicá-los no banco de dados replicado. Pode ser, ao mesmo tempo, primário e replicado, de acordo com a estratégia utilizada;

Agente de Replicação (Repl Agent/LTM): é responsável por procurar, no *log* do banco primário, as transações que devem ser replicadas e enviá-las ao Servidor de Replicação Primário. Também é responsável por coordenar a recuperação entre o *log* de transação e o Servidor de Replicação;

Stable Device: é um meio de armazenamento não-volátil usado e gerenciado pelo Servidor de Replicação. É composto por *stable queues*, que é a estrutura de dados responsável por guardar os dados a serem replicados até que eles sejam confirmados;

Sistema de Banco de Dados do Servidor de Replicação (RSSD): é responsável por armazenar o catálogo de sistema do RS, incluindo a definição das replicações, subscrições, informações sobre rota, *log* de erro, recuperação e transações rejeitadas.

5.2. Arquitetura

Figura 3 – Arquitetura de Replicação do Replication Server Sybase

A princípio, é preciso identificar quais dados serão replicados; nesse momento, cria-se a publicação dos dados, chamada de “Replication Definition”. Após a publicação, é necessário estipular quem poderá/irá receber estes dados, isto é, quem irá subscrever/requisitar os dados publicados anteriormente. Esta definição é chamada de “Subscription”.

Essas informações sobre publicação e subscrição são armazenadas no próprio sistema de banco de dados do servidor de replicação (RSSD), e que, por sua vez, também são replicados para os outros Servidores de Replicação.

A replicação é iniciada quando uma transação, no banco de dados primário, é efetivada, pois o agente de replicação recebe do banco de dados primário o *log* das

transações gravadas e que possuem publicação. O agente de replicação, por sua vez, envia essas transações para o Servidor de Replicação Primário (PRS), que é responsável por gerenciar e truncar o *log* do banco de dados primário.

Nesse momento, o Servidor de Replicação Primário mantém as transações na Stable Device (item 5.1). Utilizando-se das informações contidas no RSSD, o PRS decide o que fazer com a transação, isto é, se a transação é descartada, por não haver subscrição, se ela é escrita no banco de dados replicado, que pode estar na mesma máquina do banco de dados primário - logo, o próprio PRS se encarrega de atualizar o banco de dados replicado - ou se a distribui para o Servidor de Replicação Replicado (RRS), que está em outro site.

O Servidor de Replicação Replicado, ao receber as informações do PRS, mantém as transações na sua estrutura de filas (Stable Device) e atualiza o banco de dados replicado.

5.3. Exemplos de Estratégias de Consistência Assíncrona de Replicação de Dados

Para representar algumas das estratégias de replicação [31][32][33] de dados do gerenciador de replicação ilustrado, será utilizada como forma demonstrativa, a distribuição dos dados nas cidades de São Paulo, Piracicaba, Ribeirão Preto e São Carlos. Num modelo real, verifica-se que este banco de dados pode ser corporativo, portanto pode estar sendo utilizado por vários sistemas.

5.3.1. Sites Replicados somente para leitura

Figura 4 – Sites Replicados somente para leitura

Sites Replicados

- Os sites replicados apenas necessitam de acesso de leitura para os dados.
- Atualizações são feitas no site primário e então propagadas para os sites replicados.

- Os sites replicados utilizam cópias replicadas, em modo apenas de leitura.

Atualização dos Dados Primários

- Somente um pedido do cliente local pode atualizar o site primário.

5.3.2. Atualização remota do Site Primário por conexões do cliente

Figura 5 – Atualização remota do Site Primário por conexões do cliente

Sites Replicados

- Os sites replicados desejam modificar os dados.
- Atualizações são feitas no site primário e então propagadas para os sites replicados.

- Os sites replicados utilizam cópias replicadas, em modo apenas de leitura.

Atualização dos Dados Primários

- Os sites replicados se conectam remotamente ou usam uma conexão direta do cliente, através de requisições diretas ou RPCs¹¹, para fazer mudanças no dado primário.

Nesse exemplo verifica-se que:

- Piracicaba e São Carlos são somente de leitura.
- Ribeirão Preto tem uma conexão direta ao banco primário em São Paulo.

¹¹ RPC (Remote Procedure Call) é a chamada a procedimentos remotos.

5.3.3. Site Remoto requisita atualização do Site Primário

Figura 6 – Site Remoto requisita atualização do Site Primário

Sites Replicados

- Os sites replicados desejam modificar os dados.
- Atualizações são feitas no site primário e então propagadas para os sites replicados.

- Os sites replicados utilizam cópias replicadas, em modo apenas de leitura.

Atualização dos Dados Primários

- Os sites replicados requisitam mudanças no site primário e esperam até que as modificações sejam distribuídas para as cópias replicadas.

Nesse exemplo:

- Piracicaba, Ribeirão Preto e São Carlos podem ler os dados replicados ou podem pedir alterações nos dados primários (via seus bancos de replicação, através da chamada de procedimentos replicados).

5.3.4. Site Remoto requisita atualização do Site Primário e mantém atualização temporária no local

Figura 7 – Site Remoto requisita atualização do Site Primário e mantém atualização temporária no local

Sites Replicados

- Os sites replicados desejam modificar seus dados locais.
- Atualizações são feitas no site primário e então propagadas para os sites replicados.

- Os sites replicados utilizam cópias replicadas, em modo apenas de leitura, mas modificam uma cópia local dos seus dados replicados.

Atualização dos Dados Primários

- Os sites remotos (replicados) modificam uma cópia local dos seus dados e requisitam mudanças, no site primário, para trazer tudo atualizado.

- Quando a atualização chega ao site replicado, os dados replicados são alterados, e os dados redundantes da cópia local são eliminados.

Nesse exemplo:

- Ribeirão Preto não é mostrado.
- São Carlos mantém uma cópia local dos seus dados - usa a cópia local até que os dados cheguem de São Paulo.
- São Carlos não está restrito ao primário, mas ainda espera sua atualização.

5.3.5. Fragmentos Primários Distribuídos

Figura 8 – Fragmentos Primários Distribuídos

Dado Primário Particionado

- Os dados primários são fragmentados em vários bancos de dados do sistema.
- Somente uma imagem de um subconjunto de dados primários existe, mas não em uma única tabela num servidor.

Sites Remotos

- Os sites remotos controlam seus próprios dados primários e podem modificá-los (como se fossem sites primários sobre seus dados).
- Os sites remotos também recebem cópias replicadas de dados dos outros sites.
- Atualizações feitas nos dados primários são propagadas para todos os outros sites.
- Os sites remotos podem receber todos os dados primários dos outros sites ou parte deles.

Nesse exemplo:

- São Paulo controla seus dados e tem, em modo de leitura, os dados replicados de Piracicaba, Ribeirão Preto e São Carlos.
- Quando São Paulo atualiza seus dados, essa atualização é enviada para Piracicaba, Ribeirão Preto e São Carlos.

5.3.6. Site Corporativo recebe dados dos outros Sites (Corporate Roll-Up)

Figura 9 – Site Corporativo recebe dados dos outros Sites (Corporate Roll-Up)

Dado Primário Particionado

- Os dados primários são fragmentados em vários bancos de dados do sistema.
- Somente uma imagem de um subconjunto de dados primários existe, mas não em uma única tabela num servidor.

Sites Remotos

- Os sites remotos controlam seus próprios dados primários e podem modificá-los (como se fossem sites primários sobre seus dados).
- Os sites remotos não podem ver o restante do banco de dados, isto é, só possuem acesso aos seus dados.
- Atualizações feitas nos dados primários são propagadas somente para o site corporativo.

Site Corporativo

- É o site primário de seus dados, e recebe cópia dos dados dos sites remotos.

- Os sites remotos podem receber todos os dados primários dos outros sites ou parte deles.

Nesse exemplo:

- São Paulo controla seus dados e tem, em modo de leitura, os dados replicados de Piracicaba, Ribeirão Preto e São Carlos.

Quando Piracicaba, Ribeirão Preto e São Carlos atualizam seus dados, essa atualização é enviada para São Paulo.

5.3.7. Sites Corporativos enviando Dados para Sites Replicados

Figura 10 – Sites Corporativos enviando Dados para Sites Replicados

Idem ao modelo anterior.

Sites Replicados

- Recebem dados replicados do site corporativo.
- Podem ou não ter seus próprios dados locais.

Nesse exemplo:

- Quando São Paulo atualiza seus dados, esses dados são atualizados em Bauru e Pirassununga.
- São Paulo envia as atualizações de Piracicaba, Ribeirão Preto e São Carlos para Bauru e Pirassununga.

5.3.8. Sites Primários Redundantes – esperando queda do primário (Warm Standby)

Figura 11 – Sites Primários Redundantes – esperando queda do primário
Primários Redundantes

- Duas versões completas do banco de dados primário existem, um principal ativo e outro para reserva.
- Somente um deles atua como primário, num dado momento.
- Um banco de dados primário propaga mudanças para o outro quando ele envia as mudanças para os sites replicados (linha contínua).
- O primeiro banco de dados primário está ativo para receber atualizações do backup do banco de dados primário, se eles estão chaveados (linha tracejada).
- Somente um primário, o que estiver ativo, pode ser atualizado por vez.

Sites Replicados

- Os sites remotos (replicados) são somente de leitura.
- Os sites remotos (replicados) estão aptos a receberem dados de outra possível origem primária (linha tracejada), mas atualmente recebem os dados do primário ativo (linha sólida).
- Sites remotos podem submeter pedidos ao primário para ativar o banco de dados primário.

5.3.9. Sites Primários Redundantes – trocando os primários (Switching Over)

Figura 12 – Sites Primários Redundantes – trocando os primários

Trocando os Primários

- Quando o primeiro servidor primário está indisponível, a aplicação que atualiza a troca de primário ativa o backup do site primário.
- A aplicação deve atualizar somente um primário por vez.
- O backup do site primário propaga mudanças para os sites replicados (agora linha contínua)
- Os sites replicados estão aptos a receberem atualizações de outro site primário, mas atualmente recebe os dados do primário ativo.
- O backup do site primário também propaga mudanças para o primeiro site primário (linha sólida).

5.3.10. Ponto a Ponto (Peer-to-Peer)

Figura 13 – Ponto a Ponto (Peer-to-Peer)

Neste caso, não há site primário e replicado, pois a estratégia permite a atualização em qualquer lugar.

Não existe restrição na atualização, porém os conflitos devem ser resolvidos depois das transações terem sido *confirmadas*.

Tem como principal vantagem que qualquer site pode atualizar todos os dados localmente, sem se importar com a “propriedade” do dado.

Mas, em contrapartida, esse tipo de estratégia deve tolerar inconsistências nos dados globais.

Mesmo com o controle de *versão para as linhas*¹², que será visto adiante, são necessários outros tipos de resolução de conflitos, conforme a seção 5.4.

¹² Do inglês: row-version.

5.3.11. Site Primário Dinâmico (Dynamic Site Ownership)

Figura 14 – Site Primário Dinâmico (Dynamic Site Ownership)

Esta estratégia também possibilita que qualquer site altere os dados, porém de uma forma mais organizada.

É necessário obter a “propriedade” do registro para se fazer a alteração.

Assim, quando se deseja alterar algum dado, é chamado um procedimento para alterar essa propriedade no site primário, que por sua vez replica essa informação, passando a propriedade do dado para o site requisitante (tornando-se primário). Desta forma, o dado pode ser alterado e automaticamente replicado para os demais sites.

Tem como principal vantagem que qualquer site pode atualizar todos os dados localmente, mantendo sua consistência global.

Mas, em contrapartida, pode haver uma degradação no desempenho, pois o usuário fica esperando a mudança de “propriedade” ser propagada para que ele possa alterar os dados.

5.4. Conflitos na Replicação de Dados

A estratégia de *versão para as linhas* consiste em acrescentar, na tabela, uma nova coluna, que guardará a versão da linha. Assim, ao se fazer a atualização de uma linha, é verificada a versão da linha corrente. Desta forma, pode-se determinar se há ou não conflito na sua atualização.

Figura 15 – Conflitos na Replicação de Dados

Neste exemplo, pode-se verificar que o Site A e o Site B fizeram simultaneamente alterações na nota do aluno José, que estava antes da alteração com versão de linha igual a 1. Cada site, ao alterar sua informação, acrescentou 1 na coluna de versionamento de linha, porém os registros, ao serem replicados, causaram um conflito, pois o site, ao receber um novo registro, verifica se o registro que está recebendo é uma versão subsequente à versão do registro existente. Nesse caso, a versão recebida é a mesma que a versão do registro existente, caracterizando, assim, o conflito na atualização dos registros.

Alguns cenários para resolução de conflitos:

- **Prioridade de Site (Site Priority):** cada site possui sua prioridade. O site que possuir maior prioridade terá efetivada a sua transação. Essa resolução pode ser implementada através de *trigger* ou *stored procedures* que, por exemplo, devem consultar uma tabela que armazene um critério de prioridade entre os sites existentes;
- **Tempo Global (Global Time):** é uma resolução um pouco mais complexa, pois pressupõe que todos os sites mantêm em sincronia os seus relógios. Essa resolução também é implementada usando-se *trigger* ou *stored procedures*;

- **Atualização Comutativa (Commutative Update):** a base de sua estrutura é ignorar o conflito. Esse tipo de resolução só é viável em algumas situações onde o valor correto do dado é obtido ignorando o conflito das alterações. Essa situação prevê a mesma prioridade para todos os sites;
- **Manual:** o conflito não é automaticamente solucionado. O conflito é gravado no *log* e resolvido manualmente. Dessa forma, os dados podem ficar inconsistentes por um longo período de tempo.

6. Modularização e Replicação de Dados

Ao analisar as necessidades e os benefícios da distribuição de dados (capítulo 2), em conjunto com o projeto de modularização (capítulo 3) e os aspectos da replicação de dados (capítulo 4), pode-se perceber uma grande ligação entre o projeto de distribuição e o projeto de modularização. Assim, a proposta deste trabalho é de unificar, em um único projeto, a modularização e a distribuição do banco de dados, através da replicação dos dados.

Dessa forma, após a modularização, inicia-se uma nova análise, sempre considerando-se as necessidades que cada subsistema tem de realizar operações de leituras e escritas dos dados, levando-se em conta o aspecto da distribuição e, principalmente, trazendo todos os benefícios da distribuição, para um ambiente controlado, de fácil manutenção e reutilização, que é o ambiente obtido com a modularização de dados.

6.1. Adaptação do Projeto de Replicação nas Fases da Modularização

Através de algumas adaptações no modelo proposto por Ferreira e Busichia [10], pode-se, além de modularizar o banco de dados, conforme a necessidade dos subsistemas, fazer a melhor escolha de como distribuir um módulo ou parte dele em um ambiente complexo, como o ambiente de sistemas corporativos.

Essa adaptação consiste na adição de mais uma fase no processo de projeto de bases de dados, chamada de *Coleta e Análise de Tipos de Replicação*, e algumas mudanças na fase de *Projeto de Modularização*, vista no capítulo 3, visando também, o propósito da replicação dos dados. Por isso, essa fase adaptada está sendo chamada de fase de *Projeto de Modularização e Replicação de Dados*, conforme mostra a Figura 16.

Figura 16 – Fases do Projeto de Base de Dados com Modularização e Replicação de Dados

6.1.1. Coleta e Análise de Tipos de Replicação

Durante a fase de Coleta e Análise de Tipos de Replicação, deve-se estudar e definir, dentre as estratégias de replicação existentes, quais as que melhor atendem às necessidades, considerando aspectos como a utilização ou não de um gerenciador automático de replicação e as estratégias de consistência de que o ambiente replicado necessita (item 4.3).

No caso da utilização de um gerenciador de replicação, é necessário saber quais estratégias ele gerencia, e, se não for utilizado um gerenciador de replicação, é preciso levantar quais as estratégias de replicação devem ser implementadas.

Estas definições também devem considerar alguns aspectos descritos no capítulo 4, como a necessidade ou não da consistência dos dados replicados, isto é, se

os dados replicados precisam estar sempre atualizados com os dados primários, deve-se escolher estratégias de consistência forte; caso contrário, estratégias de consistência fraca ou relaxada devem ser escolhidas para serem utilizadas na distribuição dos módulos do modelo corporativo.

Portanto, é nessa fase que deve-se levantar de que formas os dados podem ser replicados, suas estratégias e como é sua implantação, isto é, manual ou automática. Essas definições servirão de base para a próxima fase, pois a definição da forma como replicar algum módulo, ou parte dele, estará sendo atendida por algumas das regras estabelecidas aqui.

Por exemplo, analisando, superficialmente, uma rede de lojas de departamento que utilizam bancos de dados descentralizados, podem-se levantar algumas necessidades, como a da atualização, em todas as filiais da empresa, dos itens vendidos em uma lista de casamento e a de informe, das filiais para a matriz, do total vendido no dia.

No primeiro caso, a atualização precisa ocorrer em todas as filiais, para que não ocorra, por exemplo, a venda do mesmo produto da lista mais de uma vez. Portanto, nesta situação, é importante a utilização de uma estratégia de consistência forte como o *ROWA* (item 4.3.1). Já no segundo caso, onde a necessidade de atualização é diária, cada filial pode armazenar suas vendas localmente e, no final do dia, atualizar a base de dados central, que pode tomar algumas decisões a partir desta informação, como despacho de mercadoria para as filiais, controle de estoque, etc. Assim, esse tipo de necessidade poderia utilizar a estratégia de consistência fraca (item 4.3.2), como a sincronização por marcas de tempo ou até mesmo alguma estratégia manual (item 4.3.4) de atualização do banco de dados da matriz.

Cabe nessa fase, portanto, fazer o levantamento das estratégias que podem ser úteis para a replicação dos dados conforme as necessidades do negócio e que, por sua vez, tenham as suas funcionalidades implementadas ou por um gerenciador de replicação ou por outros procedimentos.

Nesse caso, todas as estratégias que não estejam aptas a serem implementadas por quaisquer razões, como restrições de tecnologia, conhecimento ou forma como as máquinas estão fisicamente distribuídas, não serão escolhidas, bem como aquelas estratégias que, de acordo com o contexto do negócio e distribuição, sejam

consideradas desnecessárias, obsoletas ou até mesmo muito complexas para o ambiente do sistema corporativo analisado.

É claro que, para realizar essa fase é necessário, por parte de seu responsável, um bom conhecimento das estratégias de replicação existentes, quais estão disponíveis em seu ambiente e principalmente as necessidades do sistema corporativo.

6.1.2. Projeto de Modularização e Replicação de Dados

O projeto de modularização, inicialmente proposto por Ferreira e Busichia [10], conforme visto no capítulo 3, é subdividido em quatro etapas. O acréscimo de mais uma etapa, chamada de Definição da Replicação de Dados nos Módulos, é a adaptação realizada nesta fase.

- **Definição da Replicação de Dados nos Módulos:** esta nova etapa tem o objetivo de analisar, a partir dos módulos gerados pela terceira etapa *Definição dos módulos da base de dados* (item 3.1.2), quais as tabelas, dentro de cada módulo, são realmente alvo de replicação; e qual estratégia, dentre as estratégias definidas na fase de Coleta e Análise de Tipos de Replicação (item 6.1.1), é a mais adequada para ser utilizada.

Essa nova etapa foi inserida neste ponto, pois, dependendo da estratégia escolhida para a replicação dos dados, será necessário maior controle por parte das interfaces. Essas interfaces são geradas na próxima etapa, garantindo assim, a correta implementação das estratégias aqui escolhidas.

Após esta etapa, então, com base nos módulos gerados e na definição de qual estratégia de replicação adotar para as tabelas dos módulos, inicia-se a quinta etapa, *Definição da interface dos módulos da base de dados* (item 3.1.2), que no projeto original constava como quarta etapa.

Através de um estudo de caso resumido, obtido do modelo de dados dos sistemas corporativos da USP, é possível ilustrar como fazer a adaptação do modelo descrito no capítulo 3 com as novas etapas:

O exemplo do projeto de modularização e replicação de dados foi simplificado, bem como as especificações das transações foram omitidas para exemplificar a modularização e replicação da base de dados. O exemplo em questão é o controle de matrícula dos alunos, o gerenciamento de processos e o controle de pessoal da universidade, que, por sua vez, pode ser acessado por quatro subsistemas. O primeiro subsistema S_1 é responsável pela gerência dos processos abertos na USP e seus andamentos. Entende-se “processo” como todos os procedimentos que requerem documentação e trâmite dos documentos. O subsistema S_2 controla os dados de graduação como: cursos, disciplinas, turmas, dados dos alunos, suas matrículas e carteiras estudantis. O gerenciamento das matrículas e dados dos alunos de pós-graduação, bem como a emissão de suas carteiras são realizados pelo subsistema S_3 . E o subsistema S_4 se encarrega de gerenciar os dados de todas as pessoas que mantêm vínculo com a universidade, como servidores, professores e alunos, além da emissão de carteiras para professores e funcionários.

Com base nos subsistemas e no modelo conceitual global, aplicaram-se as etapas do Projeto de Modularização e Replicação de Dados:

Através da primeira etapa, que é o “Particionamento do esquema conceitual global de dados”, obtêm-se os subesquemas que correspondem aos acessos realizados aos dados, pelas transações dos subsistemas existentes. Esses subesquemas podem ser vistos na Figura 17, representado pelo modelo relacional simplificado.

Elemento	Pertence ao(s) subesquema(s)				Tipo de Operação L-leitura / E-escrita				Tipo de Compartilhamento
	E_1	E_2	E_3	E_4	S_1	S_2	S_3	S_4	
Acompanhamento	x				L/E				não compartilhado
Aguardo	x				L/E				não compartilhado
Andamento	x				L/E				não compartilhado
Assunto	x				L/E				não compartilhado
Juntada	x				L/E				não compartilhado
Processo	x		x		L/E		L		unidirecional
Requisicaoproc	x				L/E				não compartilhado
Cursogr		x				L/E			não compartilhado
Disciplinagr		x				L/E			não compartilhado
Habilitacaogr		x				L/E			não compartilhado
Habilproggr		x				L/E			não compartilhado
Habilturma		x				L/E			não compartilhado
Histescolargr		x				L/E			não compartilhado
Histproggr		x				L/E			não compartilhado
Programagr		x				L/E			não compartilhado
Turmagr		x				L/E			não compartilhado
Agprograma			x				L/E		não compartilhado
Area			x				L/E		não compartilhado
Credarea			x				L/E		não compartilhado
Diplomaposgrad			x				L/E		não compartilhado
Disciplina			x				L/E		não compartilhado
Idioma			x				L/E		não compartilhado
Oferecimento			x				L/E		não compartilhado
R39pgmoridoc			x				L/E		não compartilhado
R41pgmmattur			x				L/E		não compartilhado
R46pgmproidi			x				L/E		não compartilhado
R48pgmtrbdoc			x				L/E		não compartilhado
Pessoa	x	x	x	x	L	L/E	L/E	L/E	multidirecional
Alunogr		x				L/E		L/E	multidirecional
Alunopos			x				L/E	L/E	multidirecional
Docente			x	x			L/E	L/E	multidirecional
Servidor				x				L/E	não compartilhado
Cartoseas		x	x	x		L/E	L/E	L/E	multidirecional
Endpessoa		x	x	x		L/E	L/E	L/E	multidirecional
Telefpessoa		x	x	x		L/E	L/E	L/E	multidirecional
Tipovinculo		x	x	x		L	L	L/E	unidirecional
Vinculopessoa		x	x	x		L	L	L/E	unidirecional

Tabela 1 – Tipo de compartilhamento de cada elemento do esquema conceitual global de dados

Após a elaboração da tabela anterior, inicia-se a terceira etapa, que é a “Definição dos módulos da base de dados”. O primeiro passo é agrupar os elementos de acordo com os subsistemas mantenedores dos dados, isto é, aqueles que efetuam operações de escritas, obtendo-se os seguintes grupos:

$$Gs_1 = \{\text{Acompanhamento, Aguardo, Andamento, Assunto, Juntada, Processo, Requisicaoproc}\}$$

$$Gs_2 = \{\text{Cursogr, Disciplinagr, Habilitacaogr, Habilproggr, Habilturma, Histescolargr, Histproggr, Programagr, Turmagr, Alunogr, Cartcoseas, Endpessoa, Pessoa, Telefpessoa}\}$$

$$Gs_3 = \{\text{Agprograma, Area, Credarea, Diplomaposgrad, Disciplina, Idioma, Oferecimento, R39pgmorigdoc, R41pgmmattur, R46pgmproidi, R48pgmtrbdoc, Alunopos, Cartcoseas, Docente, Endpessoa, Pessoa, Telefpessoa}\}$$

$$Gs_4 = \{\text{Alunogr, Alunopos, Cartcoseas, Docente, Endpessoa, Pessoa, Servidor, Telefpessoa, Tipovinculo, Vinculopessoa}\}$$

A seguir, é feita a interseção entre os grupos, afim de obter-se aqueles que podem gerar módulos diretamente. Assim:

$$Gs_1 \cap Gs_2 = \emptyset$$

$$Gs_1 \cap Gs_3 = \emptyset$$

$$Gs_1 \cap Gs_4 = \emptyset$$

$$Gs_2 \cap Gs_3 = \{\text{Cartcoseas, Endpessoa, Pessoa, Telefpessoa}\}$$

$$Gs_2 \cap Gs_4 = \{\text{Cartcoseas, Endpessoa, Pessoa, Telefpessoa}\}$$

$$Gs_3 \cap Gs_4 = \{\text{Alunopos, Cartcoseas, Docente, Endpessoa, Pessoa, Telefpessoa}\}$$

Nesse caso, Gs_1 tem interseção vazia com os demais grupos, por isso ele origina diretamente o módulo M_1 . Já os grupos Gs_2 , Gs_3 e Gs_4 , por terem interseção entre si, e devido aos elementos que se interseccionam pertencerem a uma hierarquia de generalização, unem-se seus elementos, originando o módulo M_4 . Os elementos do grupo Gs_2 que não possuem interseção com outros grupos, isto é, os elementos não-compartilhados e os com compartilhamento unidirecional, geram o módulo M_2 ; e os elementos do grupo Gs_3 que também não sofreram interseção, logo, os elementos que o subsistema mantém, geram o módulo M_3 .

A modularização está, portanto, intimamente ligada aos subsistemas que fazem atualizações nos dados, isto é, aos subsistemas que mantêm essas informações.

A Tabela 2, ilustra os módulos M_1 , M_2 , M_3 e M_4 gerados:

Módulo	Elemento	É mantido por			
		S_1	S_2	S_3	S_4
M_1	Acompanhamento	x			
	Aguardo	x			
	Andamento	x			
	Assunto	x			
	Juntada	x			
	Processo	x			
	Requisicaoproc	x			
M_2	Cursogr		x		
	Disciplinagr		x		
	Habilitacaogr		x		
	Habilproggr		x		
	Habiliturma		x		
	Histescolagr		x		
	Histproggr		x		
	Programagr		x		
	Turmagr		x		
M_3	Agprograma			x	
	Area			x	
	Credarea			x	
	Diplomaposgrad			x	
	Disciplina			x	
	Idioma			x	
	Oferecimento			x	
	R39pgmorigdoc			x	
	R41pgmmattur			x	
	R46pgmproidi			x	
	R48pgmtrbdoc			x	
M_4	Pessoa		x	x	x
	Alunogr		x		x
	Alunopos			x	x
	Docente			x	x
	Servidor				x
	Cartcoseas		x	x	x
	Endpessoa		x	x	x
	Telefpessoa		x	x	x
	Tipovinculo				x
	Vinculopessoa				x

Tabela 2 – Definição dos módulos da base de dados

Nesse momento, com a modularização completa, pode-se iniciar a análise com relação à distribuição dos dados.

A quarta etapa dessa fase, chamada de “Definição da replicação sobre os módulos”, utiliza-se das estratégias de replicação, definidas na fase de Coleta e Análise de Tipos de Replicação (item 6.1.1), e da tabela, gerada pela segunda etapa dessa fase, para definir a melhor forma de replicar a base de dados. Pelo tipo de compartilhamento de cada elemento dos módulos gerados, pode-se analisar a forma de acesso dos elementos do módulo pelo subsistema e decidir pela forma de replicação ou até pela centralização dos dados.

Nesse exemplo, o módulo M_2 é utilizado apenas pelo subsistema S_2 e o módulo M_3 é utilizado apenas pelo subsistema S_3 , isto é, não há compartilhamento desses módulos, portanto, não há a necessidade de replicar nenhum de seus elementos.

Já nos elementos do módulo M_1 , existe um compartilhamento do tipo unidirecional, onde alguns elementos do módulo são acessados por S_1 e S_3 , porém somente o subsistema S_1 realiza operações de escrita sobre os dados. Nessa situação, pode-se adotar a estratégia de replicação de *Sites Replicados somente para leitura*, conforme visto no capítulo 5.

No módulo M_4 , as entidades *Tipovinculo* e *Vinculopessoa* também possuem compartilhamento unidirecional, e adota-se a mesma estratégia de *Sites Replicados somente para leitura*. Porém, no módulo M_4 existe o tipo de compartilhamento multidirecional em alguns elementos. Nesse caso, é preciso analisar o contexto em que esses elementos são atualizados pelos subsistemas, pois, em alguns casos, podem-se adotar estratégias de consistência forte que, às vezes, tem seu custo de implementação tão caro que se opta pela atualização centralizada e perde-se um dos grandes benefícios da replicação, que é a disponibilidade local. Ou podem-se adotar estratégias que permitam um pequeno tempo de latência entre a atualização e o recebimento do dado replicado atualizado, como, por exemplo, as estratégias *Atualização remota do Site Primário por conexões do cliente*, *Site Remoto requisita atualização do Site Primário*, *Site Remoto requisita atualização do Site Primário e mantém atualização temporária no local*, também vistas anteriormente no capítulo 5. É claro, que há situações em que não é possível relaxar a consistência das

atualizações em diversos locais, então é necessária a utilização de estratégias como o *ROWA* e suas variações, *Quorum consensus (QC)* ou *votação*, *Híbrido ROWA / QC* (item 4.3.1) ou outras estratégias existentes.

Por isso, a fase de Coleta e Análise de Tipos de Replicação deve definir quais serão as possíveis estratégias necessárias no ambiente corporativo e replicado, sendo eles implementados por um gerenciador, ou não.

Nessa etapa, pode-se averiguar que uma estratégia não tenha sido levantada e definida, na fase de Coleta e Análise de Tipos de Replicação, por isso cabe aqui uma reanálise dos tipos de replicação existentes, em conjunto com as reais necessidades de distribuição do negócio, além das restrições de ambiente e complexidade de implementação. Por isso, há a importância de uma boa documentação da fase anterior, para que não haja muitos ciclos em redefinição de estratégias.

No módulo M_4 , portanto, pode-se caracterizar que o envolvimento dos elementos *Alunogr*, *Alunopos*, *Docente* e *Pessoa*, que são atualizados pelos subsistemas S_2 , S_3 e S_4 , deverão implementar um protocolo de consistência forte, como *ROWA* ou *2PC*, e uma cópia de leitura de *Pessoa* deve ser disponibilizada para o subsistema S_1 . Para os elementos *Endpessoa* e *Telefpessoa*, que também possuem compartilhamento multidirecional para os subsistemas S_2 , S_3 e S_4 , pode-se identificar que seus dados permitem uma pequena latência de atualização e que dificilmente mais de um subsistema tenta alterar a mesma informação simultaneamente, por isso a estratégia escolhida pode ser a de *Site Remoto requisita atualização do Site Primário e mantém atualização temporária no local*. Dessa forma, é necessário eleger qual subsistema será o verdadeiro responsável pela atualização, e qual subsistema manterá a cópia local. Como o subsistema S_2 e S_3 somente gerencia os dados de alunos, e o volume de atualização de dados do subsistema S_4 é maior, devido à manipulação, também, dos dados de funcionários e professores, elege-se o subsistema S_4 como o real mantenedor dos dados. Assim, os subsistemas S_2 e S_3 mantêm uma atualização temporária de *Endpessoa* e *Telefpessoa* localmente até que sua requisição de atualização seja efetivada e replicada. Já o elemento do módulo M_4 , *Cartcoseas*, apesar de ter sido considerado de compartilhamento multidirecional pela etapa 2 desta fase, ao ser analisada sua forma de distribuição, pode-se constatar que os dados desta entidade são restritos a cada subsistema, pois o subsistema S_2 somente emitirá

carteiras para alunos de graduação; o subsistema S_3 , para os alunos de pós-graduação; e o subsistema S_4 , para funcionários e professores. Portanto, a replicação de dados desse elemento é desnecessária, pois ela pode ser particionada, isto é, cada subsistema fica encarregado de manter seus próprios fragmentos horizontais, porque, na realidade, seus dados são não-compartilhados. Caso os subsistemas fiquem em banco de dados separados, cada subsistema terá acesso a sua respectiva tabela referente à entidade *Cartco seas*.

Assim, tem-se, conforme a Tabela 3, uma visualização dos módulos, onde os dados são realmente atualizados, e a estratégia de replicação adotada para cada elemento do módulo, lembrando que as estratégias de replicação são definidas em sua maioria na fase de Coleta e Análise de Tipos de Replicação.

Módulo	Elemento	É atualizado por				Estratégia de Replicação
		S_1	S_2	S_3	S_4	
M_1	Acompanhamento	x				1
	Aguardo	x				1
	Andamento	x				1
	Assunto	x				1
	Juntada	x				1
	Processo	x				2
	Requisicaoproc	x				1
M_2	Cursogr		x			1
	Disciplinagr		x			1
	Habilitacaogr		x			1
	Habilproggr		x			1
	Habilturma		x			1
	Histescolargr		x			1
	Histproggr		x			1
	Programagr		x			1
	Turmagr		x			1
M_3	Agprograma			x		1
	Area			x		1
	Credarea			x		1
	Diplomaposgrad			x		1
	Disciplina			x		1
	Idioma			x		1
	Oferecimento			x		1
	R39pgmoridoc			x		1
	R41pgmmattur			x		1
	R46pgmproidi			x		1
	R48pgmtrbdoc			x		1

M_4	Pessoa		x	x	x	3
	Alunogr		x		x	3
	Alunopos			x	x	3
	Docente			x	x	3
	Servidor				x	1
	Cartcoseas		x	x	x	5
	Endpessoa		local	local	x	4
	Telefpessoa		local	local	x	4
	Tipovinculo				x	1
	Vinculopessoa				x	1
1 - não replica 2 - Sites Replicados somente para leitura 3 - Two Phase-Commit 4 - Site Remoto requisita atualização do Site Primário e mantém atualização temporária no local 5 - Atualização fragmentada por subsistema (não replica)						

Tabela 3 – Definição das estratégias de replicação para cada elemento dos módulos da base de dados

A quinta etapa “Definição da interface dos módulos da base de dados” é responsável por definir a interface de acesso aos dados para cada módulo da base de dados, especificado anteriormente, e considerando-se, nesta proposta, a estratégia de replicação, definida na etapa anterior.

Se nenhum gerenciador de replicação estiver sendo utilizado, é necessário que estas interfaces implementem o controle e consistências das réplicas. Por exemplo, nos casos em que o dado precisa ser atualizado em mais de um local, as interfaces de inserção, deleção e atualização precisam sincronizar essas atualizações, conforme definido na etapa anterior. Portanto, qualquer alteração em *Alunogr*, *Alunopos*, *Docente* e *Pessoa* deve ser feita sobre o protocolo *2PC*, garantindo, assim, a integridade na atualização.

Em sistemas homogêneos de estratégias de réplicas, esse tipo de controle torna-se transparente e automático pelo gerenciador de replicação, considerando que este implementa a estratégia de réplica estabelecida pelo projeto de modularização e replicação de dados. Entretanto, como neste trabalho queremos que coexistam várias estratégias é necessário que tal ferramenta tenha algumas adaptações.

As entidades que aparecem pontilhadas não existem realmente no módulo, apenas representam o relacionamento com as entidades dos outros módulos. Neste caso, é necessário manter o inter-relacionamento entre os módulos, indicando a

necessidade de manutenção da integridade referencial entre os módulos e entidades replicadas.

A seguir, os esquemas de dados de cada módulo gerado:

Figura 18 – Esquema de dados do módulo M_1

Figura 19 – Esquema de dados do módulo M_2

Figura 20 – Esquema de dados do módulo M_3

Figura 21 – Esquema de dados do módulo M_4

Mesmo com a adaptação do gerenciador de replicação, para o perfeito funcionamento de algumas estratégias, a definição da interface de cada módulo é importante na utilização simultânea de várias estratégias de replicação em um banco de dados. Através do estudo de caso ilustrado no capítulo 7, a interface será responsável, por exemplo, por manter as regras que permitam detectar todas as remoções conflitantes nos diversos bancos de dados.

Neste estudo de caso, será demonstrado um ambiente corporativo, com um grande número de entidades, relacionamentos e sistemas. A elaboração de módulos, nesse tipo de ambiente, é imprescindível para o gerenciamento do contexto das

transações, pois cada módulo gerado possui apenas os dados relevantes das suas transações. Além disso, o controle transacional é realizado pelas interfaces dos módulos, o que facilita a manutenção, caso o módulo venha a ser utilizado por outros subsistemas ou haja alteração no tipo de estratégia de replicação utilizado pela entidade.

7. Replicação de Dados e um estudo de caso

Conforme visto no capítulo anterior, o Projeto de Modularização e Replicação de Dados traz como principais vantagens um alto nível de gerenciamento das transações por negócio, e alta disponibilidade dos dados aos subsistemas que o utilizam. No entanto, o controle da integridade referencial, existente entre os módulos, torna-se ainda mais complexo quando aplicamos a replicação de dados sobre as entidades pertencentes a esses módulos.

7.1. Cenário da USP

O ambiente corporativo da USP, em ambiente de baixa plataforma, teve seu início por volta de 1994 e seu desenvolvimento foi de forma crescente e gradativa. O primeiro sistema teve como principal desafio a quebra de alguns paradigmas como, por exemplo, a substituição da plataforma mainframe, que mantinha o ambiente corporativo desde 1986, pela plataforma baseada na arquitetura cliente-servidor, inicialmente centralizada. Outro fator importante foi a padronização, no ambiente de desenvolvimento das aplicações, para que os usuários, ao adquirirem conhecimento do funcionamento de um sistema e sua interface visual, não sentissem dificuldade em trabalhar com outros sistemas, exceto no que dizia respeito ao próprio negócio tratado pelo sistema.

Hoje o modelo de dados corporativo da USP é composto por cerca de 880 entidades e mais de 1600 relacionamentos entre elas. Entre os principais sistemas corporativos existentes estão: o sistema Fênix, que controla todo o ambiente relacionado à Pós-Graduação; o sistema Júpiter, que gerencia toda a parte acadêmica da Graduação; o sistema Mercúrio, que trata da parte financeira, bens e materiais da Universidade de São Paulo; e o sistema Marte, que é responsável pelo controle da parte de recursos humanos; pessoal e folha de pagamento, entre outros sistemas - este último ainda em desenvolvimento.

No sentido de controlar este ambiente distribuído através de diversas estratégias de replicação, foi necessária a construção de uma ferramenta que auxilia a manutenção e gerenciamento das informações de distribuição, que será demonstrada no Apêndice A.

Cada sistema, para melhorar a sua disponibilidade, possui, como base principal, um banco de dados, localizado em uma máquina servidora. E, como todo modelo é corporativo, a necessidade de comunicação, entre as bases de dados, torna-se fundamental.

Dessa forma, foi necessário identificar quais os sistemas deviam manter os dados, isto é, atualizar as informações, e quais deveriam recebê-las via replicação.

Conforme as necessidades de acesso de cada sistema às informações, como no projeto de modularização, definiram-se os sistemas e, por conseguinte, os bancos de dados responsáveis, por manterem a atualização dos dados utilizando, se necessário, alguma das estratégias de replicação vistas a seguir.

Nesse ponto, o modelo corporativo USP, difere do projeto de modularização e replicação de dados, pois não se abstraiu o controle das transações através de pequenos subsistemas. Assim, quando um novo sistema precisa utilizar um contexto de transação já utilizado por outro sistema, é preciso analisar novamente quais são as entidades envolvidas e posteriormente disponibilizá-las.

Não se levando em conta a modularização dos subsistemas, foram estabelecidas estratégias de replicação, utilizando-se o Gerenciador de Replicação Sybase para controlar as réplicas nos diversos bancos de dados da USP. Mesmo com a utilização da modularização, essas estratégias de replicação podem ser utilizadas, pois levam em conta a implementação de como distribuir os dados, estejam eles em módulos ou não.

O projeto do modelo corporativo da USP é complexo, devido ao número de entidades e relacionamentos envolvidos. E este modelo, apesar de não ter sido elaborado com o projeto de modularização, é um estudo de caso que se adaptaria facilmente ao Projeto de Modularização e Replicação de Dados, como foi demonstrado, de forma simplificada, no exemplo do capítulo anterior, onde quatro módulos foram gerados a partir do modelo conceitual global (Figura 17).

Através da contextualização de pequenos subsistemas responsáveis pelo controle de matrícula dos alunos de graduação e pós-graduação, pelo gerenciamento de processos, pelo controle de pessoal da universidade e pela emissão de suas carteiras, foi possível criar módulos, o que trouxe, como principais vantagens, um nível maior de abstração no momento de reutilizarmos uma funcionalidade, e a

melhora no gerenciamento de qualquer alteração, seja ela no tipo de estratégia ou na interface do módulo.

7.2. Tipos de Tabelas

Como na fase Coleta e Análise de Tipos de Replicação do Projeto de Modularização e Replicação de Dados, este foi o momento de se levantar as formas e possibilidades de como os dados seriam replicados, conforme as suas necessidades de acesso, suas limitações ou as facilidades por parte do gerenciador de replicação utilizado pela Universidade de São Paulo.

Verificou-se, entre as necessidades existentes, uma forma de distinguir as tabelas, conforme o tipo de replicação em que ela estaria envolvida. Por isso, a criação da nomenclatura de tipos de tabelas vistas a seguir:

- **Única (U):** tabela que não participa de estratégias de replicação. Normalmente são tabelas utilizadas por apenas um sistema; quando mais de um sistema a utiliza, essa tabela representa uma partição horizontal, sem necessidade de replicação;
- **Primária (P):** tem o mesmo esquema da tabela lógica. É o tipo de tabela mais importante, pois é a que mantém os dados, isto é, a que permite atualização dos registros, não permitindo ser atualizada por meio de replicação. Conforme visto no capítulo 5, é nesse tipo de tabela que criamos a “Definição de Replicação”, que representa uma forma de publicação, indicando quais dados podem ser replicados;
- **Replicada (R):** também é semelhante à tabela lógica, possuindo o mesmo esquema. Porém, não permite atualizações em seus dados, que são originados via replicação. Isto é, sua atualização é feita via replicação quando a tabela tipo primária correspondente é atualizada. Aqui há a subscrição, isto é, um pedido de recebimento de informações replicadas, conforme a publicação da tabela primária. Este tipo de tabela recebe um subconjunto das linhas da tabela tipo primária, desde que, ao ser definida a publicação, algumas colunas tenham sido habilitadas para participarem como filtros de replicação, através de um parâmetro chamado de “cláusula where”, que, na realidade, são predicados de consulta do filtro. Assim, ao criar a subscrição, é possível

informar quais os valores dos predicados, trazendo, por replicação, somente um subconjunto das linhas da tabela original;

- **Fragmento Primário (F):** possui o mesmo esquema da tabela lógica, porém permite atualização em toda ou em apenas um subconjunto das linhas da tabela, podendo ou não, também, receber linhas replicadas. Este tipo de tabela permite, conforme sua configuração, tanto replicar, quanto receber dados. Logo, pode ter tanto publicação, quanto subscrições definidas, para a mesma tabela;
- **Projeção Vertical (V):** este tipo de tabela não possui o mesmo esquema da tabela lógica, pois possui um subconjunto de colunas dessa tabela. As linhas da tabela e seus dados são atualizados somente por replicação. Esta tabela subscreve um conjunto das colunas do esquema da tabela primária, isto é, possui uma subscrição, que é a seleção de apenas algumas colunas, declaradas na publicação da tabela com os dados tipo primário;
- **Consolidada (C):** é constituída por apenas um subconjunto do esquema lógico da tabela e tem um conjunto consolidado das linhas da tabela. Este tipo de tabela, funciona como centralizadora dos dados, provenientes de tabelas de mesmo esquema, mas particionadas. Também recebe somente por replicação a atualização de seus dados. O que difere este tipo de tabela da tabela (tipo) Replicada é que ela permite replicar as informações recebidas para outros locais, através da criação de uma nova publicação;
- **Auxiliar (A):** possui o mesmo esquema lógico da tabela e funciona como um repositório temporário nas estratégias de replicação. Mantém localmente os dados atualizados, até que a atualização ocorra via replicação, evitando, assim, problemas de latência.

7.3. Estratégias de Replicação utilizadas no Ambiente USP

Neste momento, dentre as estratégias de replicação existentes e as necessidades de estratégias específicas, foram elencadas as estratégias de replicação mais importantes para serem utilizadas no ambiente corporativo da USP.

As estratégias escolhidas estão representadas, conforme o seguinte modelo:

Figura 22 – Modelo de Representação da Tabela em um Banco de Dados

- **Estratégia 1:** tabela única, sem replicação.

Figura 23 – Representação da replicação com a estratégia 1

- **Estratégia 2:** um site primário, replicando integralmente a tabela.

Figura 24 – Representação da replicação com a estratégia 2

- **Estratégia 3:** um site primário, replicando iguais projeções verticais da tabela.

Figura 25 – Representação da replicação com a estratégia 3

- **Estratégia 4:** um site primário, replicando diferentes projeções verticais da tabela, com nomes diferentes.

Figura 26 – Representação da replicação com a estratégia 4

- **Estratégia 5:** “N” sites com fragmentos primários, replicando integralmente os fragmentos entre si.

Figura 27 – Representação da replicação com a estratégia 5

- **Estratégia 6:** “N” sites com fragmentos primários, replicando iguais projeções verticais: não há replicação entre os fragmentos, mas replicam-se projeções verticais entre os sites, com os nomes iguais.

Figura 28 – Representação da replicação com a estratégia 6

- **Estratégia 7:** “N” sites com fragmentos primários, replicando diferentes projeções verticais: não há replicação entre os fragmentos, mas replicam-se projeções verticais entre os sites, com os nomes distintos.

Figura 29 – Representação da replicação com a estratégia 7

- **Estratégia 8:** “N” sites com fragmentos primários, replicando para um site consolidado: não há replicação entre os fragmentos, mas destes para um consolidado de mesmo nome.

Figura 30 – Representação da replicação com a estratégia 8

- **Estratégia 9:** “N” sites com fragmentos primários, replicando para um site consolidado e, a partir daí, replicando integralmente a tabela: não há replicação entre os fragmentos, mas destes para um consolidado, de nome igual, e deste para replicadas.

Figura 31 – Representação da replicação com a estratégia 9

- **Estratégia 10:** “N” sites com fragmentos primários, replicando para um site consolidado e, a partir daí, replicando iguais projeções verticais da tabela: não há replicação entre os fragmentos, mas destes para uma consolidada, de nome igual, e desta para as projeções.

Figura 32 – Representação da replicação com a estratégia 10

- **Estratégia 11:** “N” sites com fragmentos primários, replicando para um site consolidado e, a partir daí, replicando diferentes projeções verticais da tabela: há replicação dos fragmentos para a consolidada, de nome igual, e desta para as projeções de nomes distintos.

Figura 33 – Representação da replicação com a estratégia 11

- **Estratégia 12:** “N” sites com fragmentos primários, replicando integralmente os fragmentos entre si, para vários outros sites replicados, e um consolidado; e, a partir do consolidado, para vários outros sites replicados.

Figura 34 – Representação da replicação com a estratégia 12

- **Estratégia 13:** “N” sites com fragmentos primários, replicando entre si e para vários outros sites replicados, todos com os mesmos nomes.

Figura 35 – Representação da replicação com a estratégia 13

- **Estratégia 14:** “N” sites com fragmentos primários, replicando integralmente os fragmentos entre si e para um consolidado, e, a partir do consolidado, para vários outros sites replicados.

Figura 36 – Representação da replicação com a estratégia 14

- **Estratégia 15:** um site primário, replicando integralmente várias tabelas com nomes diferentes para uma tabela consolidada em outro site.

Figura 37 – Representação da replicação com a estratégia 15

- **Estratégia 16:** um site primário, replicando integralmente a tabela, sendo que, na inserção e alteração, a linha é inserida ou alterada no site primário e local. No caso de inserção, deve existir uma função no site local para evitar uma redundância do dado.

Figura 38 – Representação da replicação com a estratégia 16

Nesta situação, para que o dado seja inserido no site local, é obrigatória a inserção simultânea no site primário, indicado por (i). No momento em que o site primário replica os dados para os sites, é verificado, através de uma coluna na tabela A, qual o site efetuou o cadastro ou alteração da informação, para que, neste momento, a replicação (ii), ao chegar no site local, seja desprezada para evitar a sua redundância.

- **Estratégia 17:** um site primário, replicando integralmente a tabela. No site replicado, mantém-se uma tabela auxiliar com as atualizações pendentes.

Figura 39 – Representação da replicação com a estratégia 17

- **Estratégia 18:** um site primário, replicando integralmente a tabela para vários sites replicados, e um site consolidado; e, a partir do consolidado, para vários outros sites replicados.

Figura 40 – Representação da replicação com a estratégia 18

Essas são as possíveis estratégias de replicação definidas, porém, nem todas são necessariamente utilizadas.

Algumas estratégias de replicação necessitam de regras bem claras para sua perfeita utilização, pois todo o ambiente corporativo precisa estar íntegro, como no caso da utilização de tabelas auxiliares, em que o dado deve ser removido da tabela auxiliar, quando a informação for atualizada, por intermédio da replicação, na tabela replicada. Estas regras podem ser programadas na própria estrutura do gerenciador

de replicação, e, em algumas situações, deve-se especificar um tratamento nas interfaces geradas nos módulos (item 6.1.2).

Além disso, como o modelo corporativo teve um crescimento progressivo, quanto às suas necessidades, em algumas situações, não foi possível aplicar a replicação direta a partir do conteúdo das tabelas. Isto é, as colunas existentes não são consideradas como predicados válidos para filtrar as informações (cláusula where) que se deseja replicar.

Neste caso, foi necessário criar regras de contextualização da informação, onde os dados pudessem ser extraídos e, então, serem replicados utilizando uma das estratégias acima. Essas regras de contextualização, na realidade, são regras de negócio, que estipulam quais informações são realmente foco de interesse de replicação.

Por exemplo, uma unidade da USP pode desejar receber informações dos sistemas corporativos, com o propósito de emitir relatórios localmente ou, até mesmo, criar sistemas locais, evitando redundâncias de informações já existentes nos sistemas corporativos. Nesse caso, é importante contextualizar as informações que devem ir para essa unidade.

Se é desejado um relatório com todas as disciplinas cursadas por um aluno, é provável que as disciplinas cursadas em outras unidades sejam necessárias. Nessas situações, são identificados os relacionamentos do aluno com as disciplinas, extraindo-se, assim, as informações relevantes à real necessidade do negócio e não só as disciplinas da unidade do aluno ou todas as disciplinas de todas as unidades, o que causaria falta ou excesso de informação no local.

Com os dados extraídos e tratados pela regra de negócio (definição dos dados necessários), há a criação de tabelas intermediárias, por contexto da unidade, e então é aplicada a replicação integral destas tabelas para os bancos de dados locais das unidades, que, portanto, subscrevem a tabela intermediária.

7.4. Formas de Remoção

Outra regra a ser tratada é a integridade referencial entre as tabelas replicadas. Este cuidado, já mencionado no Projeto de Modularização e Replicação de Dados, (item 6.1.2) é importante, pois, ao analisar a replicação de dados, os contextos de

relacionamentos tornam-se distintos em cada banco de dados, isto é, quando é aplicada a replicação, pode-se verificar que os relacionamentos encontrados no esquema conceitual global foram dispersos pelos módulos gerados e posteriormente pela replicação física dos dados em vários BD.

Nesse caso, uma ação de remoção, realizada com sucesso no BD primário, pode ocasionar uma inconsistência no ambiente corporativo, pois outros relacionamentos, que não se encontram no BD primário, podem ter sua integridade referencial corrompida.

Por se tratar de um processo assíncrono, a remoção já efetivada no BD primário acaba causando esta perda de integridade, pois, ao se replicar a ação de remoção no BD replicado, este retorna um erro de integridade referencial ao gerenciador de replicação daquele banco.

Essa quebra de integridade referencial é identificada pelo gerenciador de replicação, que interrompe a conexão entre o servidor de replicação e o BD replicado, porém enfileirando toda e qualquer operação a ser realizada no BD replicado.

Este tipo de problema não ocorre para ações de atualização, pois uma das regras adotadas no ambiente corporativo da USP é a proibição na alteração das chaves primárias. Nesse caso, todas as consistências ocorrem no BD primário e, então, são replicadas sem nenhum problema para os BD replicados.

Porém, o problema da remoção, causando a interrupção, na conexão entre o servidor de replicação e o banco de dados, pode acarretar ainda a perda de informações, caso o tamanho da fila em que os dados são armazenados não seja suficiente. Por isso, criaram-se algumas regras para se reduzir a queda no servidor de replicação, ocasionado na remoção dos dados por problemas de integridade referencial:

- **Remoção proibida:** a solução é obtida eliminando-se o caso gerador do problema, isto é, impedindo-se a remoção do dado. É claro que esta estratégia de remoção só é possível no caso em que os dados não precisem realmente ser apagados, como tabelas de histórico. A restrição deve ser programada na interface que controla a remoção da informação no módulo, na aplicação cliente ou em *trigger*;

- **Remoção por verificação:** neste caso, uma ação de remoção sobre uma tabela responsável por manter o dado, isto é, uma remoção sobre um dado primário, além de verificar a integridade referencial com os relacionamentos existentes no próprio BD local, deve verificar também a integridade referencial nos outros bancos, antes de efetivar esta remoção no BD local. As regras de verificação devem ser programadas na interface de remoção do módulo ou no cliente;
- **Remoção por *trigger*:** esta estratégia de remoção considera que os vínculos de integridade referencial existentes entre as tabelas no BD replicado não são restritivos para a ação de remoção. Por isso, substitui a integridade referencial declarativa dos BD replicados, por integridade referencial programada, através de *triggers*, optando por remoção em cascata, nulificação ou imposição do valor padrão para a referência;
- **Remoção por n-phase commit:** nesta estratégia é realizada a remoção simultânea no BD primário e nos BD replicados. Esse tipo de estratégia deve ser programada na interface de remoção do módulo ou no cliente. É mais complexa, pois exige a programação do n-phase commit através de bloqueios exclusivos e simultâneos, em várias estruturas de diferentes BD, além de ser necessário desativar a replicação para a ação de remoção.

7.5. Metamodelo de Apoio à Replicação

Para manter o ambiente corporativo, com bancos de dados replicados, de forma controlada e gerenciável, foi necessária a elaboração de um catálogo para armazenar as informações relativas ao gerenciador de replicação utilizado, às definições do ambiente de replicação e suas estratégias, e aos servidores de banco de dados e suas localizações. Assim, as informações deste próprio catálogo também podem participar das definições de replicação, considerando-se, portanto, esse catálogo, um metamodelo de auxílio às definições de replicação.

Através desse metamodelo é possível armazenar, por exemplo, qual a estratégia de replicação foi definida para cada entidade, quais as suas localizações, isto é, em quais bancos de dados aquela entidade irá existir e de que forma: replicando ou subscrevendo informações.

O metamodelo, portanto, é responsável por manter as informações analisadas e definidas no Projeto de Modularização e Replicação de Dados.

Alguns detalhes do metamodelo estão no Apêndice B. A seguir, apresenta-se o modelo relacional do metamodelo:

Figura 41 – Modelo Relacional do Metamodelo de Apoio à Replicação

Ao analisarmos o exemplo de modularização e replicação do capítulo 6, podemos encontrar algumas inconsistências, devido à replicação de tabelas para um banco de dados, conforme a necessidade do sistema.

A entidade *Processo*, no nosso exemplo, é necessária para o subsistema S_1 e S_3 (Tabela 1). Ao aplicar-se a modularização, definiu-se que o módulo responsável por manter esta entidade é o módulo M_1 (Tabela 2) que, por sua vez, é utilizado pelo subsistema S_1 , responsável por atualizar suas informações. Ao estudar-se a melhor forma de replicação, optou-se pela estratégia de *Sites Replicados somente para leitura*, que é implementada pela estratégia 2 (item 7.3), utilizada pela USP. Nesse caso, o subsistema S_1 possui as informações primárias, e o subsistema S_3 recebe as informações replicadas somente para leitura.

Porém, ao analisarmos o modelo conceitual global, encontramos a existência de um relacionamento intermodular entre as entidades *Processo*, do módulo M_1 , e a entidade *Diplomaposgrad*, do módulo M_3 . Nesse caso, conforme visto anteriormente, caso haja a remoção de algum registro de *Processo* pelo subsistema S_1 , e este estiver relacionado com algum registro de *Diplomaposgrad*, acarretará no erro do gerenciador de replicação, que suspenderá a replicação até que o problema seja solucionado.

Para que isso não ocorra, é necessário que a interface de remoção da entidade *Processo*, gerada pela quinta etapa, “Definição da interface dos módulos da base de dados” do Projeto de Modularização e Replicação de Dados, verifique inicialmente se existe a relação do registro de *Processo* com *Diplomaposgrad*. O registro deve ser removido no momento em que não estiver sendo referenciado por nenhuma outra entidade, mantendo, assim, a integridade referencial entre os dados.

Com o controle transacional sobre a interface, o gerenciamento torna-se mais fácil e centralizado, pois o controle da remoção não precisa ser feito pelos aplicativos desenvolvidos em diversas linguagens e plataformas, e sim pela implementação de algum procedimento no próprio banco de dados.

Estas inconsistências podem ser causadas pelos relacionamentos intermódulos e pelos relacionamentos entre tipos diferentes de tabelas, oriundos da diversificação de estratégias de replicação, utilizados em um banco de dados. No sentido de detectar essas inconsistências, foi construído o verificador, visto a seguir.

7.6. Verificador de Consistência de Replicação em um Banco de Dados

O metamodelo é capaz de armazenar todas as informações necessárias para controlar as inconsistências entre os relacionamentos de tabelas replicadas em diversos bancos de dados. Nele é possível especificar qual regra de remoção deve ser adotada para cada tabela, diminuindo, assim, o risco de o servidor de replicação ficar fora de operação, devido a uma inconsistência de integridade referencial, causada pela violação em um banco de dados com informações replicadas. Porém, esta identificação da regra de remoção é apenas cadastral e pode estar desatualizada com o tempo.

A fim de solucionar esse problema, foi construído um verificador, que percorre todas as informações das tabelas e relacionamentos definidos para um banco de dados específico, a partir do modelo conceitual global. Dessa forma, é possível analisar, em qualquer momento, se os relacionamentos físicos cadastrados no metamodelo estão compatíveis com as regras de remoção, e os tipos das tabelas que fazem parte deste relacionamento.

Assim, como foi visto anteriormente:

- as tabelas replicadas não devem estar relacionadas com outras tabelas replicadas em um banco de dados, pois suas informações são sempre atualizadas nas respectivas tabelas primárias, onde seus dados são consistidos e posteriormente atualizados nos demais bancos de dados, via replicação;
- as tabelas com dados primários devem estar relacionadas com as outras tabelas com dados primários. É a forma comum de se manter a integridade referencial entre elas;
- nenhuma tabela com dados primários pode estar relacionada com tabelas com dados replicados, pois, as tabelas com dados replicados apenas recebem dados replicados de suas próprias tabelas primárias, localizadas em outro banco de dados;
- as tabelas com dados replicados, que são referenciadas por tabelas constituídas de dados primários, estão sujeitas a inconsistências, pois os dados efetivamente removidos dos sites primários, ao serem replicados

para sites replicados, encontram relacionamentos dessas tabelas replicadas com outras tabelas com dados primários, causando erro na replicação.

A partir dessas regras fundamentais, é possível analisar todo o banco de dados, identificando-se para cada tabela, os relacionamentos pertencentes ao modelo conceitual global, obrigatórios para aquele banco de dados específico, e os que não são obrigatórios. Essa informação deve ser acrescentada à exigência ou não da existência do relacionamento, através do cadastramento no metamodelo.

7.7. Implementação Física da Base de Dados e Ferramenta de Apoio ao Metamodelo

A partir da elaboração do metamodelo, responsável por controlar as informações de replicação do ambiente corporativo da USP, foi necessária a construção de um aplicativo (Apêndice A) que permitisse o cadastro e controle das informações, desde o projeto conceitual global (como efetivar a distribuição de dados, através das diversas estratégias de replicação) até se chegar ao projeto de replicação dos dados em desenvolvimento, inclusive gerando esquemas necessários para o gerenciador de replicação utilizado.

O verificador de inconsistências, implementado nessa ferramenta, analisa, principalmente, os tipos das tabelas envolvidas em cada relacionamento. Uma tabela com dados primários (tabelas Única, Primária ou Fragmento), ao referenciar informações de uma tabela com dados replicados (tabelas Replicada, Consolidada ou Projeção Vertical), pode gerar a inconsistência do BD. Nesse caso, analisa-se se a tabela referenciada possui estratégia de remoção, e, caso a estratégia exista, se o relacionamento é exigido ou não. Dessa forma, é possível comparar a obrigatoriedade da existência do relacionamento com os dados cadastrados no metamodelo, sabendo-se, assim, se o relacionamento foi devidamente informado no metamodelo ou não. Há situações em que o relacionamento está cadastrado no metamodelo, mas a estratégia de remoção proíbe a existência desse relacionamento.

Portanto, com o auxílio do verificador, é possível fazer uma avaliação, tanto nos relacionamentos entre tabelas, quanto na definição das estratégias de remoção escolhidas.

É possível adotar a estratégia de remoção para qualquer tipo de tabela, por isso o verificador deve analisar a relação, mesmo entre os tipos de tabelas que não

causam inconsistência. Por exemplo, pode-se especificar que a integridade referencial entre duas tabelas tipo primária será feita por *trigger*; nesse caso, opta-se pela estratégia de remoção por trigger. O verificador, por sua vez, é responsável por analisar, e proibir a existência desse relacionamento, apesar do relacionamento ser considerado compatível.

No Apêndice C, encontra-se um exemplo do resultado gerado pelo verificador. Parte da lógica implementada nesse verificador encontra-se a seguir:

```
Se tabela (U ou P ou F) referenciar tabela (R ou C ou V), então
  Se tabela possui estratégia de remoção, então
    Se estratégia de remoção exige o relacionamento, então
      Se o relacionamento existe, então
        "OK. A definição física do relacionamento foi encontrada
        conforme a estratégia de remoção exige"
      Se o relacionamento não existe, então
        "Erro. A definição física do relacionamento não foi
        encontrada, conforme a estratégia de remoção exige"
    Se estratégia de remoção não exige o relacionamento, então
      Se o relacionamento existe, então
        "Erro. A estratégia de remoção da tabela não permite o
        relacionamento"
      Se o relacionamento não existe, então
        "OK. A definição física do relacionamento não foi
        encontrada, conforme a estratégia de remoção proíbe"
  Se tabela não possui estratégia de remoção, então
    "Aviso: Relacionamento pode gerar inconsistência.
    Favor cadastrar estratégia de remoção"
Se tabela (U ou P ou F) não referenciar tabela (R ou C ou V), então
  Se tabela possui estratégia de remoção, então
    Se estratégia de remoção exige o relacionamento, então
      Se o relacionamento existe, então
        "OK. A definição física do relacionamento foi encontrada
        conforme a estratégia de remoção exige"
      Se o relacionamento não existe, então
        "Erro. A definição física do relacionamento não foi
        encontrada, conforme a estratégia de remoção exige"
    Se estratégia de remoção não exige o relacionamento, então
      Se o relacionamento existe, então
        "Erro. A estratégia de remoção da tabela não permite o
        relacionamento"
      Se o relacionamento não existe, então
        "OK. A definição física do relacionamento não foi
        encontrada, conforme a estratégia de remoção proíbe"
  Se tabela não possui estratégia de remoção, então
    Se o relacionamento existe, então
      "OK. A definição física do relacionamento foi encontrada"
    Se o relacionamento não existe, então
      "Erro. A definição física do relacionamento não foi
      encontrada"
Se tabela (R ou C ou V) referenciar outro tipo de tabela, então
  "Erro. Este tipo de tabela não pode referenciar outra tabela,
  pois, seus dados atualizados por replicação"
```


8. Conclusão

8.1. Considerações Iniciais

Este trabalho teve como objetivos:

- a) aplicar um conjunto de diretrizes para a decomposição do esquema global de dados (modularização), para simplificar o controle das estratégias de réplicas em modelos corporativos de dados;
- b) redefinir um repositório de dados para armazenar informações das estratégias de réplicas, e desenvolver um verificador de inconsistências para controlar o projeto lógico de réplicas dos bancos de dados.

Esses dois objetivos puderam ser atendidos com a adaptação de um metamodelo de dados, desenvolvido com a aplicação das técnicas de modularização.

No levantamento bibliográfico (capítulo 2, 3, 4), descreveram-se os recursos disponíveis para controle de réplicas, as técnicas para os mais variados tipos de fragmentação e distribuição de dados, e ainda os conceitos e fases da modularização de dados.

Com tal levantamento, conseguiu-se caracterizar a necessidade da representação de um metamodelo de dados para o controle de réplicas, com regras e delimitações, realizadas pelo processo de modularização e replicação de dados (capítulo 6).

Após a aplicação das fases de modularização, criou-se a representação dos módulos, no metamodelo de dados, para controle das estratégias de réplicas. Uma vez representado devidamente tal metamodelo, procedeu-se à construção do verificador de inconsistências (capítulo 7).

8.2. Contribuições Diretas

No início de nosso estudo, pôde-se constatar que a principal dificuldade para gerenciar um ambiente com diversas estratégias de replicação era a compatibilização entre estas estratégias. Porém, em nosso estudo de caso, pôde-se perceber, que o conflito entre as estratégias de replicação não existe. Há, na realidade, uma

incompatibilidade entre o relacionamento de tabelas de tipos diferentes, utilizadas ao se escolher uma estratégia de replicação.

O uso da replicação com a modularização tornou mais compreensivo e, além de tudo, mais fácil o gerenciamento do controle de remoção de informações de tabelas replicadas, pois as regras residem, de forma centralizada e documentada, na própria interface criada no módulo. Caso se necessite replicar alguma entidade desse módulo, é possível fazê-lo de forma mais fácil, pois as regras estão devidamente atualizadas em um único módulo. Por exemplo, se o relacionamento de tabelas pertencentes a módulos diferentes possuem conflito na remoção em um determinado site, é necessário apenas definir, na interface da tabela, qual a regra de controle, para que a verificação do conflito ocorra anteriormente à efetivação da remoção. Assim, a regra estará sempre presente onde os dados são atualizados (onde os dados são primários), não dependendo do controle da aplicação para gerenciar um possível conflito, e, além disso, caso a aplicação mude de linguagem ou alguma regra seja alterada, basta documentar na interface do módulo, e atualizar fisicamente, no banco de dados responsável por manter os dados, isto é, onde a tabela possui dados primários.

Com a construção do verificador de consistência do banco de dados, também pôde-se perceber que o informe sobre as estratégias de remoção deveriam estar em cada relacionamento, pois há situações em que alguns conflitos, na remoção de um dado de uma tabela, em um banco de dados específico, podem ser tratados de formas diferentes (item 7.4).

8.3. Contribuições Decorrentes

8.3.1. Validação do Trabalho com a Utilização do Estudo de Caso

A utilização de um estudo de caso, para melhor delimitar e ilustrar o problema, e ainda validar as soluções propostas, mostra-se uma estratégia interessante para abordagem de dissertações de mestrado, pois permite a mediação entre teoria e prática. O trabalho em questão contribui para os trabalhos de dissertação que objetivam ter tal enfoque.

8.3.2. Delimitação e Diminuição do Grau de Heterogeneidade dos Sistemas

Com a modularização, consegue-se diminuir o grau da heterogeneidade dos sistemas, na medida em que a interface de acesso aos módulos está bem-definida. Com isso, além da diminuição, consegue-se uma delimitação da heterogeneidade dos sistemas, pois toda e qualquer interação entre os módulos deve ser feita através de interface bem-definida. Essas interfaces contemplam os serviços responsáveis pela identificação das transações e dos métodos, o mapeamento da localização dos procedimentos e regras ativas nos módulos, a conectividade com os servidores de dados, e a execução dos procedimentos e das regras ativas.

8.3.3. Contribuição para Projetos e Técnicas de Fragmentação e Distribuição de Dados mais Eficientes

Com a adição da modularização ao projeto de bases de dados, as técnicas disponíveis na literatura, para fragmentação, podem ser melhor aplicadas, pois a caracterização da utilização de fragmentos de dados, do ponto de vista dos subsistemas, já está pré-estabelecida. Isso diminui o volume de conflitos para a determinação dos fragmentos de dados. Em outras palavras, um grande fragmento de dado, o módulo de dados, já foi criado. As técnicas de fragmentação poderão recair sobre os módulos, que foram criados através da proposta apresentada neste trabalho.

Podemos comprovar isso através de uma afirmação de Ceri e Pelagatti [28], em que se ressalta a necessidade de diferenciar as classes de fragmentos de dados no momento anterior ao processo de distribuição de dados. Considerando um esquema de dados corporativo, segundo Ceri e Pelagatti [28], alguns fragmentos de dados têm caráter administrativo, outros de produção e outros de recursos humanos.

Nessa afirmação, podemos encontrar, na verdade, uma certa obscuridade na utilização dos fragmentos. Com o uso da modularização, podemos caracterizar os fragmentos através do esquema parcial de dados do qual se originaram. Dessa maneira, o esquema parcial de um determinado módulo de dados é que caracteriza os fragmentos, que serão criados e localizados no ambiente de distribuição de bases de dados.

Desse modo, a caracterização do esquema parcial de dados está vinculada à especificação de cada subsistema. As possíveis alternativas de fragmentação e distribuição de dados devem agora ser aplicadas às instâncias dos dados dos esquemas parciais, armazenados nos respectivos módulos de dados.

Com a modularização, o controle das formas de réplicas que recaem sobre os módulos podem ser mais facilmente controladas. O metamodelo de dados, por sua vez, é simplificado e o verificador tem sua implementação mais eficiente podendo ser utilizado em tempo de execução.

8.4. Futuras Pesquisas

8.4.1. Transformar as Diretrizes para Controle das Formas de Réplicas em Metodologia

Utilizando-se recursos de especificação de software formal, é possível criar uma metodologia para controle das formas assíncronas de réplicas. O trabalho apresentado nessa dissertação é um ponto de partida para a concepção da metodologia mencionada.

8.4.2. Incorporar o Controle em uma Ferramenta *CASE*¹³

É possível organizar as diretrizes de controle de formas de replicação, levantadas nesse trabalho, de modo a criar um ambiente de distribuição, associado a ferramentas *CASE* de apoio ao desenvolvimento de bases de dados. A criação dos controles das formas de réplica seria realizada pelo ambiente em questão, através da entrada de parâmetros que caracterizariam os aspectos mencionados nos capítulos 3, 6 e 7.

8.4.3. Alterar o Catálogo do Gerenciador de Replicação

Com o objetivo de garantir eficiência na verificação das consistências, pode-se incorporar ao catálogo do gerenciador de replicação a estrutura do metamodelo e o verificador desenvolvido por essa dissertação.

Isso poderia ser feito incluindo-se a base de dados do metamodelo como uma

¹³ Do inglês: Computer Aided Software Engineering.

base descritora de dados, e transformando as regras de consistências do verificador em procedimentos armazenados nessa base.

8.4.4. Desenvolver Regras Ativas para Controle de Consistências Complexas

Atualmente, a implementação de regras ativas restringe-se a variações de implementação de *triggers* em SGBD's. No metamodelo de dados, é possível representar vários tipos de consistências. Entretanto tal implementação pode pressupor muitos níveis de recursividade. Para que não haja limites para controlar as complexidades das consistências, é preciso incorporar as soluções utilizadas em situações de regras ativas à base descritora. Como exemplo, para garantir o atendimento às consistências, pode-se implementar a utilização de uma máquina de estados finitos ou estruturas análogas.

8.4.5. Apoio à Data-Mining Distribuído

Considerando a ferramenta para controle de inconsistências de replicação de dados é possível incorporar técnicas de data-mining para otimizar as análises de distribuição e replicação dados. Tais técnicas de data-mining devem ser incorporadas ao metamodelo de dados para descobrir os comportamentos de distribuição e replicação de dados. Isso trará um grande poder de escala no controle otimizado das informações.

Apêndice A

A.1. Gerenciador Lógico de Replicação de Dados

Este aplicativo foi concebido pelo Departamento de Informática da USP, baseado no metamodelo, com o propósito de manter um controle sobre as formas de replicar as tabelas nos diversos bancos de dados da Universidade de São Paulo.

No entanto, as definições não passaram por uma fase de modularização, segundo as transações realizadas sobre as tabelas, conforme visto no capítulo 6. As definições para realizar a replicação foram elaboradas diretamente a partir do informe de acesso às tabelas pelos sistemas existentes, semelhante à Tabela 1, porém sem delimitar os subsistemas pertencentes aos sistemas.

É necessário observar, portanto, que o uso desse aplicativo, no contexto deste trabalho, é o auxílio na quarta etapa do Projeto de Modularização e Replicação de Dados (item 6.1.2), que é a “Definição da replicação sobre os módulos”, conforme visto no exemplo da escolha das estratégias de replicação do capítulo 6, sem aplicar, no entanto, as etapas anteriores que justamente geram os módulos.

As informações básicas, oriundas do modelo conceitual global de dados, como tabelas, colunas, tipo de dado, tamanho, obrigatoriedade, regras e o relacionamento entre as tabelas são extraídas da ferramenta de modelagem de dados, utilizada pelo departamento.

Após essas informações iniciais estarem cadastradas é possível, através do aplicativo aqui chamado de GARAD (Gerenciador de Ambiente de Replicação Assíncrono de Dados), fazer os seguintes cadastros:

Primeiramente, todos os servidores de bancos de dados utilizados pelo ambiente corporativo devem ser cadastrados. Isto é, tanto os servidores com função de gerenciar os bancos de dados com informações do modelo corporativo, propriamente dito, como os servidores responsáveis pelo gerenciamento do ambiente de replicação que, conforme visto no capítulo 5 (item 5.2), também mantém um banco de dados chamado RSSD (Replication Server System Database), devem ser cadastrados nesse momento.

O banco de dados RSSD é responsável por armazenar todas as informações, para que a replicação ocorra de forma correta, por parte do gerenciador de replicação. Esse banco de dados possui, internamente, a estratégia de fragmento primário, pois todos os Servidores de Replicação recebem as alterações realizadas pelos demais Servidores de Replicação que compõem a rede.

Outras informações, como *tipo do servidor*, isto é, se o servidor é responsável por dados de desenvolvimento ou produção, *nome físico do servidor*, *nome da máquina* onde o servidor se encontra e o seu *endereço* também são necessárias nessa tela.

A imagem mostra a interface gráfica do sistema GARAD, especificamente a janela 'Servidores (1/1)'. A janela possui uma barra de menu com 'Arquivo', 'Editar', 'Buscar', 'Objeto', 'Consultar' e 'Janela'. Abaixo da barra de menu, há uma barra de ferramentas com ícones para salvar, cancelar, adicionar, remover e atualizar. O formulário principal contém os seguintes campos:

- Nome do Servidor: PROJETO (campo de texto com seta para baixo)
- Tipo: Desenvolvimento (campo de texto com seta para baixo)
- Função do Servidor: Dados (campo de texto com seta para baixo)
- Nome Físico do Servidor: PROJETO (campo de texto)
- Nome da Máquina: sirius (campo de texto)
- Endereço da Rede: 143.107.1.4 (campo de texto)

Na base da janela, há uma barra de status que indica: 'Busca, no banco de dados, o(s) elemento(s) especificado(s)'.

Figura 42 – Tela de Servidores de Banco de Dados

Após o cadastro das informações dos servidores de bancos de dados, é possível fazer o informe dos bancos de dados utilizados pelos Sistemas Corporativos e fazer o vínculo desses bancos de dados com seus respectivos servidores, isto é, com os responsáveis pelo gerenciamento de suas informações, conforme mostra a próxima figura.

Figura 43 – Tela de Banco de Dados

A Figura 44 representa a definição das estratégias de replicação, definidas na fase de Coleta e Análise de Tipos de Replicação. É aqui onde se definem quais os tipos de tabelas a estratégia permite. Outros parâmetros utilizados pelo gerenciador de replicação, como o número mínimo e máximo de publicações e subscrições devem existir, também são informados nesta tela.

No caso da estratégia 18, pode-se perceber que a estratégia permite as tabelas do tipo *primária*, *replicada* e *consolidada* e, no mínimo, duas “Publicações” e três “Subscrições”. As publicações são criadas nas tabelas tipo *primária* e *consolidada*.

A tabela *consolidada*, por sua vez, também possui uma subscrição da tabela *primária*. Ainda deve existir, pelo menos, uma tabela *replicada*, subscrevendo informações da tabela *primária*; e uma outra tabela *replicada*, subscrevendo informações da tabela *consolidada*, totalizando o mínimo de três subscrições para essa estratégia.

Estratégias de Replicação (18/18)

Número: **18** Descrição da Estratégia: Um site primário, replicando integralmente a tabela para n sites replicados e um site consolidado e, a partir do consolidado, para outros n sites replicados (P+ <n>R & C+<n>R): tabelas de nome igual.

Opções	Mínimo	Máximo
Replication definition de tabela	2	2
Subscription de tabela	3	0
Function string de inserção	0	0
Function string de deleção	0	0
Function string de alteração	0	0
Tabela auxiliar	0	0
Replication definition de procedure	0	0
Subscription de procedure	0	0
Permite mudar o nome da tabela replicada:	não	

Tipos de tabelas previstas

- Única: não
- Primária: sim
- Replicada: sim
- Fragmento primário: não
- Consolidada: sim
- Projeção vertical: não
- Auxiliar: não

Buttons: salvar, cancelar, incluir, excluir, ajuda

Figura 44 – Tela de Estratégias de Replicação

Neste momento, com todas as informações referentes ao modelo conceitual global, com os servidores de bancos de dados, com os bancos de dados e com as estratégias cadastradas, inicia-se, então, a fase de cadastro das definições de

replicação, obtidas na quarta etapa do Projeto de Modularização e Replicação de Dados (item 6.1.2).

A partir das estratégias de replicação (item 7.3), que foram elaboradas de acordo com o levantamento efetuado no capítulo 6, na fase de Coleta e Análise de Tipos de Replicação (item 6.1.2), cada tabela do modelo conceitual global passa por uma fase de definição do modelo físico de replicação, onde são cadastradas a estratégia de replicação de cada tabela e as definições dos bancos de dados em que essa tabela deve existir, publicando ou recebendo as informações, isto é, replicando ou subscrevendo informações, através de publicações (Replication Definitions) e subscrições (Subscriptions).

Figura 45 – Tela de Representação do Modelo Físico de Replicação de Dados

É possível identificar nessa tela, através do informe em *Estratégia de Remoção*, qual a estratégia deve ser adotada para a tabela naquele banco de dados específico, no momento em que houver conflito na remoção de um registro referenciado por outra tabela.

É essa informação que é utilizada pelo verificador de consistência dos relacionamentos do banco de dados, construído nesta dissertação.

Na publicação, conforme citado anteriormente neste trabalho, é possível identificar quais as colunas da tabelas serão replicadas e se alguma coluna poderá fazer parte de filtros, através da opção de predicados (“cláusula where”):

"Replication Definition"

servidor: UBATUBA banco: DVANGOGH tabela: CONVENIO tipo: Primária

descrição: Convênios firmados com a USP.

estratégia de replicação: Um site primário, replicando integralmente a tabela (P+<n>R).

replication definition: f_dva_convenio send standby all: ☐ replicate minimal: ☒

colunas	replicada	cláusula where	ordem na chave de replicação	replicar quando
nomcvn	☒	☐		
sglcvn	☒	☐		
dtaasicvn	☒	☐		
dtadtvcvn	☒	☐		
stafin	☒	☒		
staacd	☒	☒		
objcvn	☒	☐		

alterada

Figura 46 – Tela de Publicações

Nesse exemplo, a tabela em questão possui colunas que podem ser utilizadas no momento da criação da subscrição para selecionar apenas os registros com um determinado valor. Essas colunas estão marcadas, conforme o item *cláusula where* da Figura 46.

Já na definição da subscrição, caso alguma coluna tenha sido habilitada a participar de filtros, através de predicados de consulta na definição da publicação, é possível especificar qual o valor dos predicados (colunas), para que assim apenas os registros onde as colunas que satisfaçam o valor informado sejam replicados. Dessa forma, por exemplo, se uma coluna do tipo unidade-federativa (UF) de uma tabela estiver apta na publicação, é possível, na subscrição, informar ao gerenciador de replicação que só se deseja receber as informações cuja UF seja "SP":

nome	cláusula where
@1 codevn	☐
lipcvn	☒
amtcvn	☒
nomcvn	☐
sglcvn	☐
dtaasicvn	☐
dtadtvcvn	☐

Figura 47 – Tela de Subscrições

No nosso exemplo, pode-se constatar que a tabela está subscrevendo a publicação do banco de dados "DVANGOGEH". Nesse caso, o banco de dados DFIDIAS, que está subscrevendo, está recebendo apenas os registros que tenham o predicado, isto é, a coluna *staacd* = "S". Em outras palavras, receberá apenas os registros da tabela CONVENIO que representem convênios acadêmicos (*staacd*).

A tela de Relacionamentos mostra os relacionamentos possíveis, por banco de dados, para uma tabela específica. Os relacionamentos possíveis são obtidos a partir dos relacionamentos do modelo conceitual global e das tabelas que realmente existem no banco de dados em questão. São analisadas também algumas regras, vistas anteriormente, como, por exemplo, a relação entre duas tabelas tipo *replicada*, que são consideradas como relacionamentos inválidos. Além disso, deve-se respeitar as estratégias de remoção, previstas para as outras tabelas relacionadas, e a estratégia de remoção da própria tabela.

Figura 48 – Tela de Relacionamentos

Após o informe do modelo físico de replicação da tabela e o cadastro dos relacionamentos físicos, que devem existir no banco de dados, existe uma tela responsável por analisar se as definições de replicação estão de acordo com a estratégia escolhida, se os relacionamentos estão todos cadastrados e se as colunas, regras e mensagens estão compatíveis. Essa tela é chamada de Consolidação dos Procedimentos de Distribuição.

Figura 49 – Tela de Consolidação dos Procedimentos de Distribuição

Porém, a tela anterior verifica o contexto individualizado de cada tabela, com foco nas definições de replicação. E com o decorrer do tempo, alterações nas estratégias de replicação das tabelas, novos relacionamentos e mudanças, na estratégia de remoção, podem trazer inconsistências nos dados e na própria replicação. Por essa razão, a construção de um verificador que analise todo o contexto do banco de dados "físico", verificando cada tabela, seus relacionamentos e estratégia de remoção, é imprescindível para que haja um acompanhamento das definições realizadas e para que a correção de algumas informações sejam efetuadas.

Este verificador foi implementado na tela de Consistência de Banco de Dados, que tem como saída um arquivo, informando se os relacionamentos estão corretos ou não, conforme os tipos das tabelas relacionadas e a estratégia de remoção definida.

Figura 50 – Tela de Consistência do Banco de Dados

É possível, a partir de um banco de dados, escolher as tabelas que se deseja consistir. Caso nenhuma tabela seja selecionada, todas as tabelas passarão pelo processo de consistência.

Além disso, é possível analisar somente os relacionamentos compatíveis com as tabelas existentes no banco de dados. Para isso, é preciso optar apenas pelo item *Relacionamentos* do banco de dados selecionado.

Um exemplo do relatório gerado pelo verificador pode ser encontrado no Apêndice C.

Apêndice B

B.1. Detalhes do metamodelo

Os objetivos das principais tabelas do metamodelo são:

- **BANCODEDADOS**

Armazenar o nome de todos os bancos de dados existentes.

- **CHECKCONSTR**

Manter as verificações associadas às tabelas para garantir a consistência dos dados.

- **COLUNA**

Conter informações sobre as colunas das tabelas.

- **CONEXAO**

Identificar as conexões dos servidores de replicação com os bancos de dados, ou seja, a conexão "adiciona" o banco de dados ao servidor de replicação.

- **DEPENDSTPROC**

Manter informações dos procedimentos que chamam outros procedimentos.

- **DIAGTABELA**

Manter os relacionamentos de todas as tabelas e os diagramas que as contêm.

- **ESQUEMADIAG**

Armazenar informações dos diagramas que mantêm os esquemas de dados. Esses diagramas vêm da ferramenta que administra o cadastro do modelo conceitual global.

- **ESTRATEGIA**

Cadastrar as estratégias de replicação que podem ser adotadas e também os parâmetros de controle para o Gerenciador de Replicação.

- **INDICE**

Armazenar os índices das tabelas.

- **MENSAGEM**

Manter as mensagens de erro, relativas às verificações, associadas às tabelas dos bancos de dados.

- MODULO

Identificar os módulos existentes para controle do ambiente replicado.

- MODULOTAB

Armazenar o vínculo entre as tabelas e os módulos, isto é, identificar quais tabelas pertencem aos módulos. Somente as tabelas identificadas com servidor e banco PROJETO/PROJETO são vinculadas ao módulo.

- REGRA

Guardar as regras associadas às colunas para garantir a consistência dos dados.

- RELACREGRA

Especificar as regras do relacionamento, no caso, de inconsistências entre a relação de tabelas com tipos diferentes. Contém a descrição da regra e a definição se o cadastro do relacionamento físico entre as tabelas deve existir ou não.

- RELACIONAMENTO

Identificar todos os relacionamentos entre tabelas.

- REPLICTABELA

Armazenar logicamente todas as replicações das tabelas. Mantém o relacionamento entre o cadastro da tabela lógica e o cadastro das tabelas físicas definido após a análise de replicação.

- SERVBANCODEDADOS

Mapear os bancos de dados existentes em um servidor de bancos de dados.

- SERVSGBD

Armazenar informações de todos os servidores de banco de dados.

- STOREDPROCEDURE

Cadastrar informações sobre os procedimentos existentes e à qual módulo pertencem.

- STPROCPARAM

Guardar os parâmetros de entrada e saída dos procedimentos.

- SUBSCRICAO

Manter as subscrições das tabelas.

- TABELA

Cadastrar informações relativas a todas as tabelas utilizadas pelos sistemas. As tabelas identificadas com servidor e banco PROJETO/PROJETO representam as tabelas "lógicas", vindas do modelo conceitual global.

- TABTRIGGER

Armazenar as *triggers* associadas a uma tabela.

- TIPODEDADO

Relacionar os tipos de dados permitidos pelo gerenciador de banco de dados utilizado no momento da extração do modelo conceitual global.

ESQUEMADIAG	
Physical Display	
coddgr	NUM
nomgrpdgr	VARC
nomdgr	VARC
timestamp	TIME

REGRA	
Physical Display	
nomrga	CHAR
staobtrga	CHAR
txtrga	VARC
dtascp	SMAL
dtaultalt	SMAL
timestamp	TIMES

COLUNA	
Physical Display	
nomsrvsbd	CHAR
nombcoddo	CHAR
nomtab	CHAR
nomcol	CHAR
tipcol	CHAR
tamcol	VARC
staobg	CHAR
virini	VARC
nomrga	CHAR
staobtrga	CHAR
dsccol	VARC
starpl	CHAR
staclw	CHAR
starplalt	CHAR
numordtab	TINYI
numordchaprm	TINYI
numordcharpl	TINYI
dtaultaltcol	SMAL
dtaultalttrpl	SMAL
timestamp	TIME

TIPODEDADO	
Physical Display	
tipcol	CHA
staexitam	CHA
timestamp	TIME

Apêndice C

C.1. Exemplo da Verificação no Ambiente USP

Para a execução desta verificação, foram selecionadas as tabelas contidas no Módulo M_2 , e que, no ambiente USP, pertencem ao banco de dados, que contém, em sua maioria, informações do Sistema de Graduação.

ANALISANDO O BANCO DE DADOS DFIDIAS localizado no servidor UBATUBA...

*** Processando a tabela ALUNOGR...

Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 5

Relacionamentos compatíveis conforme as tabelas existentes no banco: 4

Relacionamentos efetivamente cadastrados no banco: 4

- Analisando os relacionamentos possíveis para a tabela ALUNOGR...

ALUNOGR P -> HISTESCOLAREXTGR U (refhistescolarextgr1)

OK. A definição física do relacionamento 'refhistescolarextgr1' foi encontrada

ALUNOGR P -> MONITORIA U (refmonitoria2)

OK. A definição física do relacionamento 'refmonitoria2' foi encontrada

ALUNOGR P -> PROGRAMAGR P (refprogramagr1)

OK. A definição física do relacionamento 'refprogramagr1' foi encontrada

ALUNOGR P <- PESSOA R (refalunogr1)

OK. A definição física do relacionamento 'refalunogr1' foi encontrada,
conforme a estratégia de remoção '2' exige

- Analisando os relacionamentos lógicos para a tabela ALUNOGR...

ALUNOGR -> FAVORECIDO (reffavorecido7)

OK. A tabela 'FAVORECIDO' não existe neste banco de dados

*** Processando a tabela CURSOG R...

Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 4

Relacionamentos compatíveis conforme as tabelas existentes no banco: 4

Relacionamentos efetivamente cadastrados no banco: 3

- Analisando os relacionamentos possíveis para a tabela CURSOG R...

CURSOG R P <- COLEGIADO R (refcursogr1)

Erro: Relacionamento pode gerar inconsistência. Favor cadastrar estratégia de remoção!

A definição física do relacionamento 'refcursogr1' foi encontrada

CURSOG R P -> HABILITACAOGR P (refhabilitacaoogr1)

OK. A definição física do relacionamento 'refhabilitacaoogr1' foi encontrada

CURSOG R P -> VINCPESOAACURSOG R P (refvincpessoacursogr2)

OK. A definição física do relacionamento 'refvincpessoacursogr2' foi encontrada

CURSOG R P -> HISTCURSOG R P (trihistcursogr1)

Aviso: A definição física do relacionamento 'trihistcursogr1' não foi encontrada, mas o relacionamento será implementado via trigger

- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

*** Processando a tabela DISCIPLINAGR...

Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 13

Relacionamentos compatíveis conforme as tabelas existentes no banco: 13

Relacionamentos efetivamente cadastrados no banco: 13

- Analisando os relacionamentos possíveis para a tabela DISCIPLINAGR...

DISCIPLINAGR P -> DEPTORESP U (refdeptoresp1)

OK. A definição física do relacionamento 'refdeptoresp1' foi encontrada

DISCIPLINAGR P -> DISCIPGRCOMPL U (refdiscipgrcompl1)

OK. A definição física do relacionamento 'refdiscipgrcompl1' foi encontrada

DISCIPLINAGR P -> DISCIPHABIL U (refdisciphabil2)

OK. A definição física do relacionamento 'refdisciphabil2' foi encontrada

DISCIPLINAGR P -> EQUIVALENCIAGR P (refequivalenciagr1)

OK. A definição física do relacionamento 'refequivalenciagr1' foi encontrada

DISCIPLINAGR P -> ESPDISCIPGR U (refespdiscipgr1)

OK. A definição física do relacionamento 'refespdiscipgr1' foi encontrada

DISCIPLINAGR P -> GRADECURRICULAR P (refgradecurricular2)

OK. A definição física do relacionamento 'refgradecurricular2' foi encontrada

DISCIPLINAGR P -> GRUPOREQUISITO P (refgruporequisito2)

OK. A definição física do relacionamento 'refgruporequisito2' foi encontrada

DISCIPLINAGR P -> HABILIDADEDOC P (refhabilidadedoc2)

OK. A definição física do relacionamento 'refhabilidadedoc2' foi encontrada

DISCIPLINAGR P -> MANIFDISCIPGR U (refmanifdiscipgr1)

OK. A definição física do relacionamento 'refmanifdiscipgr1' foi encontrada

DISCIPLINAGR P -> MONITORIA U (refmonitorial)

OK. A definição física do relacionamento 'refmonitorial' foi encontrada

DISCIPLINAGR P -> REQUISITOG P (refrequisitogr2)

OK. A definição física do relacionamento 'refrequisitogr2' foi encontrada

DISCIPLINAGR P -> RESPDISCIPGR P (refrespdiscipgr2)

OK. A definição física do relacionamento 'refrespdiscipgr2' foi encontrada

DISCIPLINAGR P -> TURMAGR P (refturmagr1)

OK. A definição física do relacionamento 'refturmagr1' foi encontrada

- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

*** Processando a tabela HABILITACAOGR...

Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 15

Relacionamentos compatíveis conforme as tabelas existentes no banco: 15

Relacionamentos efetivamente cadastrados no banco: 13

- Analisando os relacionamentos possíveis para a tabela HABILITACAOGR...

HABILITACAOGR P <- CURSOG P (refhabilitacaoagr1)

OK. A definição física do relacionamento 'refhabilitacaoagr1' foi encontrada

HABILITACAOGR P -> CURRICULOGR P (refcurriculogr1)

OK. A definição física do relacionamento 'refcurriculogr1' foi encontrada

HABILITACAOGR P -> DISCIPHABIL U (refdisciphabil1)

OK. A definição física do relacionamento 'refdisciphabil1' foi encontrada

HABILITACAOGR P -> GRUPOREQUISITO P (refgruporequisito1)

OK. A definição física do relacionamento 'refgruporequisito1' foi encontrada

HABILITACAOGR P -> HABILAREACONHEC U (refhabilareaconhec1)

OK. A definição física do relacionamento 'refhabilareaconhec1' foi encontrada

HABILITACAOGR P -> HABILDEPEND P (refhabildepend1)

OK. A definição física do relacionamento 'refhabildepend1' foi encontrada

HABILITACAOGR P -> HABILDEPEND P (refhabildepend2)

OK. A definição física do relacionamento 'refhabildepend2' foi encontrada

HABILITACAOGR P -> HABILITATUALWEB U (refhabilitatualweb1)

OK. A definição física do relacionamento 'refhabilitatualweb1' foi encontrada

HABILITACAOGR P -> HABILITATUALWEB U (refhabilitatualweb3)

OK. A definição física do relacionamento 'refhabilitatualweb3' foi encontrada

HABILITACAOGR P -> HABILPROGGR P (refhabilproggr1)

OK. A definição física do relacionamento 'refhabilproggr1' foi encontrada

HABILITACAOGR P -> HABILTURMA P (refhabilturma2)

OK. A definição física do relacionamento 'refhabilturma2' foi encontrada

HABILITACAOGR P -> INSCRICAAGR U (refinscricaogr2)

OK. A definição física do relacionamento 'refinscricaogr2' foi encontrada

HABILITACAOGR P -> FUVEST U (trifuvest1)

Aviso: A definição física do relacionamento 'trifuvest1' não foi encontrada, mas o relacionamento será implementado via trigger

HABILITACAOGR P -> HISTHABILGR U (trihisthabilgr1)

Aviso: A definição física do relacionamento 'trihisthabilgr1' não foi encontrada, mas o relacionamento será implementado via trigger

HABILITACAOGR P <- TIPOHABILITGR U (refhabilitacaogr2)

OK. A definição física do relacionamento 'refhabilitacaogr2' foi encontrada

- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

*** Processando a tabela HABILPROGGR...

Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 3

Relacionamentos compatíveis conforme as tabelas existentes no banco: 3

Relacionamentos efetivamente cadastrados no banco: 3

- Analisando os relacionamentos possíveis para a tabela HABILPROGGR...

HABILPROGGR P <- HABILITACAOGR P (refhabilproggr1)

OK. A definição física do relacionamento 'refhabilproggr1' foi encontrada

HABILPROGGR P <- PROGRAMAGR P (refhabilproggr2)

OK. A definição física do relacionamento 'refhabilproggr2' foi encontrada

HABILPROGGR P <- TIPOENCERRAMENTOGR P (refhabilproggr3)

OK. A definição física do relacionamento 'refhabilproggr3' foi encontrada

- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

*** Processando a tabela HABILTURMA...

Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 2

Relacionamentos compatíveis conforme as tabelas existentes no banco: 2

Relacionamentos efetivamente cadastrados no banco: 2

- Analisando os relacionamentos possíveis para a tabela HABILTURMA...
HABILTURMA P <- HABILITACAOGR P (refhabilturma2)
OK. A definição física do relacionamento 'refhabilturma2' foi encontrada
HABILTURMA P <- TURMAGR P (refhabilturma1)
OK. A definição física do relacionamento 'refhabilturma1' foi encontrada
- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

*** Processando a tabela HISTESCOLARGR...
Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 9
Relacionamentos compatíveis conforme as tabelas existentes no banco: 9
Relacionamentos efetivamente cadastrados no banco: 8

- Analisando os relacionamentos possíveis para a tabela HISTESCOLARGR...
HISTESCOLARGR P -> APROVEITINTGR U (refaproveitintgr2)
OK. A definição física do relacionamento 'refaproveitintgr2' foi encontrada
HISTESCOLARGR P -> EQUIVEXTERNAGR U (refequivexternagr1)
OK. A definição física do relacionamento 'refequivexternagr1' foi encontrada
HISTESCOLARGR P -> ESTAGIOGR U (refestagiogr4)
OK. A definição física do relacionamento 'refestagiogr4' foi encontrada
HISTESCOLARGR P -> FREQPARCIALGR U (reffreqparcialgr1)
OK. A definição física do relacionamento 'reffreqparcialgr1' foi encontrada
HISTESCOLARGR P -> HISTESCSUBTUR U (refhistescsubtur1)
OK. A definição física do relacionamento 'refhistescsubtur1' foi encontrada
HISTESCOLARGR P -> NOTAPARCIALGR U (refnotaparcialgr1)
OK. A definição física do relacionamento 'refnotaparcialgr1' foi encontrada
HISTESCOLARGR P -> REQUERHISTESC U (refrequerhistesc2)
OK. A definição física do relacionamento 'refrequerhistesc2' foi encontrada
HISTESCOLARGR P <- PESSOA R (trihistescolargr3)
Erro: O relacionamento 'trihistescolargr3' não pode ser implementado via trigger, pois, a estratégia de remoção '2' exige a existência do relacionamento
HISTESCOLARGR P <- TURMAGR P (refhistescolargr2)
OK. A definição física do relacionamento 'refhistescolargr2' foi encontrada

- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

*** Processando a tabela HISTPROGGR...
Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 1
Relacionamentos compatíveis conforme as tabelas existentes no banco: 1
Relacionamentos efetivamente cadastrados no banco: 0

- Analisando os relacionamentos possíveis para a tabela HISTPROGGR...
HISTPROGGR P <- PROGRAMAGR P (trihistproggr1)
Aviso: A definição física do relacionamento 'trihistproggr1' não foi encontrada, mas o relacionamento será implementado via trigger
- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

*** Processando a tabela PESSOA...
 Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 81
 Relacionamentos compatíveis conforme as tabelas existentes no banco: 36
 Relacionamentos efetivamente cadastrados no banco: 9

- Analisando os relacionamentos possíveis para a tabela PESSOA...

PESSOA R <- ESTADO R (refpessoa4)
 A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.
 OK. A definição física do relacionamento 'refpessoa4' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R <- LOCALIDADE R (refpessoa3)
 A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.
 OK. A definição física do relacionamento 'refpessoa3' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R <- PAIS R (refpessoa2)
 A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.
 OK. A definição física do relacionamento 'refpessoa2' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> ALUNOGR P (refalunogr1)
 OK. A definição física do relacionamento 'refalunogr1' foi encontrada, conforme a estratégia de remoção '2' exige

PESSOA R -> ALUNOPOS R (refalunopos1)
 A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.
 OK. A definição física do relacionamento 'refalunopos1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> ANALISECURRICGR U (refanalisecurricgr2)
 OK. A definição física do relacionamento 'refanalisecurricgr2' foi encontrada, conforme a estratégia de remoção '2' exige

PESSOA R -> ANALISEDISCIPGR U (refanalisediscipgr3)
 OK. A definição física do relacionamento 'refanalisediscipgr3' foi encontrada, conforme a estratégia de remoção '2' exige

PESSOA R -> ANALISEREQUERGR U (refanaliserequergr3)
 OK. A definição física do relacionamento 'refanaliserequergr3' foi encontrada, conforme a estratégia de remoção '2' exige

PESSOA R -> CARTCOSEAS U (refcartcoseas1)
 OK. A definição física do relacionamento 'refcartcoseas1' foi encontrada, conforme a estratégia de remoção '2' exige

PESSOA R -> CONVENIO R (refconvenio4)
 A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.
 OK. A definição física do relacionamento 'refconvenio4' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> CONVENIO R (refconvenio5)
 A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.
 OK. A definição física do relacionamento 'refconvenio5' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> DOCENTE R (refdocentel)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'refdocente1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> EXTERNO R (refexterno1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'refexterno1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> HISTMEMBROCOLEG R (refhistmembrocoleg2)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'refhistmembrocoleg2' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> INSCRITOG R (refinscritogr1)

OK. A definição física do relacionamento 'refinscritogr1' foi encontrada, conforme a estratégia de remoção '2' exige

PESSOA R -> MEMBROCOLEG R (refmembrocoleg1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'refmembrocoleg1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> ORGANIZACAO R (reforganizacao5)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'reforganizacao5' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> ORGANIZACAO R (reforganizacao6)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'reforganizacao6' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> PESSCONVNUM R (refpessconvnum1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'refpessconvnum1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> REQUERIMENTOGR U (refrequerimentogr5)

OK. A definição física do relacionamento 'refrequerimentogr5' foi encontrada, conforme a estratégia de remoção '2' exige

PESSOA R -> SERVIDOR R (refservidor1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'refservidor1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> TITULOPES P (reftitulopes1)

OK. A definição física do relacionamento 'reftitulopes1' foi encontrada, conforme a estratégia de remoção '2' exige

PESSOA R -> USUARIO U (refusuario1)

OK. A definição física do relacionamento 'refusuario1' foi encontrada,

conforme a estratégia de remoção '2' exige

PESSOA R -> CATEGORIAHU R (tricategoriahu1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'tricategoriahu1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> COMPLPESSOA R (tricomplpeessoa1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'tricomplpeessoa1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> DEPENDENTE R (tridependente1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'tridependente1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> DEPENDENTE R (tridependente2)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'tridependente2' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> ENDPessoa R (triendpessoa1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'triendpessoa1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> FUVest U (trifuvest2)

Erro: O relacionamento 'trifuvest2' não pode ser implementado via trigger, pois, a estratégia de remoção '2' exige a existência do relacionamento

PESSOA R -> HISTESCOLARGR P (trihistescolargr3)

Erro: O relacionamento 'trihistescolargr3' não pode ser implementado via trigger, pois, a estratégia de remoção '2' exige a existência do relacionamento

PESSOA R -> HISTPESSOA R (trihistpeessoa1)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'trihistpeessoa1' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> PESSOA R (tripessoa5)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'tripessoa5' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> PESSOA R (tripessoa6)

A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'tripessoa6' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> TELEFPessoa R (tritelefpeessoal)
 A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'tritelefpeessoal' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

PESSOA R -> VINCULOPESSOA F (trivinculopessoal)
 Erro: O relacionamento 'trivinculopessoal' não pode ser implementado via trigger, pois, a estratégia de remoção '2' exige a existência do relacionamento

PESSOA R <- TIPODOCIDENT R (refpeessoal)
 A tabela 'replicada' não pode referenciar nenhuma tabela, pois, a consistência de seus dados é realizada na tabela onde os dados são atualizados.

OK. A definição física do relacionamento 'refpeessoal' não foi encontrada e o relacionamento entre as tabelas do tipo 'R' e 'R' não é permitido!

- Analisando os relacionamentos lógicos para a tabela PESSOA...

PESSOA -> ACOMPANHAMENTO (refacompanhamento2)
 OK. A tabela 'ACOMPANHAMENTO' não existe neste banco de dados

PESSOA -> AGINSCRICAO (refaginscricao1)
 OK. A tabela 'AGINSCRICAO' não existe neste banco de dados

PESSOA -> AGPROGRAMA (refagprograma2)
 OK. A tabela 'AGPROGRAMA' não existe neste banco de dados

PESSOA -> AGUARDO (refaguardo2)
 OK. A tabela 'AGUARDO' não existe neste banco de dados

PESSOA -> ANDADIPL (refandadipl4)
 OK. A tabela 'ANDADIPL' não existe neste banco de dados

PESSOA -> ANDADIPL (refandadipl5)
 OK. A tabela 'ANDADIPL' não existe neste banco de dados

PESSOA -> ANDADIPL (refandadipl6)
 OK. A tabela 'ANDADIPL' não existe neste banco de dados

PESSOA -> ANDAMENTO (refandamento4)
 OK. A tabela 'ANDAMENTO' não existe neste banco de dados

PESSOA -> ANDAMENTO (refandamento5)
 OK. A tabela 'ANDAMENTO' não existe neste banco de dados

PESSOA -> ANDAMENTO (refandamento6)
 OK. A tabela 'ANDAMENTO' não existe neste banco de dados

PESSOA -> AUTONOMO (refautonomo1)
 OK. A tabela 'AUTONOMO' não existe neste banco de dados

PESSOA -> ESTAGIARIO (refestagiario1)
 OK. A tabela 'ESTAGIARIO' não existe neste banco de dados

PESSOA -> ESTAGIARIOPOS (refestagiariopos1)
 OK. A tabela 'ESTAGIARIOPOS' não existe neste banco de dados

PESSOA -> HISTBOLSAPES (refhistbolsapes2)
 OK. A tabela 'HISTBOLSAPES' não existe neste banco de dados

PESSOA -> HISTMEMBROASSOC (refhistmembroassoc2)
 OK. A tabela 'HISTMEMBROASSOC' não existe neste banco de dados

PESSOA -> HISTPES (refhistpes1)
 OK. A tabela 'HISTPES' não existe neste banco de dados

PESSOA -> INTERESSADO (refinteressado3)
 OK. A tabela 'INTERESSADO' não existe neste banco de dados

PESSOA -> MEMBROASSOC (refmembroassoc2)
 OK. A tabela 'MEMBROASSOC' não existe neste banco de dados

PESSOA -> OCORRENCIAPROC (refocorrenciapro4)
 OK. A tabela 'OCORRENCIAPROC' não existe neste banco de dados

PESSOA -> ORIENTPES (reforientpes1)
 OK. A tabela 'ORIENTPES' não existe neste banco de dados

PESSOA -> PAGAMENTO (refpagamento6)

```

OK.      A tabela 'PAGAMENTO' não existe neste banco de dados
PESSOA -> PRODUCAOPES (refproducaopes2)
OK.      A tabela 'PRODUCAOPES' não existe neste banco de dados
PESSOA -> PUBLICACAOPES (refpublicacaopes1)
OK.      A tabela 'PUBLICACAOPES' não existe neste banco de dados
PESSOA -> R41PGMMATTUR (refr41pgmmattur3)
OK.      A tabela 'R41PGMMATTUR' não existe neste banco de dados
PESSOA -> R67ESPPRETUROFEARE (refr67esppreturofeare2)
OK.      A tabela 'R67ESPPRETUROFEARE' não existe neste banco de dados
PESSOA -> REGISTROA18 (refregistroa185)
OK.      A tabela 'REGISTROA18' não existe neste banco de dados
PESSOA -> REGISTROA18 (refregistroa186)
OK.      A tabela 'REGISTROA18' não existe neste banco de dados
PESSOA -> REGISTROA18 (refregistroa187)
OK.      A tabela 'REGISTROA18' não existe neste banco de dados
PESSOA -> REQUISICAOPROC (refrequisicaoproc2)
OK.      A tabela 'REQUISICAOPROC' não existe neste banco de dados
PESSOA -> RESPCONTABANC (refrespcontabanc1)
OK.      A tabela 'RESPCONTABANC' não existe neste banco de dados
PESSOA -> SERVIDORCONV (refservidorconv1)
OK.      A tabela 'SERVIDORCONV' não existe neste banco de dados
PESSOA -> TESEPES (refttesepes1)
OK.      A tabela 'TESEPES' não existe neste banco de dados
PESSOA -> USUARIOPROTEOS (refusuarioproteos2)
OK.      A tabela 'USUARIOPROTEOS' não existe neste banco de dados
PESSOA -> ALUNOEAD (trialunoead1)
OK.      A tabela 'ALUNOEAD' não existe neste banco de dados
PESSOA -> ALUNOPD (trialunopd1)
OK.      A tabela 'ALUNOPD' não existe neste banco de dados
PESSOA -> ASPIRACAO (triaspiracao3)
OK.      A tabela 'ASPIRACAO' não existe neste banco de dados
PESSOA -> CANDIDATO (tricandidato1)
OK.      A tabela 'CANDIDATO' não existe neste banco de dados
PESSOA -> COMPLPESSOASERV (tricomplpeessoaserv1)
OK.      A tabela 'COMPLPESSOASERV' não existe neste banco de dados
PESSOA -> EVENTOPES (trieventopes1)
OK.      A tabela 'EVENTOPES' não existe neste banco de dados
PESSOA -> IDIOMAPES (triidiomapes1)
OK.      A tabela 'IDIOMAPES' não existe neste banco de dados
PESSOA -> PESSOAFORNEC (tripessoafornecl)
OK.      A tabela 'PESSOAFORNEC' não existe neste banco de dados
PESSOA -> PREMIOPESES (tripremiopeles1)
OK.      A tabela 'PREMIOPESES' não existe neste banco de dados
PESSOA -> RESUMOQUALIFICPES (triresumoqualificpes1)
OK.      A tabela 'RESUMOQUALIFICPES' não existe neste banco de dados
PESSOA -> SERVIDORF (triservidorf1)
OK.      A tabela 'SERVIDORF' não existe neste banco de dados

```

*** Processando a tabela PROGRAMAGR...

Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 12

Relacionamentos compatíveis conforme as tabelas existentes no banco: 12

Relacionamentos efetivamente cadastrados no banco: 10

- Analisando os relacionamentos possíveis para a tabela PROGRAMAGR...

```
PROGRAMAGR P <- ALUNOGR P (refprogramagr1)
```

```
OK.      A definição física do relacionamento 'refprogramagr1' foi encontrada
```

```
PROGRAMAGR P <- CONVENIO R (refprogramagr4)
```

```
Erro:    Relacionamento pode gerar inconsistência. Favor cadastrar estratégia de remoção!
```

```
A definição física do relacionamento 'refprogramagr4' foi encontrada
```

```
PROGRAMAGR P <- DOCENTE R (refprogramagr7)
```

```

    Erro: Relacionamento pode gerar inconsistência. Favor cadastrar
          estratégia de remoção!
          A definição física do relacionamento 'refprogramagr7' foi
          encontrada
PROGRAMAGR P -> ESTAGIOGR U (refestagiogr3)
    OK. A definição física do relacionamento 'refestagiogr3' foi
    encontrada
PROGRAMAGR P -> HABILPROGGR P (refhabilproggr2)
    OK. A definição física do relacionamento 'refhabilproggr2' foi
    encontrada
PROGRAMAGR P -> NOTASINGRESSOGR P (refnotasingressogr1)
    OK. A definição física do relacionamento 'refnotasingressogr1'
    foi encontrada
PROGRAMAGR P -> PROGRAMAGR P (refprogramagr5)
    Erro: A definição física do relacionamento 'refprogramagr5' não
    foi encontrada
PROGRAMAGR P -> REQUERIMENTOGR U (refrequerimentogr1)
    OK. A definição física do relacionamento 'refrequerimentogr1'
    foi encontrada
PROGRAMAGR P -> TRANSFEXTERNA U (reftransfexternal1)
    OK. A definição física do relacionamento 'reftransfexternal1' foi
    encontrada
PROGRAMAGR P -> HISTPROGGR P (trihistproggr1)
    Aviso: A definição física do relacionamento 'trihistproggr1' não
    foi encontrada, mas o relacionamento será implementado via
    trigger
PROGRAMAGR P <- TIPOENCERRAMENTOGR P (refprogramagr2)
    OK. A definição física do relacionamento 'refprogramagr2' foi
    encontrada
PROGRAMAGR P <- TIPOINGRESSOGR P (refprogramagr3)
    OK. A definição física do relacionamento 'refprogramagr3' foi
    encontrada

```

- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

*** Processando a tabela TURMAGR...

Relacionamentos lógicos em que a tabela está envolvida no Modelo Global: 10

Relacionamentos compatíveis conforme as tabelas existentes no banco: 10

Relacionamentos efetivamente cadastrados no banco: 10

- Analisando os relacionamentos possíveis para a tabela TURMAGR...

```

TURMAGR P <- DISCIPLINAGR P (refturmagr1)
    OK. A definição física do relacionamento 'refturmagr1' foi
    encontrada
TURMAGR P <- TIPOTURMAGR P (refturmagr2)
    OK. A definição física do relacionamento 'refturmagr2' foi
    encontrada
TURMAGR P -> ATIVIDADEDOC U (refatividadedoc3)
    OK. A definição física do relacionamento 'refatividadedoc3' foi
    encontrada
TURMAGR P -> HABILTURMA P (refhabilturma1)
    OK. A definição física do relacionamento 'refhabilturma1' foi
    encontrada
TURMAGR P -> HISTESCOLARGR P (refhistescolargr2)
    OK. A definição física do relacionamento 'refhistescolargr2' foi
    encontrada
TURMAGR P -> INSCRESPGR P (refinscrespgr3)
    OK. A definição física do relacionamento 'refinscrespgr3' foi
    encontrada
TURMAGR P -> OCUPTURMA P (refocupturma1)
    OK. A definição física do relacionamento 'refocupturma1' foi
    encontrada
TURMAGR P -> SUBTURMAGR U (refsubturmagr2)
    OK. A definição física do relacionamento 'refsubturmagr2' foi

```

```
encontrada
TURMAGR P -> TURPRATICA P (refturpratical)
OK.      A definição física do relacionamento 'refturpratical' foi
encontrada
TURMAGR P -> TURPRATICA P (refturpratica2)
OK.      A definição física do relacionamento 'refturpratica2' foi
encontrada
```

- Todos os relacionamentos lógicos já foram analisados nos relacionamentos do banco!

Total de tabelas analisadas: 11

Bibliografia

- [1] A. P. Sheth and J. A. Larson. **Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases**, ACM Computing Surveys, 22:3, 183-236, September 1990.
- [2] A. Savasere, S. B. Navathe et al. **On Applying Classification to Schema Integration**, Proc. First Int. Workshop on Interoperability in Multidatabase Systems, IEEE, Japan, 8-9, April 1991.
- [3] Abdelsalam A. Helal, Abdelsalam A. Heddaya and Bharat B. Bhargava. **Replication Techniques in Distributed Systems**, Kluwer Academic Publishers, 1996.
- [4] Abraham Silberschatz, Henry F. Korth and S. Sudarshan. **Database Systems Concepts**, 3rd ed., McGraw-Hill, 1997.
- [5] C. J. Date. **Banco de Dados: Tópicos Avançados**, Campus, 1988.
- [6] E. J. S. Mesquita and M. Finger. **Projeto de Dados em Bancos de Dados Distribuídos**, 13^o Simpósio Brasileiro de Banco de Dados, Maringá-PR-Brasil, 87-101, October 1998.
- [7] H. Sheck and M.H. Shcoll. **Evolution of Data Models**, Lectures Notes in Computer Science, 466, 135-153, 1990.
- [8] H. T. Kung and J. T. Robinson. **On optimistic methods for concurrency control**, ACM Transactions on Database Systems, 6:2, 213-226, 1981.
- [9] J. M. Smith and D. C. P. Smith. **Database Abstractions: Aggregation and Generalization**, ACM Transactions on Database Systems, 2:2, 105-133, June 1977.
- [10] João E. Ferreira and Gisele Busichia. **Database Modularization Design for the Construction of Flexible Information System**, Proceedings of the IEEE International Database Engineering and Application Symposium, 442-452, 1999.
- [11] João E. Ferreira and Marcelo Finger. **Controle de Concorrência e Distribuição de Dados: a teoria clássica, suas limitações e extensões modernas**, 12^a Escola de Computação, IME-USP, 2000.
- [12] M. A. Casanova and A. V. Moura. **Princípios de Sistemas de Gerência de Banco de Dados Distribuídos**, 4^a Escola de Computação, IME-USP, 1984.

- [13] M. Stonebraker, P. M. Aoki, W. Litwin et al.. **Mariposa: a Wide-Area Distributed Database System**, The VLDB Journal, 5:1, 48-63, 1996.
- [14] M. T. Özsu and P. Valduriez. **Principles of Distributed Database Systems**, 2nd ed., Prentice Hall, 1999.
- [15] M. T. Özsu and P. Valduriez. **Distributed Database Systems: Where Are We Now?**, IEEE Computer, 24:8, 68-78, August 1991.
- [16] Marcelo Brito dos Santos and Markus Endler. **Replicação de Objetos com Consistência Relaxada**, Projeto Sidam, IME-USP, October 2000.
- [17] P. A. Bernstein, V. Hadzilacos and N. Goodman. **Concurrency Control and Recovery in Database Systems**, Addison-Wesley, 1987.
- [18] P. Feldman and D. Miller. **Entity Model Clustering: Structuring a Data Model by Abstraction**, The Computer Journal, 29:4, 348-360, 1986.
- [19] R. A. Golding and D. D. E. Long. **Design Choices for Weak-Consistency Group Communication**, UCSC – TR 92:45, University of California, October 1992.
- [20] R. A. Golding. **Weak-Consistency Group Communication and Membership**, Ph. D. Thesis, University of California, December 1992.
- [21] R. Alonso, H. Garcia Molina and D. Barbara, **Quasi-Copies: Efficient Data Sharing for Information Retrieval Systems**, EDBT, 443-468, March 1988.
- [22] R. Elmasri and S. B. Navathe. **Fundamentals of Database Systems**, 3rd ed., Addison-Wesley, 2000.
- [23] R. H. Thomas. **A majority consensus approach to concurrency control for multiple copy databases**, ACM Transactions on Database Systems, 4:2, 180-209, 1979.
- [24] R. M. Adler. **Distributed Coordination Models for Client/Server Computing**, Proceedings of the IEEE, 14-22, 1995.
- [25] R. S. Pressman. **Software Engineering: a Practitioner's Approach**, 3rd ed., McGraw-Hill, 1992.
- [26] Richard Lenz. **The “Virtual-Primary-Copy Approach” Compared to Other Approaches with Weak Consistent Data Replication**, University of Erlangen-Nuremberg, 1996.

- [27] S. B. Navathe. **Evolution of Data Modeling for Databases**, Communications of the ACM, 35:9, 112-123, September 1992.
- [28] S. Ceri and G. Pelagatti. **Distributed Databases: Principles and Systems**, McGraw-Hill, 1984.
- [29] S. Ceri, B. Pernici, and G. Wiederhold. **Distributed Database Design Methodologies**, Proceedings of the IEEE, 75:5, 533-546, May 1987.
- [30] S. Prabhakar, J. Srivastava, S.B. Navathe et al. **Federated Autonomous Databases: Project Overview**, Proc. Int. Conf. Interoperability in Multidatabase Systems (IMS'93), Austria, 19-20, April 1993.
- [31] Sybase Advanced Application Design using Replication Server: Student Guide.
- [32] Sybase Fast Track to Replication Server Administration: Student Guide.
- [33] Sybase Replication Server: "*A Practical Architecture for Distributing and Sharing Corporate Information*" available at <http://www.sybase.com>.