UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
(Bacharelado)

PROTÓTIPO DE SOFTWARE PARA CRIAÇÃO E GERENCIAMENTO DE REPLICAÇÃO EM BANCO DE DADOS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS DE DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO - BACHARELADO

MARCELO RODRIGO FACCIN FURTADO

BLUMENAU, NOVEMBRO DE 1999.

1999/2-26

PROTÓTIPO PARA CRIAÇÃO E GERENCIAMENTO DE BANCO DE DADOS GENÉRICO DISTRIBUÍDO

MARCELO RODRIGO FACCIN FURTADO

ESTE TRABALHO DE CONCLUSÃO DE CURSO FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS DA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIO PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

Prof. Marcel Hugo - ORIENTADOR

Prof. José Roque Votolini - COORDENADOR

BANCA EXAMINADORA

Prof. Marcel Hugo

Prof. Sérgio Estringari

Prof. Oscar Dalfovo

Dedicatória

Dedico este trabalho aos meus pais por terem depositado confiança e me apoiado em todos os momentos de minha trajetória.

Agradecimentos

Agradeço aos meus amigos Anísio Iahn e Adriano Pessini por terem me apoiado em nosso trabalho para que eu consegui-se desenvolver este projeto.

Ao professor Marcel Hugo, pelo incentivo, orientação e atenção dispensada durante todo o desenvolvimento do trabalho.

A todos os professores do curso que me mostraram o caminho para o conhecimento.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

Lista de Abreviaturas

Neste item será apresentado o significado das abreviaturas utilizadas neste trabalho

- ODBC: *Open DataBase Connectivity*;
- DBE: *Borland DataBase Engine*;
- DBA: Administrador de Banco de Dados;
- BD: Banco de Dados;
- BDD: Banco de Dados distribuídos;
- SGBD: Sistema Gerenciador de Banco de Dados;
- SGBDD: Sistema Gerenciador de Banco de Dados Distribuído;
- DRDA: Arquitetura de Banco de Dados Relacional distribuído;
- 2PC: *Two-phase Commit*;
- ACID: Atômico, Consistente, Isolado e Durável;
- CPU: Unidade Central de Processamento;
- I/O: *IN e OUT*;
- API: *Application Programming Interface*;
- SQL: *Structured Query Language*;
- ANSI: *American National Standard Institute*;
- SEQUEL: *Structured English Query Language*;
- ISSO: *International Organization for Standardization* ;
- MER: Modelo Entidade Relacionamento;
- SGA: *System Global Area*;

Resumo

Com as novas descobertas tecnológicas para comunicação de dados, já é possível se trabalhar com bancos de dados não mais centralizados em um servidor, mas distribuídos em diversos servidores, que podem estar localizados em uma rede local ou em pontos diferentes do planeta, compartilhando dados de forma transparente aos usuários. Este protótipo tem por objetivo descrever e implementar as principais características de bancos de dados distribuídos (BDD), apresentando os requisitos de um Sistema de Gerência de Bancos de Dados Distribuídos (SGBDD), as vantagens e desvantagens na sua utilização, considerações sobre o desenvolvimento de bancos de dados distribuídos e ainda, sobre o processamento de consultas e transações em bancos de dados deste tipo.

Abstract

With the new technologies for data communication, it is possible to work with databases not centered in a server, but distributed in some servers, that can be located at the same network, as well as at different countries, sharing information in a transparent way for the users. This prototype has the objective of describe the features of distributed databases (DDB), presenting the requirements of a distributed databases management system (DDBMS), the advantages and disadvantages of its use, considerations about the development of distributed databases and still, about the queries and transaction processing in this kind of databases.

1 INTRODUÇÃO

As organizações precisam distribuir seus dados em vários pontos de forma a facilitar suas atividades administrativas, permitindo que cada setor mesmo que distante geograficamente mantenha controle de seus próprios dados e ofereça um compartilhamento global no uso destes por outros setores da organização. Esses entre outros aspectos fazem com que o campo da pesquisa e desenvolvimento de banco de dados continue a crescer.

Um dos pontos fortes das redes corporativas é o compartilhamento de dados, o que traz grande performance no deslocamento de informações dentro da corporação. Embora exista este recurso de redes corporativas na grande parte das corporações, cada empresa possui a sua própria base de dados independente, segundo [KOR95]. A falta de integração entre as base de dados reflete diretamente no setor administrativo, inviabilizando uma visão concentrada de todas as informações da empresa. Este é um dos grande motivos porque a tecnologia de banco de dados distribuídos está crescendo dia-a-dia, a necessidade de se ter o maior e mais rápido número de informações possíveis para que se tome uma decisão rápida e precisa.

Com um ambiente de múltiplos bancos de dados independentes e heterogêneos, as aplicações que foram escritas para utilizar a base de dados não estarão integradas entre as filiais da corporação, ou seja, cada aplicação será executada em uma das filiais da corporação independente das informações existentes nas outras bases de dados.

Conforme [COU84] sem a existência de um ambiente distribuído as consultas aos dados que estão nas outras base de dados tornam-se extremamente ineficazes, porque necessitará um contato prévio com as outras partes da corporação para discutir a maneira de transportar o dado de uma empresa para outra. Esta forma de acesso aos dados é lenta e em alguns casos não obtém-se o resultado esperado. Além disso quando as informações chegam no seu destino elas estão engessadas, sem a possibilidade de analisá-las de um modo diferente.

Uma das desvantagens encontradas em um ambiente de banco de dados centralizados é que em caso do banco ou o servidor ter qualquer problema, todos os usuários que utilizam

a base de dados ficarão parados até que seja sanado o problema. Visando solucionar os problemas citados, entre outros, desenvolveu-se uma ferramenta de construção e gerenciamento de Banco de Dados Distribuídos. Com o grande crescimento das redes corporativas torna-se muito viável a corporatividade das informações de uma empresa que se localizam fisicamente separadas, devido a rápida disponibilização de todas as informações para todos os usuários da rede corporativa independente de sua localização.

Para implementação deste trabalho foi utilizado o ambiente visual DELPHI e como Banco de Dados o ORACLE 8, Microsoft SQL Server 7.0 e o Sybase Anywhere 5.0 . O método de especificação que é utilizado para a ferramenta proposta é análise estruturada. A técnica utilizada para replicação dos dados é a replicação assimétrica e o modelo *Pear-to-Pear*.

1.1 OBJETIVOS

Este trabalho tem por objetivo principal a especificação e implementação de uma ferramenta de criação e gerenciamento de ambiente distribuído em Banco de Dados genérico que seja capaz de replicar informações entre banco de dados heterogêneo.

1.2 ORGANIZAÇÃO

A seguir serão descritos brevemente cada capítulo do trabalho.

O capítulo de introdução apresenta uma visão geral deste trabalho, sua relevância, objetivos e organização do trabalho.

O segundo capítulo apresenta a fundamentação teórica.

O terceiro capítulo apresenta o desenvolvimento do protótipo.

O quarto capítulo apresenta o funcionamento do software a nível de usuário.

O quinto capítulo apresenta as conclusões e sugestões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentada uma visão conceitual sobre os assuntos que cercam e integram os sistemas de banco de dados distribuídos.

2.1 CONCEITOS BÁSICOS

Os conceitos estão dispostos de forma a representar a evolução da tecnologia de sistemas de banco de dados distribuídos.

Conforme [REN94] e [VAS95] uma arquitetura cliente / servidor é uma abordagem da computação que separa os processos em plataformas independentes que interagem , permitindo que os recursos sejam compartilhados enquanto se obtém o máximo de benefício de cada dispositivo diferente. É basicamente uma forma de computação distribuída ou em rede. Há outros paradigmas para a interação de processos concorrentes, mas estão além do escopo desta proposta.

Banco de dados segundo [CER95] e [SET89] é o arquivo físico, em dispositivos periféricos, onde estão armazenados os dados de diversos sistemas, para consulta e atualização pelo usuário.

Conforme [CHU83] e [DAT85] dado é o valor do campo quando é armazenado no Banco de Dados e informação é o valor que este campo representa para as atividades da empresa.

S.G.D.B. (Sistema Gerenciador de Banco de Dados) é o software responsável pelo gerenciamento (armazenamento e recuperação) dos dados no Banco de Dados segundo [FUR83].

Segundo [KOR95] e [SET89] um banco de dados distribuído consiste em uma coleção de nós, e cada um destes nós mantém um banco de dados localmente.

Segundo [DAT88], [COU84], [LEL78] e [KIN81] um sistema distribuído é qualquer sistema que envolve múltiplas localidades conectadas juntas em uma espécie de redes de comunicações, nas quais o usuário de qualquer localidade pode acessar os dados armazenados em outro local. Cada localidade, por sua vez pode geralmente ser considerada como um sistema de banco de dados em si, que tem seu próprio banco de dados e sua própria função

gerenciadora, seus próprios terminais e usuários, o seu próprio armazenamento local e o seu próprio processamento, rodando o próprio SGBD local.

Segundo [CAS85], a finalidade do sistema de banco de dados distribuído é realizar transações. A transação é uma unidade de trabalho. Ela consiste na execução de uma sequência de operações especificadas pela aplicação, começando com uma operação especial BEGIN TRANSACTION, e terminando ou com uma operação COMMIT ou com uma operação ROLLBACK. COMMIT é utilizado para sinalizar o término bem-sucedido (a unidade de trabalho foi completada com sucesso); ROLLBACK é usado para sinalizar o término mal-sucedido (a unidade de trabalho não pode ser completada com sucesso por haver ocorrido alguma situação excepcional – por exemplo, um registro necessário não pode ser localizado). Término aqui refere-se ao término da transação, e não necessariamente ao término do programa; uma execução de programa poderá corresponder a uma sequência de diversas transações, uma após a outra.

2.2 CLIENTE/SERVIDOR

O sistema cliente/servidor é um paradigma lógico para o processamento distribuído, sendo que o cliente e o servidor podem estar ou não em uma mesma máquina física. Quando os processos cliente e servidor são executados em máquinas diferentes, estes são conectados através de redes locais ou remotas, sendo que as redes locais são as implementações mais comuns de cliente/servidor. O usuário do sistema interage com um cliente, que por sua vez emite pedidos e recebe resultados do servidor.

Há quatro tipos de processos distribuídos: filtros, clientes, servidores e peers.

Os filtros realizam uma operação fixa no fluxo de dados, passando adiante os resultados para outro processo; os peers (não hierárquicos) são idênticos um ao outro, e interagem de forma cooperativa para realizar um trabalho útil. Desta forma, “cliente/servidor” e “não hierárquico” (peer), não são sinônimos, pois em um ambiente de comunicações peer-to-peer (não hierárquico) duas entidades comunicam-se em termos iguais e em um processamento cliente/servidor os processos cliente enviam pedidos a um processo servidor, que responde com resultados para esses pedidos. O servidor, na verdade, oferece

serviços aos seus clientes, por um meio de processamento específico e a interação entre os processos cliente e servidor é uma troca cooperativa, transacional, em que o cliente é ativo e o servidor é reativo. Sendo esta, a principal distinção entre cliente/servidor e outros paradigmas menos restritos. Mais informações em [REN94] e [VAS95].

2.3 BANCO DE DADOS DISTRIBUÍDO

Este item abordará os tópicos mais relevantes aos sistemas de banco de dados distribuídos e banco de dados distribuídos.

2.3.1 HISTÓRICO DO SGBD DISTRIBUÍDO

Os conceitos SGBDD (Sistema Gerenciador de Banco de Dados distribuídos) foram criados por volta dos anos 70 no projeto de pesquisa R*Star da IBM. A IBM levou 10 anos desenvolvendo tecnologia de SGBDD até chegar no DRDA (arquitetura de banco de dados relacional Distribuído). Esta arquitetura foi criada para integrar o banco de dados DB2 em suas diferentes versões como DB2 para os sistemas operacionais AIX, OS/2, OS/400, VM e MVS, segundo [KOR95].

Conforme [KOR95] o primeiro SGBDD lançado foi o INGRES/Star em 1987, seguido pela ORACLE que anunciou as potencialidades de seu produto apenas para não perder mercado na ocasião. O primeiro produto da ORACLE para suportar razoavelmente o processamento distribuído foi o ORACLE 7 que esta no mercado desde 1993.

2.3.2 SISTEMAS GERENCIADORES DE BANCO DE DADOS DISTRIBUÍDOS

Um SGBDD é um software que geralmente executa em um servidor de uma rede remota que tem a finalidade de disponibilizar dados de uma forma controlada às aplicações que rodam nas estações clientes. Normalmente usa a linguagem SQL para o acesso a esses dados. Segundo [KOR95] é aconselhável que o SGBDD esteja instalado em um servidor diferente do Servidor de Arquivos da empresa, por ele ter características e exigências diferentes dos softwares Servidores de Arquivos, além de não dividir processamento com o mesmo. Entre as características necessárias para que se tenha um bom SGBDD são as seguintes: Performance, Integridade de dados, Recursos de *Stored Procedures* e *Triggers*, *Backup On-Line*, Replicação Controlada, Multiplataforma, Heterogêneos, Facilidade de

manutenção e que ofereça uma alta segurança no acesso aos dados. Todas essas características são necessárias devido esta arquitetura de ambiente ser muito complexa.

Segundo [DAT88]e [COU84] no começo da utilização das redes locais, os sistemas informatizados eram executados nas estações clientes e os dados ficavam localizados fisicamente nos servidores. Mas, mesmo com o acesso sendo multiusuário (acesso de mais de um usuários à aplicações ao mesmo tempo), todas os processamentos de acesso aos dados eram realizados nas estações clientes, ficando para o servidor o papel de guardar os dados. Com o advento da tecnologia de banco de dados distribuído, o processamento e manipulação e acesso ao dados passam a ser feitos em vários servidores, ficando a estação cliente apenas com o processamento das aplicações. Com isso, as aplicações podem ficar mais rápidas e a rede local “desafoga” o seu tráfego de dados, beneficiando a todos da rede.

Os SGBDs são usados principalmente em aplicações de missão crítica como bancos, indústrias, grandes grupos comerciais, etc. Mas, as médias e pequenas empresas estão começando a adotá-lo por causa do seu baixo custo e facilidade de implementação. É uma tendência mundial que as aplicações migrem de um ambiente mono-usuário e de rede e passe para ambientes cliente/servidor, devido principalmente a grande popularização da Internet.

As redes se tornam economicamente mais convenientes à medida que cai o custo das máquinas, embora o custo e as dificuldades de transmissão possam ser também fatores limitantes.

O critério de localização dos dados é influenciado pela estrutura de controle adotada na rede. Na estrutura hierárquica (sistema hierarquicamente distribuído), os computadores executam tarefas que interagem de modo mais ou menos estruturado e controlado pêlos membros de mais alto nível na hierarquia. Na estrutura simétrica (sistema simetricamente distribuído), todos os computadores cooperam a um mesmo nível lógico, embora relações Cliente/Servidor possa ser criadas dinamicamente na execução de alguma tarefa.

Segundo [KOR95] um problema difícil é manter a consistência entre as cópias quando ocorre uma atualização em um nodo contendo uma das cópias . A atualização deve atuar corretamente, independentemente da velocidade de transmissão dos dados; o nodo que vai empreender a atualização deve notificar a todos os demais que contenham cópias sua intenção

de atualizar o dado, e, se houver algum conflito, este deve ser resolvido nesse momento (antes portanto das fases de bloqueio e da atualização propriamente dita); com isso se procura inclusive evitar impasses durante os conflitos de dados.

A área de bancos de dados distribuídos ainda necessita de muita pesquisa em virtude de sua novidade e complexidade. Problemas como os de manter integridade, consistência e segurança se agravam; a recuperação após um erro se complica se atinge diversos nodos. Além disso, surgem problemas novos como os envolvidos na execução distribuída de transações, conversão de dados, sincronização de eventos nos nodos, e escolha de estratégias "globais " que otimizam a manipulação levando em conta maquinas e sistemas heterogêneos.

2.3.3 ARQUITETURA DE BANCO DE DADOS DISTRIBUÍDO

Instrumentos de SGBD distribuídos clássicos baseiam-se em um mecanismo chamado *two-phase commit* (2CP). Este mecanismo une todas as atualizações para todas as localizações participantes em uma atualização. Na prática, segundo [DAT88], ele realiza um bloqueio sincronizado em todas as partes de uma transação. A integridade dos dados é preservada através de diversas e diferentes máquinas e localizações.

A fim de aumentar a confiabilidade de uma implementação, segundo o modelo de [DAT88], a ferramenta responsável pela replicação (cópia dos dados de um banco de dados para outros) deve ter um conceito central de transação ACID isto significa:

1. Atômico: A transação vai completar de forma bem sucedida em todos os nodos participantes do sistema distribuído, ou em nenhum.

2. Consistente: A transação sempre produzirá os mesmos resultados se aplicada mais de uma vez (isto é, deve ser consistentemente reproduzível) .

3. Isolado: A visão dos dados através de todo o banco de dados distribuído no momento da transação deve estar protegido de mudanças até que a transação tenha atualizados todos os nós.

4. Durável: Uma vez encerrado normalmente, os dados atualizados ou adicionados através da transação se tornam persistentes.

Segundo [DAT88] e [LEL78] o ponto fraco desta arquitetura está na escalabilidade limitada e em impor rígida posição de integridade de dados mesmo quando isto não é pedido. Num contexto de banco de dados distribuído uma transação é basicamente, pelo menos, atômica (senão também consistente, isolada e durável). Sendo atômica, ela ou atualiza de forma bem sucedida todos os nós ou nenhum deles. Se um nó está sofrendo com problemas de rede, então a transação não terá sucesso em nenhum dos nós restantes. Também deve ser considerado o tráfego de rede associado a uma atualização que se espalha por centenas de máquinas localizadas em diferentes lugares. Se uma transação deve ser protegida em centenas de nodos, o tempo de resposta do sistema logo se tornará frustrante para os usuários.

2.3.4 TEMPO DE REPLICAÇÃO EM UM AMBIENTE DISTRIBUÍDO

O tempo de replicação deve fornecer diversas alternativas de atualização conforme [COU84], [KOR95] e [DAT88]:

a) imediato: Logo que possível. Neste caso os dados são movidos através de filas e servidores de replicação o mais rápido possível.

b) programado: como determinado pelo seu administrador de sistema. Neste caso os dados permanecem no servidor de replicação até que esteja programado para distribuição.

c) desencadeado: Por critério definido pelo usuário, como em um evento acontecendo, o número de registros excedendo um limite ou hora do dia. Quando a *trigger* é disparada, o servidor move os dados para fila de distribuição para processamento remoto.

d) sob controle manual, ou seja, o usuário irá interagir em um determinado momento por meio de aplicações para replicar as informações.

Em uma situação com 100 nodos de banco de dados, somente 90 dos quais estão disponíveis, um sistema de banco de dados distribuído tentaria realizar uma transação em todos os nodos. Mesmo se todos os 100 não estivessem on-line, o sistema seguraria bloqueios em todos os 100 nodos até que todos os 100 estivessem atualizados. Em um cenário WAN (modelo de rede) não confiável por exemplo, claramente a solução de banco de dados distribuído simplesmente não funciona.

Enquanto que uma absoluta sincronização de dados é requerido para aplicações que trabalham com transações financeiras e transferências de fundos bancários, há também muitas aplicações que não precisam das características oferecidas pelo protocolo *two-phase commit*. Neste caso, a replicação de dados simples é a mais aconselhável na maioria dos negócios.

A replicação fornece aos usuários sua própria cópia local. Estas cópias de dados locais atualizáveis podem apoiar a diminuição no tráfego na rede, a fácil escalabilidade e abordagens mais baratas para processamentos distribuídos que não param. O verdadeiro desafio para os fabricantes de banco de dados é alcançar o mesmo nível de integridade , oferecido até agora pelos sistemas que seguem o modelo monolítico central apoiado por mensagens ACID e protocolos *two-phase commit*.

Dada a natureza intrínseca da replicação de dados, quando alguém fala sobre integridade, se refere ao conceito de integridade adiada, que significa uma integridade que é eventualmente alcançada num ponto de tempo adiado (postergado), isto é, quando os dados através de vários nodos se encontram [CAS85].

2.3.5 FORMAS DE REPLICAÇÃO

Há duas formas fundamentais de replicação: Síncrona e Assíncrona.

2.3.5.1 Replicação Síncrona

Conforme [KOR95] e [CAS85] com a replicação síncrona todas as cópias de dados são mantidas exatamente sincronizados e consistentes. Se alguma cópia for atualizada a atualização será imediatamente aplicada para todas as outras cópias dentro da mesma transação. A replicação síncrona é apropriada quando uma consistência exata é importante para a aplicação do negócio.

Segundo [KOR95] e [CAS85] a replicação Síncrona é o modelo que está mais próximo da mensagem ACID; a diferença é que a transação, ao invés de ser aplicada para todos os nodos concorrentes, é dividida em um conjunto de *two-phase commit* para cada localização de cópia dos nodos. Se um nodo está momentaneamente desligado, ele não afetará toda a transação. Logo que o nodo esteja devolta on-line, a transação com a localização primária será recomeçada e a Segunda localização será atualizada.

O modelo de replicação síncrono é caro para implementar e absorve muitos recursos de rede, mas é a única alternativa para aplicações financeiras e bancárias.

2.3.5.2 Replicação Assíncrona

O grande problema da replicação assíncrona é a colisão. Uma colisão é quando o mesmo registro que é fisicamente replicado em dois ou mais nodos, é atualizado durante o período de desenvolvimento assíncrono. Com a replicação assíncrona, cópias ou réplicas de dados se tornaram temporariamente fora de sincronia um com o outro. Se uma cópia é atualizada a mudança será propagada e aplicada as outras cópias como um segundo passo, dentro de transações separadas, que pode acontecer segundos, minutos, horas ou até mesmo dias mais tarde.

As cópias portanto podem ficar temporariamente fora de sincronização, mas com o tempo os dados devem sincronizar os valores em todos os nodos. Há duas abordagens para replicação assíncrona:

a) abordagem Cliente/Servidor;

b) modelo peer-to-peer.

2.3.5.2.1 Abordagem Cliente/Servidor

A abordagem Cliente/Servidor replica dados do Servidor para o cliente, pedindo atualizações no servidor antes que a transação seja considerada um sucesso. Esta abordagem elimina colisões de transação uma vez que somente um nodo pode atualizar uma tabela que é mais tarde replicada para o nodo cliente. O cliente vê a tabela como sendo somente de leitura apenas.

Segundo [KOR95] e [CAS85] do ponto de vista de aplicação existe quatro tipos de replicação de dados dentro do modelo Cliente/Servidor:

1. disseminação de dados: define apenas um nodo Servidor capacitado a acessar as tabelas a serem replicadas em modo de leitura/escrita. Todos os outros nodos participantes na replicação de banco de dados não poderão atualizar as tabelas replicadas, que são renovadas periodicamente do Servidor para o Cliente. Esta é a forma mais simples de replicação de dados, implementada por todos os bancos de dados.

As tabelas em comum entre o Cliente e o Servidor são chamadas de *Snapshots*. Um exemplo de disseminação de dados é uma cadeia de lojas de livros cuja a matriz envia listas de preços atualizadas de livros disponíveis durante a noite. As lojas tem acesso aos preços somente para leitura, enquanto que a matriz pode modificá-los.

2. modelo de consolidação de dados: implementa uma filosofia oposta, onde os nodos periféricos atualizam cada um suas tabelas e então o servidor principal recebe os dados consolidados de todos os sites. O servidor central não pode atualizar suas tabelas (Snapshots) mas simplesmente pede as tabelas replicadas para produzir relatórios reunindo todos os nodos.

Um exemplo de consolidação de dados é uma cadeia de lojas de varejo que reúne informação de vendas o dia todo. Cada loja precisa mandar uma cópia desta informações para a matriz diariamente. No final do dia, estas informações são transmitidas para a matriz e reunidas dentro do depósito central de dados, onde a gerência pode realizar análises de tendências em seu negócio.

3. modelo de divisão de workload: apresenta outro conceito importante amplamente usado na replicação de dados a propriedade de dados. Isto significa que só pode alterar os dados quem for seu dono (owner).

O exemplo que segue mostrará a funcionalidade deste modelo : o servidor com base na Austrália tem posse da sua região e pode portanto atualizar inserir e deletar qualquer tabela que seja desta região. As mudanças são então propagadas para as regiões dos EUA e Nova Zelândia. A Austrália pode consultar as tabelas das outras regiões localmente mas não poderá atualizá-las. Esta estratégia aplica-se também às outras regiões.

4. modelo de propriedade de workflow : neste modelo as informações fluem como em uma linha de montagem. Em qualquer ponto somente um site pode atualizar os dados ,

quando os dados entram na jurisdição de um determinado nodo. Os dados são manipulados de acordo com as regras de negócio locais, e então transmitidas para o próximo nodo competente. Uma vez que a informação tenha sido transmitida o nodo local perde suas capacidades de atualização.

Um exemplo é um sistema de processamento de pedidos: O pedido é registrado pelo departamento de vendas. O pedido pode ser alterado (complementado) até que seja enviado (replicado) para o departamento financeiro. Este último realiza a aprovação de crédito e o faturamento, então o pedido é enviado para a gerência de inventário, para verificar a presença dos itens especificados no pedido no estoque (depósito) da empresa. Se tudo estiver certo, então o pedido chega ao departamento de embarque, que providência a atualização do inventário o empacotamento e a entrega. O mesmo pedido é possuído por diferentes departamentos ao longo de sua passagem através da empresa. Mas Somente um de cada vez é responsável por atualizar o pedido.

Estes quatro modelos têm como objetivo comum o de evitar a colisão de dados. Eles são relativamente fáceis de implementar e administrar. Estes métodos são muito simples de serem implementados, porém a sua flexibilidade é muito pequena. A abordagem Cliente/Servidor não é flexível o suficiente para necessidades específicas. Organizações também requerem a habilidade de apoiar atualizações em nodos múltiplos independentemente um do outro. Cada nodo local pode funcionar de forma autônoma sem levar em conta outros sistemas ou redes participantes no ambiente distribuído.

2.3.5.2.2 Modelo *Pear-to-Pear*

Segundo [KOR95] e [CAS85] este modelo também é chamado de *"Pear-to-Pear Update Anywhere"*, ou seja, atualização não hierárquica em qualquer lugar.

Em abordagens *Pear-to-Pear* atualizações podem ser feitas para qualquer localização de dados e depois são copiados para outras localizações. Uma transação está completada com sucesso logo que alguma ou uma combinação de localizações esteja apta a atualizar uma cópia completa dos dados afetados.

O *Pear-to-Pear* permite a todas as localizações possuírem e manipularem qualquer dado, e transmitir as mudanças quando for pedido. Isso significa que todos os nodos podem atualizar a mesma tabela, que mais tarde é replicada para todos os nodos. Apesar de uma abordagem *Pear-to-Pear* fornecer uma solução geral para distribuíção de transações, ela requer um software para a resolução de colisões.

2.3.5.3 Principais estratégias para resolução de colisões

Existem alguns métodos comumente usados para resolver automaticamente o conflito representado por duas transações aplicadas ao mesmo registro em localizações diferentes, conforme [DAT88].

- *Latest Timestamp* : A última (mais recente) modificação vence sobre as outras.
- *Earliest Timestamop*: A primeira (menos recente) modificação vence sobre as outras.
- *Priority Group* : Grupo de usuários ou nodos têm prioridade sobre os outros. Transações pertencentes a esses grupos vencem.
- *Site Priority* : Um nodo tem prioridade sobre os outros. Suas transações vencem sobre as outras transações dos outros nodos.

- Aditivo: o resultado final para o registro é a soma algébrica ou média de todas as transações aplicadas.

Há também outras técnicas menos usadas. Se o banco de dados é sofisticado o bastante, o administrador de banco de dados distribuído pode escrever rotinas de resolução específicas para implementar lógicas específicas de negócios e atribuir prioridade a elas, para serem usadas pelo sistema como rotinas padrões.

O administrador de banco de dados distribuído pode intervir no processo de resolução definindo rotinas de resolução a serem acionadas quando uma colisão é detectada.

2.3.6 CARACTERÍSTICAS DE UM SISTEMA DISTRIBUÍDO

Segundo [DAT88] as principais características de um sistema distribuído são:

a) dependência de um nó central - Um sistema de banco de dados distribuído não deve depender de um nó central, isso porque depender de um nó central também significa que o sistema possui um único ponto de falha, afetando todos os outros nós. Existindo um nó central também vai acarretar em perda de desempenho do sistema, porque o nó central ficará muito carregado;

b) operação contínua - Um sistema de banco de dados distribuído nunca deve precisar ser desativado. As operações de backup e a recuperação devem ser suportadas on-line. Ainda, as operações citadas anteriormente devem ser rápidas o bastante para não afetarem o funcionamento do sistema (backup incremental, por exemplo);

c) transparência e independência de localidade - Os usuários do sistema não devem saber, nem mesmo estarem cientes do local onde estão localizados os dados. Devem se comportar como se os dados estivessem armazenados localmente. A transparência de localização pode ser alcançada pela utilização de sinônimos estendidos e pelo extenso uso do dicionário de dados. A transparência de

localização permite que aplicações sejam transportadas de um nó da rede para outro sem a necessidade de modificações;

d) independência de fragmentação- As tabelas que fazem parte de um sistema de banco de dados distribuído podem estar divididas em fragmentos e estarem localizadas fisicamente em diferentes nós, de forma transparente para o usuário. Assim como na regra anterior, os usuários e as aplicações não devem estar cientes do fato que alguns dados de uma determinada tabela estão armazenados em um nó diferente do nó onde a tabela originalmente está armazenada;

e) independência de replicação - Dados podem estar replicados em vários nós da rede, de forma transparente. Assim como nas regras de independência de localização e fragmentação, a independência de replicação é projetada para livrar os usuários de preocupações como o local onde os dados estão armazenados. No caso da replicação, os usuários e as aplicações não devem saber que réplicas de dados são mantidas e sincronizadas automaticamente pelo SGBDD.

f) processamento de consultas distribuído - O desempenho de uma consulta deve ser independente do local onde a mesma é executada. Sabendo que um SGBD relacional provê um mecanismo de acesso aos dados não navegável (através de SQL), um SGBDD deve possuir um otimizador que possa selecionar não apenas o melhor caminho para o acesso a um determinado nó da rede, mas também otimizar o desempenho de uma consulta distribuída, levando em conta a localização dos dados, utilização de CPU e I/O e ainda o tráfego da rede;

g) gerenciamento de transações distribuídas - Um SGBDD deve suportar transações atômicas. As propriedades ACID (Atomicidade, Consistência, Independência e Durabilidade) das transações e ainda a possibilidade de serialização devem ser suportadas não apenas para transações locais, mas para transações distribuídas também. Um exemplo de gerenciamento de transações distribuídas é visto no processamento de um *two-phase commit*;

h) independência de hardware - Um SGBDD deve poder operar e acessar dados em uma variedade de plataformas de hardware. Um SGBDD verdadeiro não deve

depender de uma determinada característica de hardware, nem deve ser limitado a uma determinada plataforma;

i) independência de sistema operacional - Um SGBDD deve poder ser executado em sistemas operacionais diferentes. Assim como na regra anterior, um SGBDD não deve depender de um sistema operacional em especial;

j) independência de rede - Um SGBDD deve ser projetado para executar independente do protocolo de comunicação e da topologia de rede usada para interligar os vários nós que fazem parte da rede;

k) independência de SGBD - Um SGBDD ideal deve possuir capacidade para se comunicar com outros sistemas de bancos de dados executando em nós diferentes, mesmo se estes sistemas de bancos de dados forem diferentes (heterogêneos). Todos estes sistemas devem usar APIs (*Application Programming Interface*) em comum.

2.3.6.1 Vantagens e desvantagens aos bancos de dados centralizados

2.3.6.1.1 Vantagens dos sistemas distribuídos

As principais vantagens dos sistemas distribuídos segundo [DAT88], [FUR83] e [CER95] são:

a) autonomia Local - A empresa servida pelo sistema é certamente distribuída de modo lógico (em divisões, departamentos, projetos, etc.) e provavelmente é distribuída de modo físico (em fábricas, usinas, laboratórios, etc.,). A distribuição do sistema permite aos grupos individuais exercerem um controle local sobre os seu próprios dados, com contabilidade local e, de maneira mais geral, que se tornem menos dependentes de um centro de processamento de dados remoto e ao mesmo tempo permite aos grupos locais o acesso aos dados de outras localidades, quando necessário;

b) capacidade e crescimento incremental - Uma razão comum para a instalação de um sistema distribuído é, em primeiro lugar, simplesmente que não exista nenhuma máquina com capacidade adequada para esta aplicação. Uma vez instalado,

ademais, o sistema distribuído pode crescer melhor do que um sistema não distribuído;

c) confiança e disponibilidade - Um sistema distribuído oferece maior confiança do que um sistema centralizado, visto que o mesmo não é uma proposição de tudo-ou-nada - o sistema continua funcionando (a um nível reduzido) em caso de avaria em localidade individual ou de ligação de comunicações individuais entre as localidades. No caso de existência da réplica de dados, a disponibilidade é aperfeiçoada, porque um determinado objeto de dados permanece disponível à medida que pelo menos uma cópia daquele objeto esteja disponível;

d) eficiência e flexibilidade - Os dados podem ser armazenados no sistema distribuído próximo ao seu ponto normal de uso, reduzindo, desta forma, o tempo de resposta e os custos de comunicações (a maioria dos dados devem ter acesso local). O paralelismo inerente nas redes de localidades múltiplas pode fornecer uma passagem de dados aperfeiçoada e, possivelmente, melhorar os tempos de resposta em certas situações;

f) compartilhamento de dados e controle distribuído - se um número de diferentes nós estão conectados, então um usuário em um nó deve poder acessar os dados disponíveis nos outros. A principal vantagem de compartilhar os dados pela distribuição de dados é o fato de que cada nó pode reter um grau de controle sobre os dados armazenados localmente. Em um sistema centralizado, o administrador de banco de dados controla todo o banco de dados. Em um sistema distribuído, existe um administrador de banco de dados global responsável pelo sistema inteiro, mas parte destas responsabilidades são delegadas aos administradores locais de cada nó. Dependendo do projeto do sistema de banco de dados, cada administrador pode ter um diferente grau de autonomia local. A possibilidade de autonomia local é freqüentemente a maior vantagem de bancos de dados distribuídos;

g) aumento da velocidade de processamento de consultas - Se uma consulta envolve dados de diversos nós, deve ser possível dividir a consulta em subconsultas que podem ser executadas em paralelo;

h) crescimento incremental - Um sistema distribuído pode crescer mais facilmente que um sistema centralizado. Se é necessário expandir o sistema porque o volume de dados cresceu ou o volume de processamento aumentou, é mais fácil acrescentar um novo nó a rede de computadores, desde que os nós sejam autônomos, do que substituir um sistema centralizado já existente por outro maior.

2.3.6.1.2 Desvantagens dos sistemas distribuídos

As desvantagens dos sistemas distribuídos segundo [DAT88], [FUR83] e [CER95] são:

a) baixa velocidade :A grande desvantagem de um sistema distribuído, pelo menos nas redes de longa distância é a baixa velocidade, em comparação com a velocidade de leitura dos discos, milhares de vezes superior. Portanto, deve-se procurar minimizar o número e o volume de mensagens no sistema. Este objetivo por sua vez levanta problemas em várias áreas subsidiárias, como processamento de consultas; propagação de atualização, concorrência, recuperação e gerenciamento de catálogos;

b) complexibilidade: complexidade requerida para assegurar a adequada coordenação entre os nós. Esta complexidade adicional toma a forma de custos no desenvolvimento de softwares, grande potencial de bugs, e aumento do overhead de processamento;

c) processamento de consulta - Nos sistemas distribuídos a otimização das consultas é mais importante que nos sistemas centralizados, devendo sempre procurar-se a melhor possibilidade para o processamento, de forma a combinar as junções com o mínimo de tráfego de mensagens possível;

d) propagação de atualização - O problema básico a ser resolvido com relação a atualização dos dados é a reprodução dos dados atualizados, ou seja a atualização de qualquer objeto lógico deve propagar-se para todas as cópias armazenadas do objeto. Nesse aspecto aparecem imediatamente as dificuldades de atualização dos objetos que se acham localizados em um ponto da rede que não esteja disponível no momento da atualização ;

e) concorrência - O controle de concorrência, na maioria dos sistemas distribuídos, baseia-se em bloqueios, da mesma maneira como na maioria dos sistemas não distribuídos. No sistema distribuído, entretanto, as solicitações de teste, conjunto e liberação de bloqueios tornam-se mensagens, e mensagens, portanto, significam sobrecarga no sistema;

f) recuperação - Os protocolos de execução de duas fases são necessários sempre que uma única transação interagir com os múltiplos gerenciadores de recursos autônomos; o propósito de tais protocolos é assegurar que todos os gerenciadores de recursos sigam o mesmo caminho na transação e que todas aceitem ou rejeitem, garantindo dessa maneira, que a transação seja genuinamente tudo-ou-nada. A execução de duas fases é particularmente importante no sistema distribuído, no qual os gerenciadores de recursos distintos encontram-se tipicamente em localidades distintas e, consequentemente, são muito vulneráveis às falhas independentes.

2.4 SQL PADRÃO ANSI

Neste trabalho foi utilizado o SQL padrão com o objetivo de simplificar o protótipo, pois senão, o protótipo deveria administrar a falta de padrão entre os vários bancos de dados.

Structured Query Language (SQL) é uma linguagem de controle e acesso a dados padrão da indústria. É o meio mais usado pelas linguagens de desenvolvimento para acesso à base de dados. Por ser uma linguagem não estruturada e ser de fácil entendimento, seu uso está cada vez mais comum entre os sistemas de informações das empresas. É controlada pelo ANSI (*American National Standard Institute*) que está atualmente na sua terceira versão (ANSI 3) e que tem a finalidade de padronizar o SQL de tal forma que uma aplicação escrita em SQL ANSI seja capaz de acessar qualquer banco de dados padrão SQL do mercado. Mas as empresas de banco de dados implementam extensões à linguagem para ter mais poderes que às outras. Por isso, os desenvolvedores tem que estar atentos a que comandos usar para que seus sistemas fiquem compatíveis a todos os bancos de dados conforme [DAT89] e [HUR90].

O nome “SQL” significa *Structured Query Language* (Linguagem Estruturada de Consulta). A linguagem SQL, segundo [DAT89], é composta por um grupo de facilidades por definição, manipulação e controle de dados em um banco de dados relacional.

2.4.1 HISTÓRICO DO SQL PADRÃO ANSI

Principais fatos na tecnologia de banco de dados que fizeram com que a linguagem SQL ficasse tão conhecida, segundo [DAT89]:

1. Em 1970, E.F. Codd, nessa ocasião membro do laboratório de Pesquisada IBM em San Jose, Califórnia, publicou um trabalho agora clássico, “Relational model of Data for Large Shared Data Banks” – um modelo relacional de dados para grandes bancos de dados compartilhados, em que estabeleceu um grupo de princípios abstratos sobre gerência de banco de dados: o assim chamado modelo relacional. Todo o campo da tecnologia de banco de dados relacional tem sua origem neste modelo. As idéias incentivaram experiências e pesquisas em universidades e laboratórios de pesquisa semelhantes, que resultaram em diversos produtos relacionais agora disponíveis no mercado. As muitas vantagens do método relacional são por demais conhecidas e não precisam ser repetidas aqui.

2. Um objeto em particular da referida pesquisa era o projeto de implementação de protótipo de uma série de linguagens relacionais. Uma linguagem relacional é uma linguagem que efetua, em alguma forma sintática ou concreta, alguma ou todas as características do modelo relacional abstrato. Diversas dessas linguagens foram criadas no início e meio dos anos 70. Uma dessas linguagens em particular foi chamada SEQUEL (structured English Query Language – linguagem de pesquisa em inglês estruturado), definida por D. D. Chamberlim e outros (1974) no Laboratório de Pesquisas da IBM chamado SEQUEL-SRM (1974-75).

3. Em parte como resultado da experiência com o SEQUEL-XRM, foi definida em 1976-77 uma versão revisada do SEQUEL, chamada SEQUEL/2. O nome foi

subsequentemente alterado para SQL por razões legais. Começou o trabalho em outro protótipo mais ambicioso da IBM, chamado System R. O System R, uma implementação de um grande subconjunto da linguagem SEQUEL/2 ou SQL, tornou-se operacional em 1977 e foi Subsequentemente instalado em uma série de estabelecimentos usuários, tanto internos à IBM, quanto sob um grupo de acordos comercias de estudo, cliente selecionados da IBM. Uma série de mudanças posteriores foram feitas á linguagem SQL duramente o projeto do System R, em parte como resposta a sugestões de usuários. Por exemplo, foi incluída uma função EXISTS para testar se um dado especificado existia no banco de dados.

4. Em grande parte, graças ao sucesso do System R, tornou-se aparente ao final dos anos 70 que a IBM provavelmente desenvolveria um ou mais produtos baseados na tecnologia do System R especificamente, produtos que implementassem a linguagem SQL. Como resultado, outros vendedores também começaram a construir seus próprios produtos baseados no SQL. De fato, pelo menos um desses produtos, a saber o Oracle, da Relational Software Inc. (que passou a se chamar Oracle Corporation), foi introduzido no mercado antes dos próprios produtos da IBM. Depois, em 1981, a IBM anunciou um produto SQL, chamado SQL/DS, para o ambiente DOS/VSE. A seguir a IBM anunciou uma outra versão do SQL/DS para o ambiente VM/CMS (1982), e outra para MVS chamada DB2, altamente compatível com o SQL/DS (1983).

5. Nos anos seguintes, diversos outros vendedores também anunciaram produtos baseados no SQL. Esses produtos incluíam tanto produtos inteiramente novos como o DG/SQL (Data General Corporation, 1984) e SYBASE (Sybase Inc., 1986), quanto interfaces do SQL para produtos estabelecidos como o INGRES e o IDM (Britton-Lee Inc.,1982, 1985). Há atualmente mais de cinquenta produtos no mercado que dão suporte a algum dialeto do SQL, rodando em máquinas que cobrem toda a faixa desde microcomputadores até computadores de grande porte. O SQL se tornou o padrão de fato no mundo de banco de dados relacional.

6. O SQL também se tornou o padrão oficial. Em 1982, o *American National Standards Institute* (ANSI) encarregou seu comitê de banco de dados de desenvolver uma proposta para uma linguagem relacional padrão. A proposta do

X3H2, que foi finalmente ratificada pelo ANSI em 1986, possuí essencialmente dialeto IBM do SQL, “letra por letra” . A proposta do comitê X3H2 logo foi aceita como padrão internacional pela ISO (International Organization for Standardization – organização internacional de padrões).

Em muitas formas, o padrão SQL não é particularmente útil por si mesmo. Ele tem sido caracterizado, talvez um pouco asperamente, como “ a interseção das implementações existentes” , e como tal é muito deficiente em uma série de pontos. Reconhecendo este fato, o comitê X3H2 está atualmente trabalhando um grupo de extensões propostas ao padrão básico.

A versão original do SQL tinha por finalidade o uso interativo. Entretanto, foram incluídas mais tarde certas facilidades para permitir a chamada de operações SQL a partir de linguagens de programação como COBOL ou PL/I. Em contraste, o SQL padrão concentra-se quase exclusivamente nessas últimas facilidades, talvez baseando-se no fato de que a padronização é muito mais significativa para portabilidade de programas do que para interfaces interativas.

2.4.2 VANTAGENS DA PADRONIZAÇÃO

Segundo [DAT89] as vantagens da padronização são as seguintes:

- custos reduzidos no treinamento: Os que desenvolvem aplicações podem mover-se um ambiente para outro, sem ser necessário um treinamento caro;

- portabilidade da aplicação: As aplicações desenvolvidas por vendedores de software associados podem ser executadas sem alteração em uma série de ambientes diferentes de hardware e software;

- longevidade da aplicação: As linguagens padrão têm vida útil razoavelmente longa. As aplicações desenvolvidas usando-se tais linguagens também têm, portanto, uma longa vida útil assegurada;

- comunicação através do sistema: Os sistemas diferentes podem se comunicar com mais facilidade uns com os outros. Em particular, diferentes sistemas de gerenciamento de banco de dados poderiam ser capazes de funcionar como

parceiros iguais em um único sistema de banco de dados distribuídos se todos eles suportassem a mesma interface padrão;

- escolha do cliente: se todos os sistemas tivessem suporte para a mesma interface, os clientes poderiam se concentrar no problema de escolher a implementação que melhor atenda as suas necessidades particulares, sem que se envolvam a complexidade adicional de escolher entre diferentes interfaces (possivelmente interfaces muito diferentes).

2.4.3 DESVANTAGENS DA PADRONIZAÇÃO

Segundo [DAT89] as desvantagens da padronização são as seguintes:

- um padrão pode reprimir a criatividade: os implementadores podem ser efetivamente privados de fornecer a melhor solução para algum problema, porque o padrão já prescreve alguma solução alternativa, menos satisfatória, para esse mesmo problema.

- ele falha por não ter qualquer suporte a certas funções claramente necessárias na prática.

Conforme [HUR90], existem vários pontos onde o SQL padrão é inadequado:

- falta de expressões;

- falta de Funções;

- restrições a campos longos;

- restrições de union;

- restrições de group by;

- falta de chave primária;

- falta de chave estrangeira;

- falta de domínios.

2.4.4 TIPOS DE DADOS

Existe uma gramática do SQL padrão que será descrita abaixo conforme [DAT89]:

Tipo de Dados

- Char;
- Numeric;
- Float;
- Date;

Expressões

- Adição (+);
- Subtração (-);
- Multiplicação (x);
- Divisão (/);

Conectores Lógicos

- AND;
- OR;
- NOT;

Predicados

- Comparação (=,<>,<,>,<=,=>);
- Intervalo (BETWEEN,AND);

- LIKE;
- NULL;
- Quantidade (ALL,SOME,ANY);
- EXISTS;
- NOT EXISTS;
- IN
- NOT IN;
- NULL;

Linguagem de Definição de Dados (DDL)

- CREATE TABLE;
- CREATE VIEW;
- CREATE INDEX;
- ALTER TABLE;
- DROP TABLE;
- DROP VIEW;
- DROP INDEX;

Linguagem de Manipulação de Dados (DML)

- INSERT;
- UPDATE;
- DELETE;

- SELECT;

Expressões Tabulares (Cláusulas)

- FROM;
- WHERE;
- GROUP BY;
- HAVING;
- ORDER BY;

Controle de Acesso

- GRANT;
- REVOKE;

Controle de Integridade

- COMMIT;
- ROLLBACK;
- LOCK;

Funções de Agregação

- AVG;
- COUNT;
- SUM;
- MAX;
- MIN;

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo será descrito todos os processos participantes da criação do protótipo.

3.1 TRABALHOS CORRELATOS

Oracle Enterprise

Está ferramenta de replicação utiliza um conjunto de objetos contidos nos bancos de Dados da Oracle (réplica simétrica) que utiliza a replicação assíncrona e modelo Pear-to-Pear. Esta ferramenta só replica dados em um ambiente homogêneo, ou seja, só replica dados em bancos de dados da Oracle na versão 7 conforme [CER95].

3.2 AVALIAÇÃO COMPARATIVA

Uma avaliação comparativa, realizada de forma subjetiva, baseada em critérios arbitrariamente escolhidos, pode ser visualizada na tabela 01. Onde existe marcação ‘X’ significa que tal ferramenta possui tal característica.

Quadro 1 – Avaliação Comparativa

Critério	Oracle Enterprise	Protótipo
Complexibilidade	X	
Facilidade de Instalação	X	X
Facilidade de Configuração		X
Facilidade de Administração		X
Replicação em banco de Dados Genérico		X
Confiabilidade	X	

Os critérios descritos no quadro 01 significam :

a) complexibilidade : este critério significa que o usuário deve possuir um alto grau de conhecimento específico para utilizar a ferramenta.

b) facilidade de instalação: significa que o usuário não necessita de conhecimentos específicos para instalar a ferramenta.

c) facilidade de configuração: significa que o usuário não necessita de conhecimentos específicos para configuração da ferramenta.

d) replicação em banco de dados genérico: significa que a ferramenta pode administrar um ambiente replicado com mais de um produto de banco de dados.

e) Confiabilidade : significa que a ferramenta já foi testada e aprovada em ambiente de produção.

3.3 OBJETIVO

Segundo [CAS85] os softwares Gerenciadores de Banco de Dados Distribuídos (SGBDD) são importantíssimos para a implementação de sistemas informatizados. As empresas, dentro de um marcado globalizado e competitivo, precisam de informações com alto grau de disponibilidade, rapidez, segurança e controle, para que elas tenham produtividade e qualidade nos serviços/produtos que elas atuam.

Para isso, os sistemas informatizados dessas empresas tem que ter performance, integridade, disponibilidade e consistência, entre outros fatores, para que os mesmos tenham sucesso nas suas implementações. Por isso estão cada vez mais sendo adotado soluções que usem Gerenciadores de Banco de Dados Distribuídos, por eles oferecerem todas as características acima citadas, além de aproveitar melhor toda a tecnologia e os recursos disponíveis da plataforma (hardware e software) adotada, alcançando um melhor custo/benefício.

Para que se possa alcançar estes objetivos foi construído este protótipo, sendo uma ferramenta de criação e administração de banco de dados distribuído, que irá disponibilizar recursos para a construção de um ambiente distribuído, independente da marca do produto do banco de dados da sua localização e da arquitetura da máquina hospedeira do banco de dados. Disponibiliza também um conjunto de ferramentas para consulta e administração do ambiente distribuído,

Para a especificação do protótipo foram utilizados diagrama de contexto, modelo de entidade e relacionamento e modelo hierárquico funcional. As ferramentas utilizadas no ciclo de vida do software foram: Oracle Designer 2000 - para o diagrama de fluxo de dados e para o modelo de entidade e relacionamento, Delphi 3.0 da Borland foi a linguagem adotada no desenvolvimento e os bancos de dados utilizados foram: Oracle 8 Personal da empresa Oracle, Microsoft SQL Server 7.0 da Microsoft e o Sybase Anywhere 5.0 da Sybase.

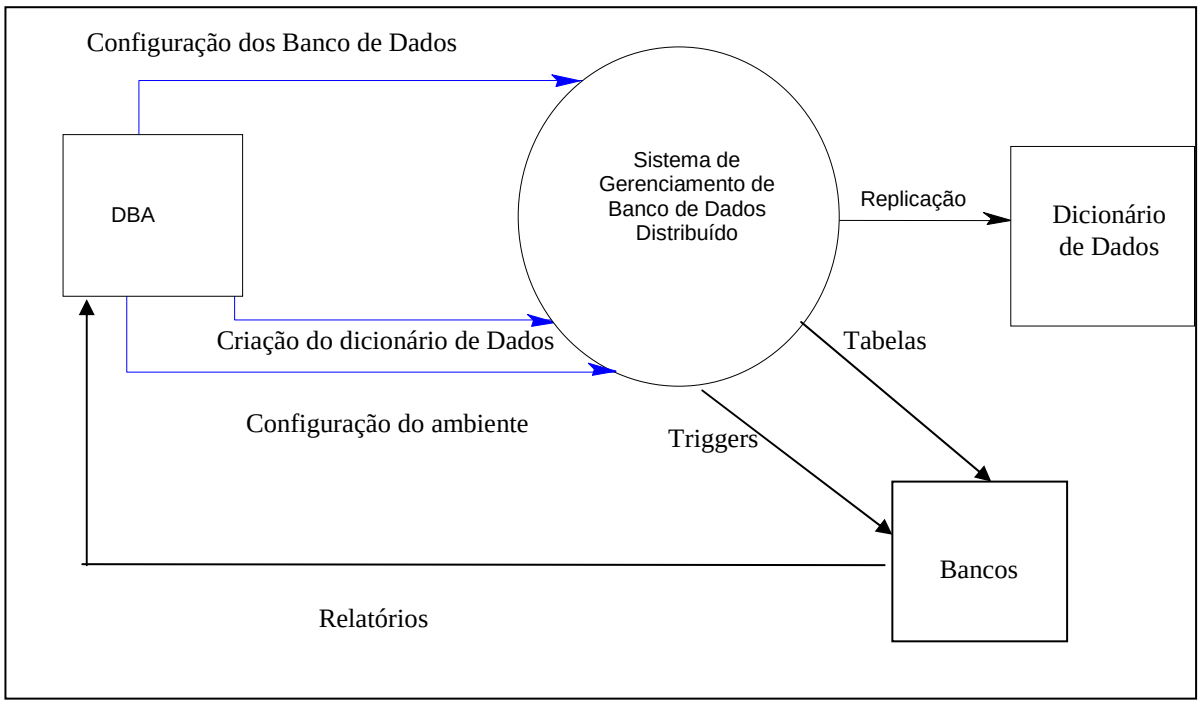
3.4 FUNCIONAMENTO TÉCNICO

Neste item será mostrado tecnicamente como foi construído o protótipo.

3.4.1 DIAGRAMA DE FLUXO DE DADOS

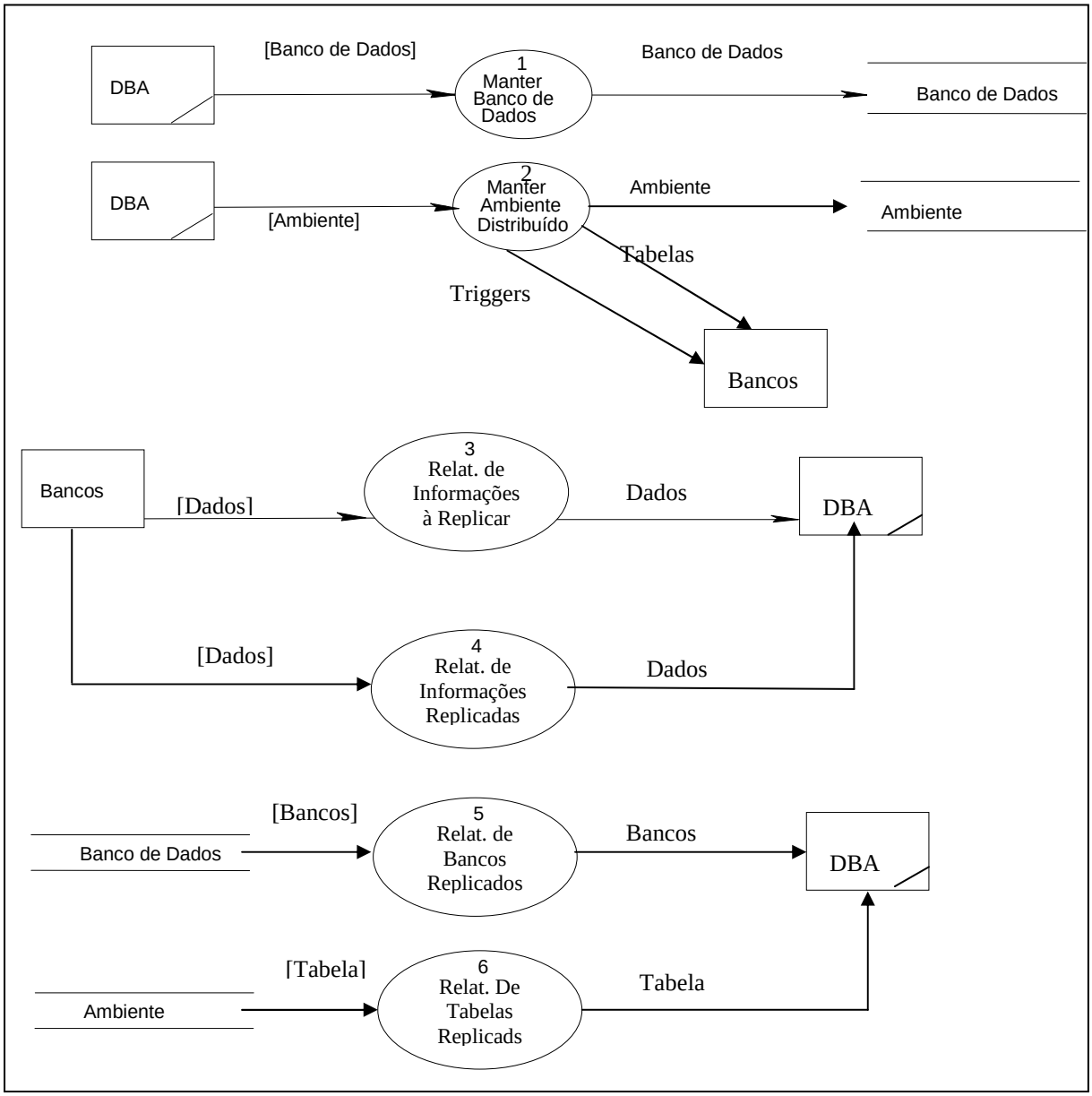
Na figura 1 e 2 são apresentados os diagramas de contexto e nível 0, respectivamente.

FIGURA 1 – DIAGRAMA DE CONTEXTO



A figura 1 mostra o principal processo deste protótipo, neste processo ocorre a criação do ambiente distribuído. Conforme a figura 1 o usuário DBA (Administrador de Banco de Dados) irá configurar o sistema, informando quais os bancos de dados que irão fazer parte do ambiente distribuído, escolhendo um destes bancos para armazenamento do dicionário de dados (onde é armazenado todas as informações do ambiente distribuído para que o protótipo possa gerenciar o ambiente) e a configuração do ambiente. A configuração do ambiente é feita através de informações escolhidas pelo usuário, ele informará quais as tabelas que serão replicadas em quais bancos e o tempo de atualização de cada tabela. No momento que o usuário for gravar sua configuração são disparados outros processos como a criação de tabelas com duas flags de controle e a trigger de replicação.

FIGURA 2 – DIAGRAMA NÍVEL 0



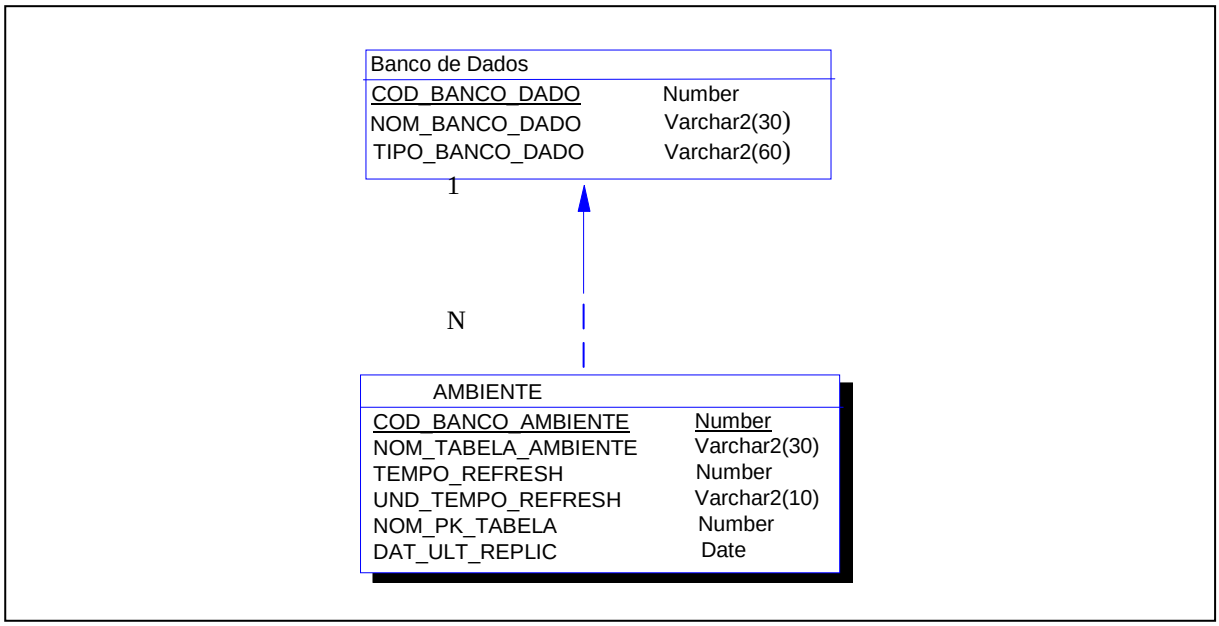
Na figura 2 é apresentado uma visão mais detalhada dos processos. O processo número 1 é o processo de armazenamento dos banco escolhidos pelo usuário que o protótipo irá armazenamento destas informações no dicionário de dados. No processo 2 acontece o armazenamento das informações de tabelas a replicar , bancos de destino e tempo de atualização no dicionário de dados. Este processo também é responsável pela criação física das tabelas a adição de dois campos de controle e também a trigger (gatilho) de replicação de dados.

Os processos 3,4,5,6 são processos de relatório de informações gerenciais do ambiente replicado, podendo o usuário escolher de quais tabelas ou bancos ele quer as informações.

3.4.2 MODELO DE ENTIDADE E RELACIONAMENTO

O Modelo Entidade Relacionamento (MER) enfatiza os principais objetos ou entidades do sistema. O Modelo Entidade Relacionamento do protótipo é apresentado na figura 3.

FIGURA 3 – MODELO ENTIDADE RELACIONAMENTO

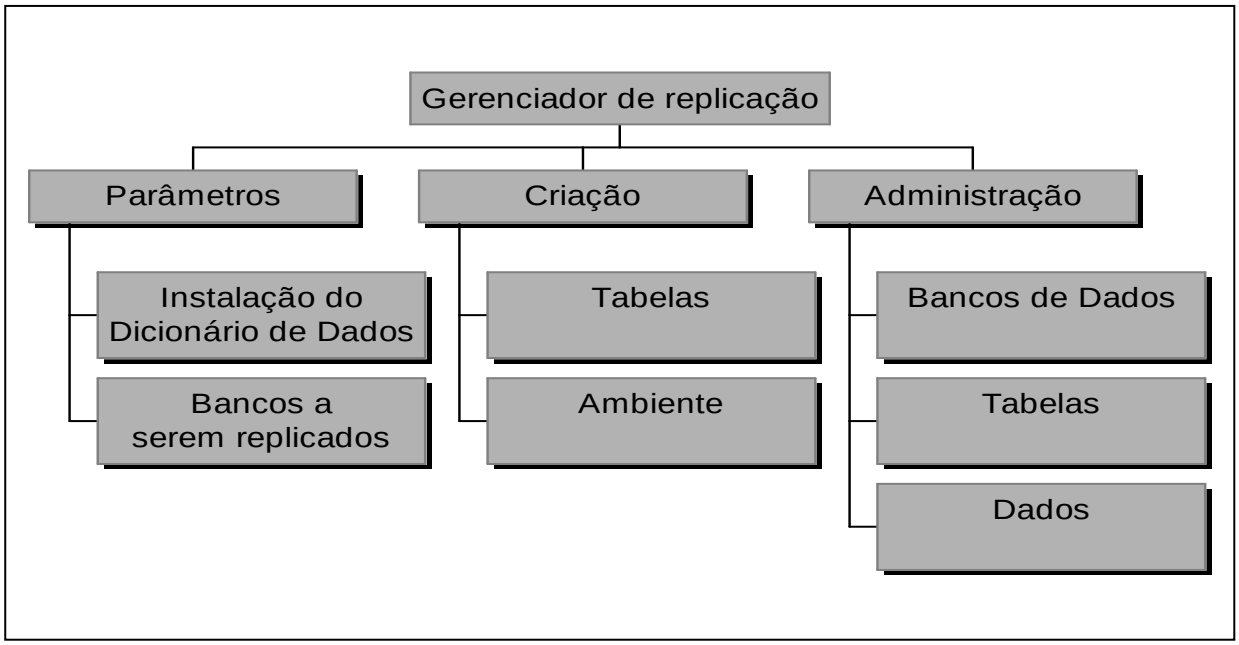


Estas entidades são armazenadas no dicionário de dados do sistema para que a aplicação possa gerenciar o ambiente distribuído.

3.4.3 DIAGRAMA HIERÁRQUICO FUNCIONAL

O diagrama hierárquico funcional do protótipo é apresentado na figura 4, ele consiste de um menu de “Parâmetros” onde o usuário terá a configuração inicial do sistema, neste menu serão cadastrados os banco que participarão da replicação e será criado o dicionário de dados no banco que o usuário escolher. A criação de ambiente e tabelas localizados no menu Criação é aonde será criado todo o ambiente replicado, tornando-se por este motivo o principal processo do sistema. No menu Administração encontram-se os processos de administração do ambiente distribuído tais como: Administração de Tabelas, Bancos e Dados replicados e a replicar.

FIGURA 4 – DIAGRAMA HIERÁRQUICO FUNCIONAL



3.5 FERRAMENTA UTILIZADAS

Neste item será apresenta as ferramentas utilizadas na construção do protótipo e suas principais características.

3.5.1 MICROSOFT SQL SERVER

O SQL Server é um sistema de gerenciamento de bancos de dados cliente/servidor de alto desempenho com alta integração com o Windows NT. Suas características segundo [MIC99] são:

- integração com os serviços de multithreading [múltiplas linhas], agendamento, Monitor de Desempenho, e log de eventos do Windows NT. Um usuário pode se conectar ao SQL Server com a mesma senha usada para a rede Windows NT.

- replicação nativa permite disseminar informações para vários locais, reduzindo a dependência de um servidor único, e deixando a informação necessária mais próxima de quem realmente precisa dela.

- arquitetura paralela, que executa as funções de banco de dados simultaneamente para diversos usuários e tira proveito de sistemas com múltiplos processadores.

- gerenciamento centralizado de todos os servidores através de uma arquitetura de gerenciamento distribuída, com uma interface visual de gerenciamento.

- SQL Server possui uma arquitetura distribuída de gerenciamento [*distributed management framework*], composta de objetos, serviços e componentes. Através dela, vários servidores podem ser gerenciados completamente a partir de qualquer local na rede.

3.5.2 SYBASE

Segundo [SYB99] o Sybase Anywhere 5.0 é um gerenciador de banco de dados que manipula a administração de dados independente da aplicação do cliente e da função de interface do usuário.

O Sybase incorpora um sistema para desenvolvimento de aplicações "ON-LINE" (ativo), endereçando-as com sua arquitetura Cliente/Servidor, na qual as funções de interfaceamento com o usuário são claramente separadas do gerenciamento dos dados e das funções transacionais. Os clientes e servidores do Sybase podem rodar, de acordo com a situação, tanto na mesma máquina como de modo transparente através de uma rede.

A seguir, são citadas algumas características técnicas do Sybase conforme[SYB99] :

- número máximo de linhas por tabela: limitado pelo espaço disponível em disco;

- número máximo de colunas por tabela: 250 colunas por tabela;

- número máximo de tabelas por base de dados: 2 bilhões.

3.5.3 ORACLE

Segundo [CER95] Oracle 8 é um sistema de Banco de Dados, ou SGBD que armazena dados em tabelas chamadas de relações. Essas relações são representações bidimensionais dos dados, onde as linhas chamadas de tuplas no jargão relaciona, representam registros e as colunas, chamadas de atributos são parte de informações contidas nos registros.

O Oracle pode ser comparado a um sistema operacional do computador onde reside. Ele tem suas próprias estruturas de arquivos, de buffer, áreas globais e uma capacidade de se ajustar muito além das capacidades fornecidas no sistema operacional. Oracle controla seus próprios processos, monitora seus registros e consistências e limpa a memória ao sair.

O Oracle no sistema operacional consiste de arquivos executáveis, de cinco a nove processos destacáveis, uma área de memória global, arquivos de dados e de manutenção. Pode ocupar apenas dois megabytes ou ser um grande globo abrangendo gigabytes. A área de memória global é chamada de área global do sistema ou , SGA – é uma área da memória da CPU reservada apenas para utilização do Oracle. Ela contém buffers utilizados para acelerar o fluxo das transações e ajudar a manter a integridade e consistência do sistema. Nenhum dados é diretamente alterado no disco, tudo passa pela SGA. O tamanho e a configuração da SGA são definidos por um arquivo chamado INIT.ORA, que pode conter informações sobre cada tipo de buffer ou área de cache na SGA conforme [CER95].

3.5.4 DELPHI

Segundo [CAN97] Delphi 3.0 é um produto único em sua categoria combinando códigos totalmente compiláveis, ferramentas visuais e tecnologia para a composição de bases de dados escaláveis, possui facilidades para um rápido desenvolvimento em plataforma Windows© e aplicações Cliente/Servidor.

O Delphi é dividido em dois produtos, conforme [CAN97]:

- Delphi Client/Server, de alta performance e facilidade para o desenvolvimento de aplicações e suporte a bancos de dados do tipo Cliente/Servidor;
- Delphi Desktop, de alta performance e facilidade para o desenvolvimento de aplicações e suporte a bancos de dados locais, permitindo total portabilidade à versão Client/Server.

O Delphi oferece o desenvolvimento de aplicações nos ambientes:

- Windows©, Windows 95© e Windows NT©;
- bancos de dados do tipo Cliente/Servidor: Oracle©, Informix©, InterBase, SyBase© e Microsoft SQL Server©;
- alta performance, em sistemas críticos;
- base de Dados locais e aplicações do tipo network;
- ambiente gráfico, visual e multimídia.

Tipos de aplicações desenvolvidas em Delphi:

- usá-lo como a linguagem de desenvolvimento para bancos do tipo Cliente/Servidor;

- ambiente heterogêneo para captura e envio de informações em diversos tipos de arquivos de dados;
- um pacote corporativo de aplicações inteligentes e interpretadores de dados. Incorporando DLL's e EXE's externos;
- pacotes multimídia com desenho e animação;
- genéricos utilitários do Windows©.

Criação de bibliotecas (DLL) para leitura por outras aplicações.

Este ambiente de desenvolvimento foi escolhido para desenvolver este projeto devido a sua grande interação com bancos de dados.

3.6 TÉCNICAS UTILIZADAS

Para implementação deste protótipo foram utilizadas várias técnicas:

- Replicação de dados Heterogênea: porque o protótipo replicará informações em qualquer banco de dados que utilize o padrão ANSI e qualquer tipo de máquina.

- Replicação de dados assíncrona: será utilizado esta técnica pela sua abrangência , ou seja, segundo [DAT88] é a técnica mais utilizada na maioria dos sistemas distribuídos devido ser uma técnica genérica e que não necessita de grande recurso de rede e máquinas.

- Modelo de replicação assíncrona *Pear-to-Pear*: este modelo é utilizado para replicação das informações. Será utilizado este modelo por ser genérico, ou seja, permite que haja atualização de qualquer localização para qualquer outra. Portanto segundo este modelo as informações são distribuídas para todas as localizações integrantes do ambiente replicado.

- *Latest Timestamp*: será utilizada esta técnica para tratamento de conflito. Esta estratégia funciona como se fosse uma pilha, ou seja, no caso de colisão permanecerá a informação que foi atualizada por último.

3.7 FUNCIONAMENTO DA REPLICAÇÃO

Para o funcionamento adequado deste protótipo os banco de dados escolhidos para fazerem parte do ambiente distribuído devem obrigatoriamente utilizar o padrão SQL ANSI, podendo conter extensões.

Para que ocorra a Replicação o protótipo tem alguns requisitos:

- todas as localizações participantes da replicação devem estar interligadas para formar uma rede seja remota (utilizando recursos de teleprocessamento) ou local .

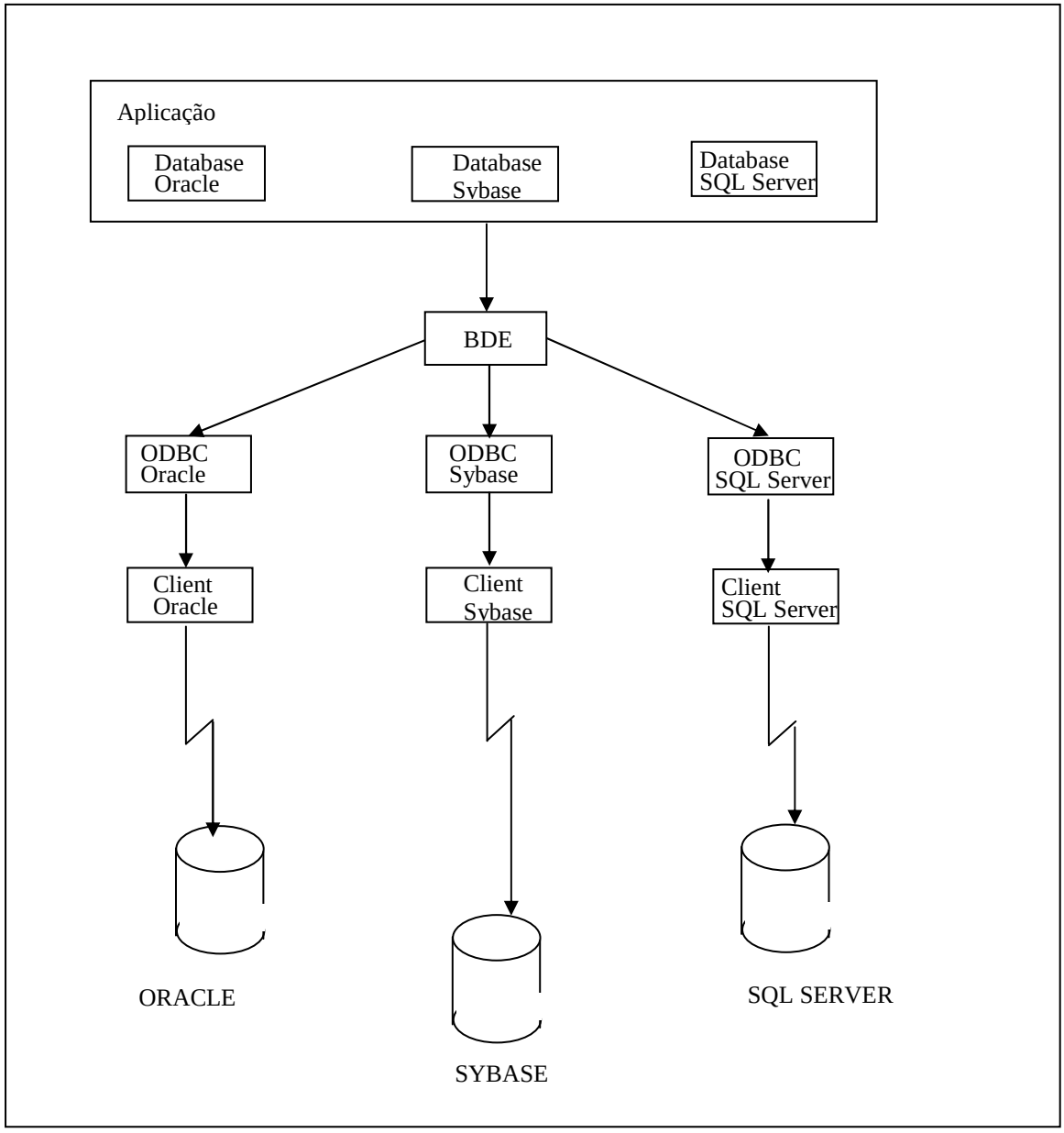
- configuração para que todos os bancos sejam acessados via ODBC;

- configuração do DBE (*Database Borland Engine*) para acessar todos os Banco via ODBC.

- configuração dos Client de cada Banco.

Na figura 5 é apresentado um exemplo dos componentes necessários para replicação:

FIGURA 5 – COMPONENTES NECESSÁRIOS PARA REPLICAÇÃO



3.8 FASES DO PROTÓTIPO

O protótipo é dividido basicamente em quatro fases: Configuração, Criação, Replicação e Administração.

3.8.1 FASE 1:

Nesta fase o usuário deve de verificar os pré-requisitos, ou seja, verificar se todas as unidades a serem replicadas estão em uma mesma rede. É necessário configuração dos clientes dos bancos de dados, ODBC e BDE.

Após a checagem dos pré-requisitos o usuário deverá configurar o protótipo, fazendo a instalação do dicionário de dados e informando quais os bancos de dados que deverão ser replicados.

3.8.2 FASE 2:

Neste passo o usuário irá criar o ambiente distribuído, ou seja, informará ao sistema quais são as tabelas que deverão ser replicadas, para onde as tabelas irão replicar, tempo e unidade de atualização. Todas estas informações serão armazenadas no dicionário de dados no qual o usuário configurou.

Quando o usuário grava a sua configuração o protótipo verifica primeiramente se a tabela que se está querendo replicar possui uma chave única. Caso isso não seja verdadeiro será emitida uma mensagem ao usuário avisando-o que não será possível replicar devido a falta de chave única. Caso contrário o protótipo irá criar a tabela desejada nos bancos assinalados durante a configuração. São adicionadas as tabelas escolhidas para a Replicação duas novas colunas DAT_ULT_ALT e FLG_TRANSMIT. Estes campos serão utilizados no momento da transmissão dos dados, neste momento também é criado uma *trigger* nas tabelas replicadas que serão utilizadas para inserir registros de controle dentro desses campos.

3.8.3 FASE 3:

Nesta etapa o usuário não tem participação, porque o protótipo irá fazer a replicação dos dados conforme a configuração do ambiente.

A replicação ocorre da seguinte forma:

1) no protótipo existe um componente chamado timer (temporizador) que irá disparar a cada 10 segundos, quando o timer é acionado ele dispara uma rotina onde o protótipo irá consultar o dicionário de dados para verificar quais são as tabelas que deverão ser atualizadas.

2) estas tabelas a serem atualizadas são armazenadas em uma fila de espera. Para cada uma destas tabelas o protótipo ira percorrer o campo FLG_TRANSMIT e verificar todos os registros que possuem 'N' como valor. O campo FLG_TRANSMIT terá este valor em duas ocasiões:

- Quando for inserido um novo registro;

- Quando o registro sofrer uma atualização, que não seja a replicação.

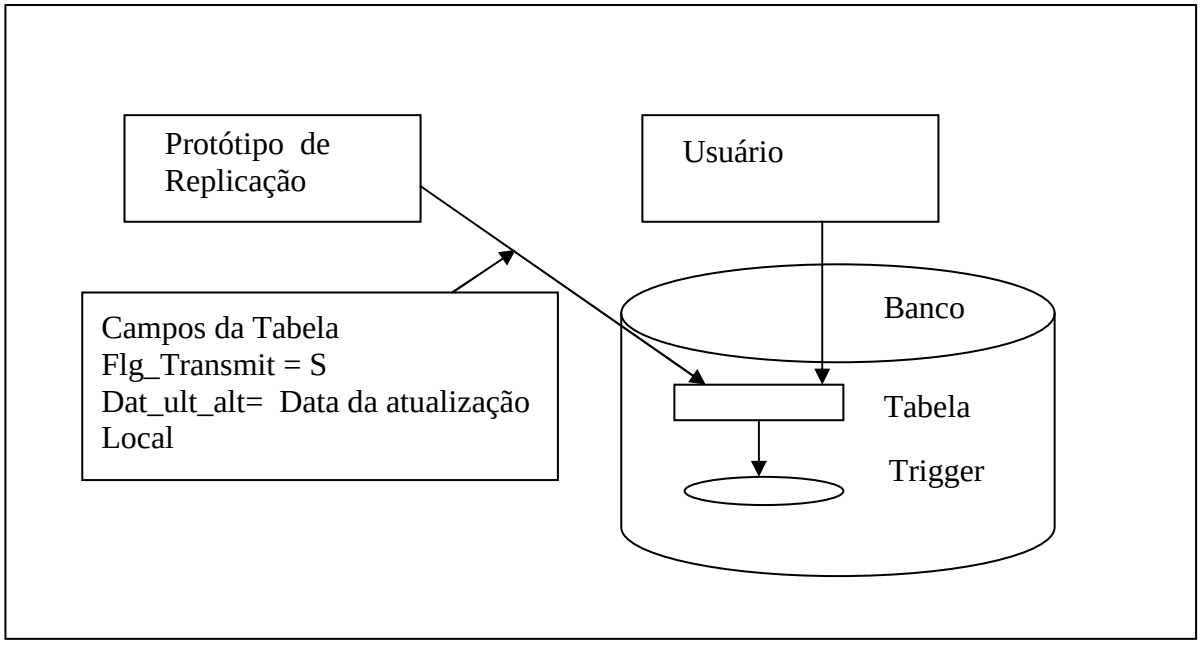
Os campos FLG_TRANSMIT, DAT_ULT_ALT são transparentes para o usuário. Estes campos são atualizados pela *trigger* que foi criada no momento em que o usuário configurou o ambiente. Estes dois campos são utilizados na transmissão dos dados para que a replicação não fique em um laço infinito e para transmitir somente registros que foram recentemente atualizados.

O campo DAT_ULT_ALT recebe da trigger no momento da inserção ou atualização de um novo registro a data do sistema.

3) depois de verificado quais os registros que possuem a flag (indicador) 'N' , estes serão replicados para os bancos configurados pelo usuário. No momento em que os registros são replicados, as flags mudam para 'S' tanto no banco origem como nos bancos de destino. Tanto a inserção como a atualização passam a flag para "N". Por este motivo existe um tratamento especial para a atualização, todo o registro que é replicado é feito uma pesquisa na tabela de destino e é verificado se o registro já existe. Caso exista é feita a atualização do mesmo, desde que o valor Campo DAT_ULT_ALT da tabela de origem seja maior do que o

da tabela de destino. Para saber se a atualização que esta sendo feita é pelo usuário ou pelo protótipo foi criado a seguinte convenção: Quando o valor do campo DAT_ULT_ALT antigo for igual ao do Novo registro, então a flag FLG_TRANSMIT fica com “N” e quando o valor do campo DAT_ULT_ALT antigo for menor que novo registro então a flag FLG_TRANSMIT fica com “S”. Esta convenção foi criada baseada na seguinte lógica: Se um usuário atualiza um registro é preciso que esta atualização seja replicada para todas as localidades. Caso a atualização seja feita pela replicação do protótipo então ela não deverá ser propaga para outras localidade conforme a figura 6.

FIGURA 6 – FUNCIONAMENTO DA ATUALIZAÇÃO

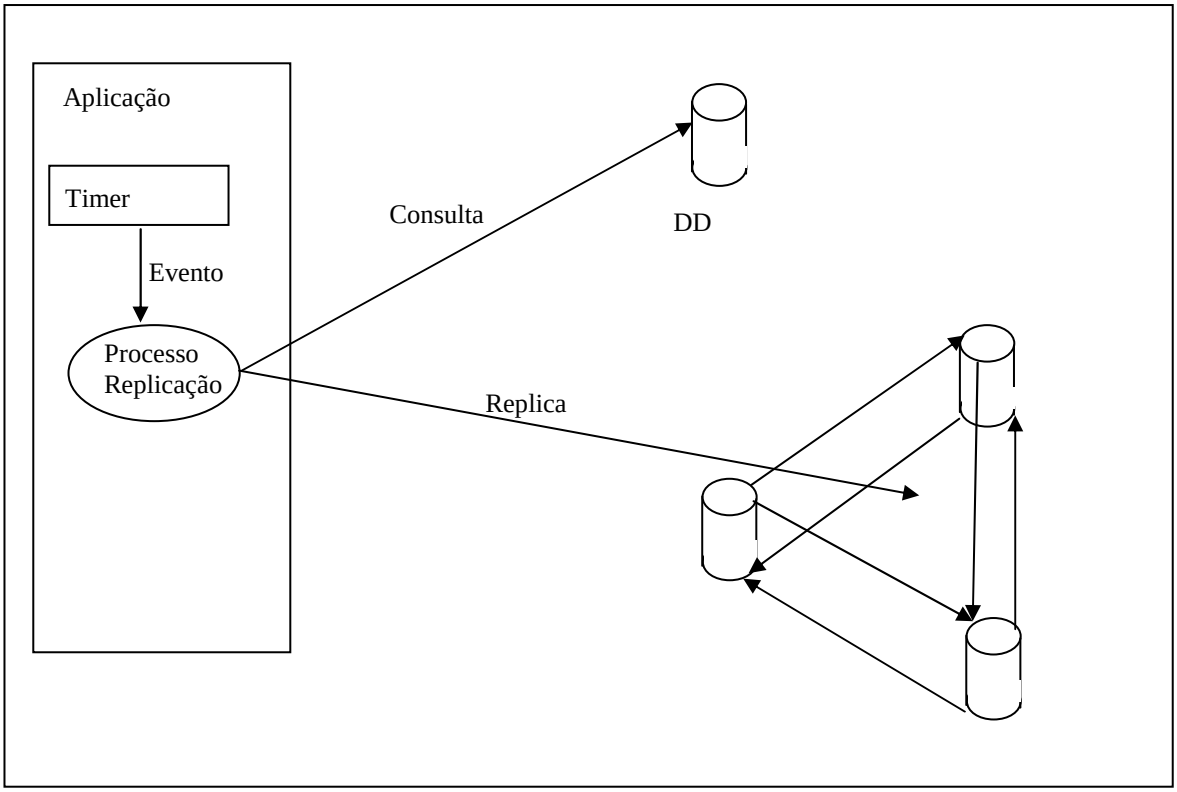


3.8.4 FASE 4:

Nesta fase o usuário fará administração de seu ambiente distribuído. Podendo criar excluir bancos e tabelas no ambiente. Também é disponibilizado neste ambiente relatório de registros replicados e registros a replicar.

Na figura 7 é mostrado de forma superficial o funcionamento da replicação.

FIGURA 7 – FUNCIONAMENTO DA REPLICAÇÃO



3.8.5 CONSTRUÇÃO DO PROTÓTIPO

3.8.5.1 Tela Principal

Utilizou-se um temporizador (Timer) que a cada 10 segundos carrega a lista de bancos de dados dentro de um TQuery (componente de acesso a banco de dados) e a lista de tabelas que estão no dicionário de dados. Para cada banco de dados, atualiza-se os registros das tabelas de destino percorrendo a tabela origem até o fim e inserindo os registros que possuam a Flag = ‘N’. Durante o processo de replicação dos dados o temporizador para a contagem.

3.8.5.2 Tela de Bancos Replicados

Ao exibir a tela de bancos replicados é carregado uma lista de aliases (apelidos) do BDE dentro de listas na tela. É utilizado um componente tipo TTable para fazer a gravação dos dados escolhidos pelo usuário, percorrendo a lista de bancos de dados selecionados e inserindo no dicionário de dados.

3.8.5.3 Criação de Tabelas

Na Tela Criação de tabelas é aberto um TQuery carregando a lista de bancos de dados que estão no dicionário de dados. Ao ser informada o nome da tabela é verificado se a tabela já existe utilizando um componente chamado TTable. Os nomes dos campos são armazenados em uma tabela em memória que é percorrida copiando a definição dos campos e índices para um TTable que criará a tabela no banco de dados selecionado.

3.8.5.4 Instalação do Dicionário

Para esta tela é aberto um TQuery que carrega a lista de bancos cadastrados no DBE. Para instalar o dicionário de dados utiliza-se o componente TQuery que executam instruções SQL (Create tables) no banco selecionado para a criação das tabelas do dicionário.

3.8.5.5 Replicação de Dados

Para a replicação de dados é carregado os bancos de dados do dicionário de dados em um TQuery e as tabelas que estão sendo replicadas em outro TQuery. Na página de dados à replicar é executado uma instrução SQL (consulta nas tabelas replicadas com a flag = ‘N’), através de um TQuery, que exibe os registros que ainda não foram replicados. Na página de dados replicados é executado uma instrução SQL(consulta nas tabelas replicadas com a flag = ‘S’) que exibe os registros que já foram replicados.

3.8.5.6 Tabelas Replicadas

É aberto uma TQuery que carrega as tabelas do dicionário de dados. Ao selecionar uma tabela é exibida as demais informações da tabela e são carregados em outro Tquery os bancos para onde a tabela selecionada está sendo replicada. Para excluir uma tabela do dicionário é utilizado um Tquery (comando de deleção) para excluir a tabela selecionada.

3.8.5.7 Tela de Ambiente

Ao exibir a tela é carregado os bancos de dados cadastrados e seus respectivos usuários para uma lista em tela. Ao ser selecionado um usuário de um banco é carregada a lista de Tabelas pertencentes ao mesmo.

A gravação das tabelas selecionadas no dicionário de dados é feita utilizando componentes TTable. A criação dos campos e da trigger de inserção e atualização na tabela é feita utilizando componentes TQuery para a execução de instruções SQL em todos os bancos. A inserção dos registros na (s) tabela (s) de destino é feita percorrendo a tabela origem desde o início inserindo registro a registro após a tabela ter sido criada no banco de destino.

OBS: TQUERY é um componente de acesso a banco de dados baseado em instruções SQL. Como por exemplo: Select, Insert, Delete e Update (DML), Create e Drop (DDL).

TTABLE é um componente de acesso a dados que simula a estrutura de uma tabela que é utilizado para fazer manutenção de registros em uma única tabela.

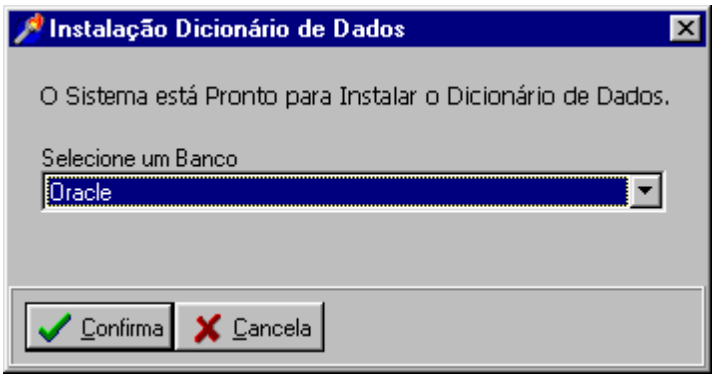
4 FUNCIONAMENTO DO SOFTWARE

Ao executar o sistema, será apresentado a tela principal do protótipo, onde dará acesso aos demais recursos do mesmo, tais como Instalação do Dicionário de Dados, Inclusão de Bancos de Dados, Criação de tabelas, Criação do Ambiente replicado, administração de Tabelas Replicadas, Administração de Bancos Replicados e relatório de dados Replicados, conforme figura 8 :

FIGURA 8 – TELA PRINCIPAL DO PROTÓTIPO

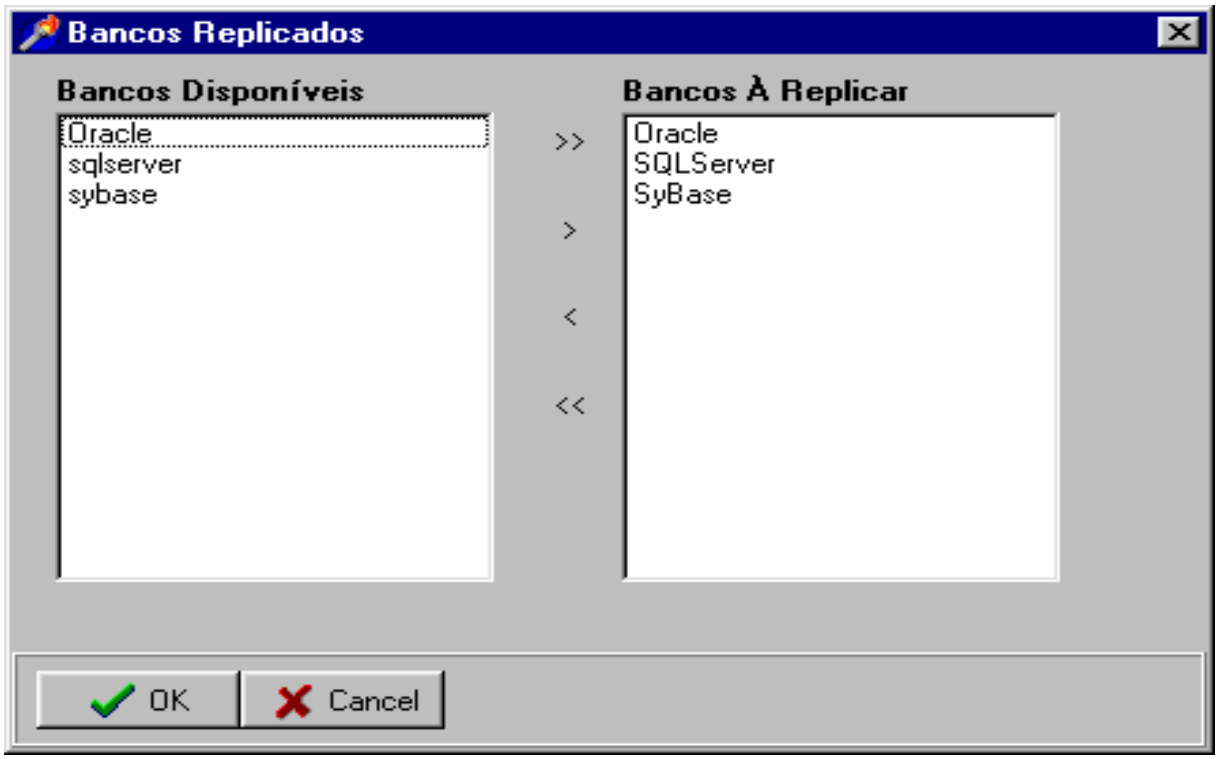
A figura 9 apresenta a tela de instalação do dicionário de dados, o usuário irá escolher o banco de dados onde irá instalar o dicionário de dados (criação de tabelas para o protótipo armazenar informações necessárias para o gerenciamento do ambiente distribuído), e irá gravar os dados da mesma.

FIGURA 9 – TELA DE INSTALAÇÃO DO DICIONÁRIO DE DADOS



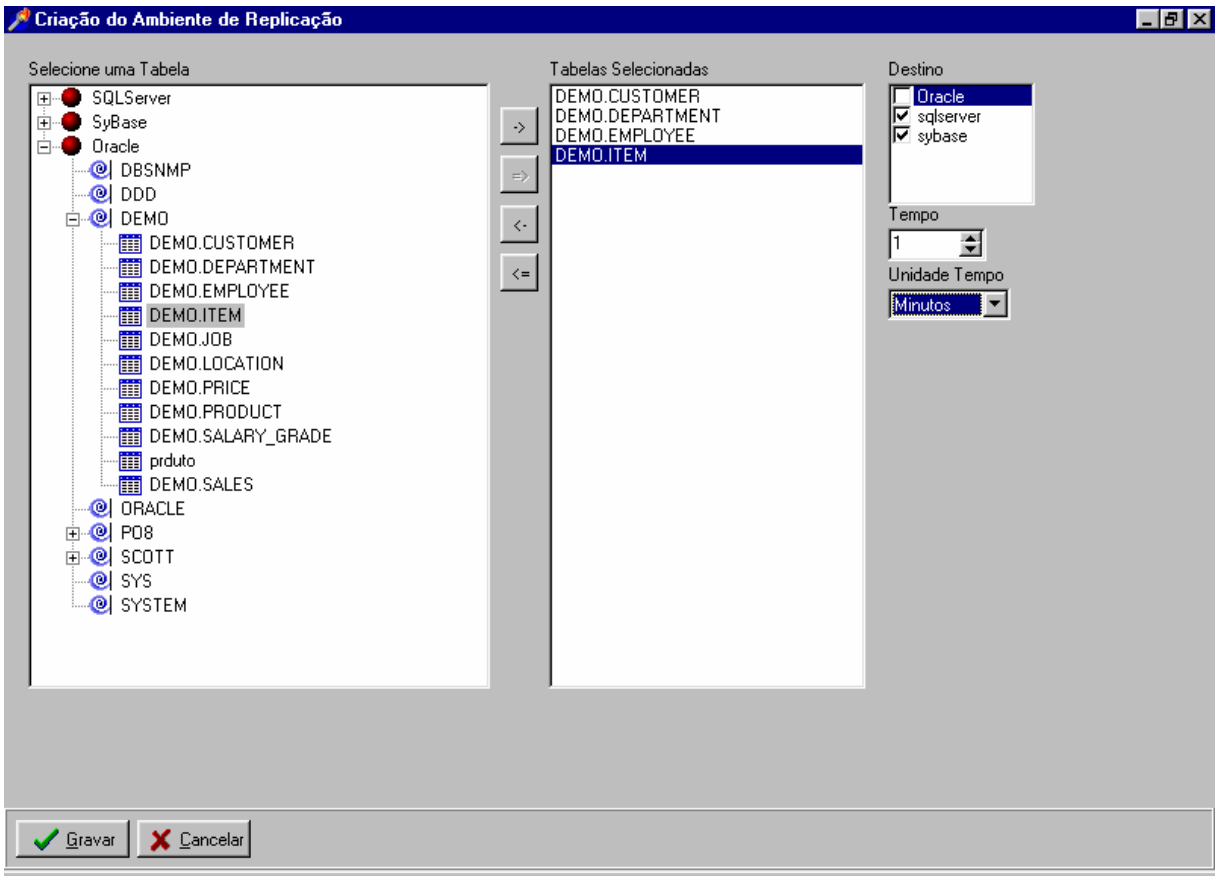
O usuário irá cadastrar os banco de dados que irão participar da replicação, conforme a figura 10.

FIGURA 10 – TELA DE CADASTRAMENTO DOS BANCO DE DADOS



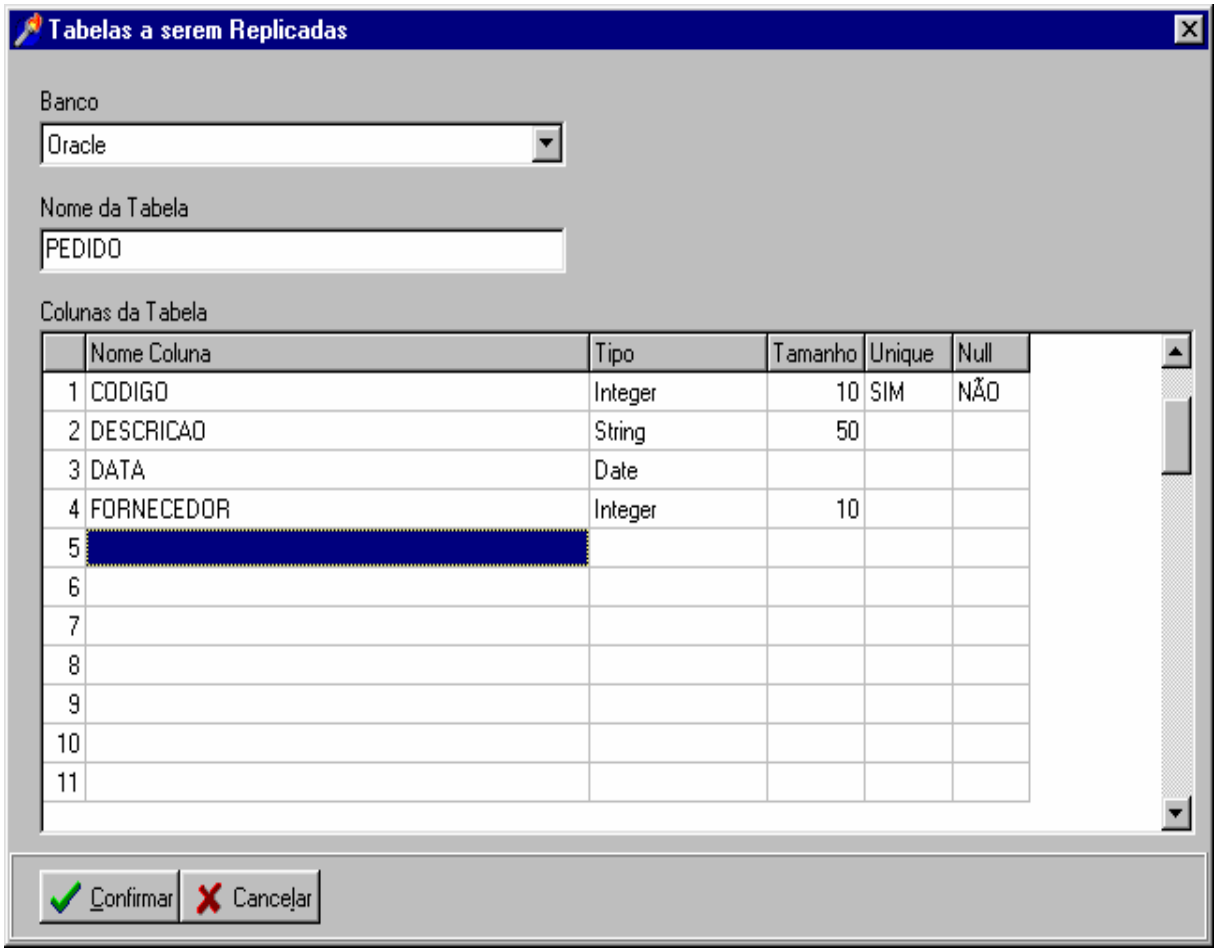
Na tela de Criação do Ambiente de Replicação o usuário irá fornecer as principais informações do sistema o que torna esta tela a mais importante. Irá informar ao sistema através de seleção quais serão as tabelas a serem replicadas, para quais bancos as mesmas serão replicadas, o tempo e unidade de atualização e então irá gravar as informações conforme a figura 11.

FIGURA 11 – TELA DE CRIAÇÃO DO AMBIENTE DE REPLICAÇÃO



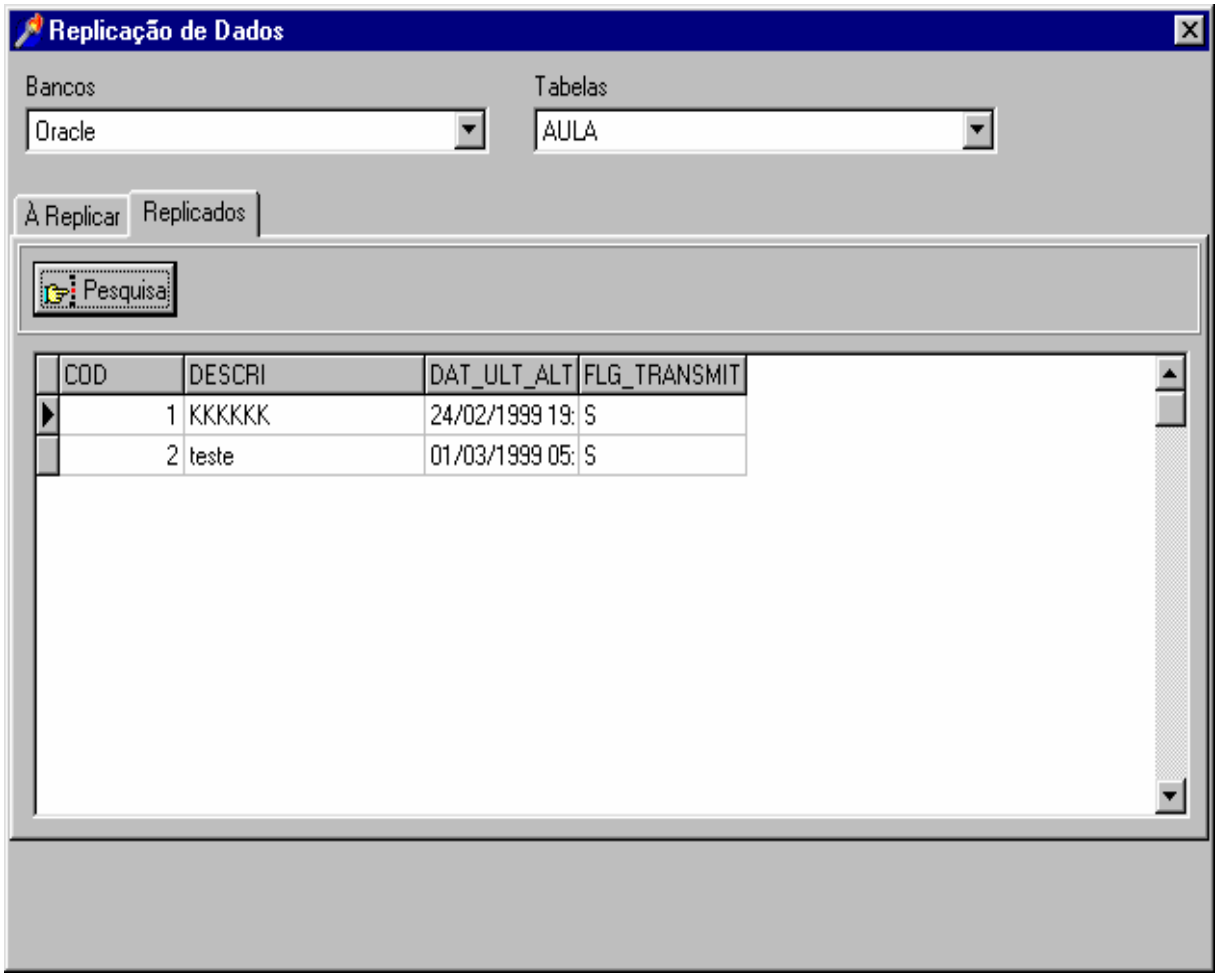
A figura 12 mostra a tela de criação de tabelas. O propósito desta tela é não deixar o usuário sair do protótipo caso não exista em nenhum banco de dados a tabela que se deseja replicar. O usuário irá informar em qual banco de dados será criada a tabela, o nome da mesma, as colunas e tipo de dados.

FIGURA 12 – TELA CRIAÇÃO DE TABELAS



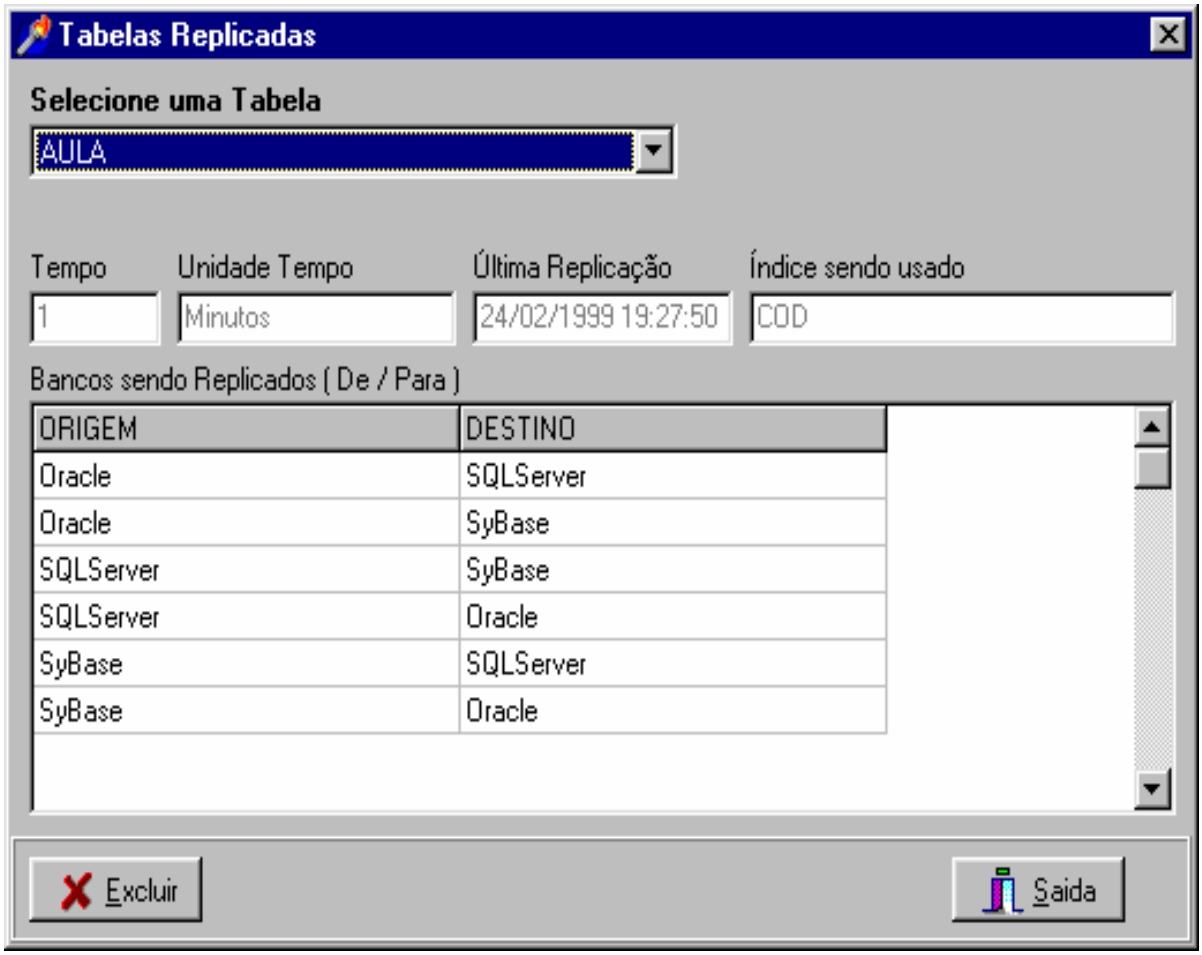
A figura 13 mostra os dados replicados ou a replicar conforme a escolha do usuário, para isso o usuário irá informará ao protótipo a tabela desejada e de qual banco de dados ele irá querer as informações.

FIGURA 13- TELA DE RELATÓRIO DE DADOS REPLICADOS



Na janela Tabelas Replicadas o usuário informará a tabela e o sistema mostrará as informações conforme a tela da figura 14.

FIGURA 14 – RELATÓRIO DE TABELAS REPLICAS



5 CONCLUSÃO

Os softwares Gerenciadores de Banco de Dados Distribuídos (SGBDD) segundo [LEL78] são peças importantíssimas para a implementação de Sistemas informatizados. As empresas, dentro de um mercado globalizado e competitivo, precisam de informações com alto grau de disponibilidade, rapidez, segurança e controle, para que elas tenham produtividade e qualidade nos serviços/produtos que elas atuam.

Um sistema de banco de dados distribuído fornece um ambiente no qual novas aplicações de banco de dados podem fazer o acesso a dados a partir de uma variedade de banco de dados preexistentes localizados em ambientes de hardware e software heterogêneos.

Após o término da fundamentação teórica ficou claro que os métodos, modelos e técnicas de replicação devem ser analisados para cada problema. Não pode-se concluir quais os modelos mais importantes ou quais são ideais, mas sim quais se adaptam melhor no problema que está querendo ser resolvido. Para utilização em ambiente genéricos onde não existe padronização de produtos de bancos de dados e para aplicações genéricas que não necessitam de uma sincronização rígida estas técnicas dentre outras são as mais eficientes. Tornando ao protótipo uma ferramenta genérica capaz de absorver um fatia maior de mercado.

As metodologia de replicação assimétrica utilizada nesta implementação é a mais adequada tendo em vista os objetivos traçados, pois a principal vantagem deste trabalho que implementou uma ferramenta de criação e gerenciamento de banco de dados genérico distribuído é a sua grande flexibilidade na replicação dos dados , pois independentemente dos tipos de banco de dados, da arquitetura da máquina hospedeira e do sistema operacional o protótipo é capaz de replicar as informações conforme configurado.

5.1 SUGESTÕES PARA FUTUROS TRABALHOS

Para um futuro TCC esta ferramenta poderia ser incrementada de rotinas para fazer a replicação entre banco de dados que utilizam padrões diferentes, ou seja, que utilizem padrões diferentes do SQL ANSI.

5.2 DIFICULDADES ENCONTRADAS

Pelo fato de contar apenas com uma máquina para o desenvolvimento deste trabalho teve-se que configurar e otimizar os três bancos de dados de forma a caberem na memória da máquina utilizada. Durante o desenvolvimento do protótipo os banco de dados tinham que estar fechados pois a máquina não conseguia abrir os banco e o Delphi ao mesmo tempo. Portanto para testar o protótipo o Delphi era fechado e os bancos de dados eram abertos com o executável do protótipo.

REFERÊNCIAS BIBLIOGRÁFICAS

[CAN97] CÁNTU, Marco. Dominando o Delphi 3. Makron Books, 1995.

[CAS85] CASANOVA, Marco Antônio. Princípios de Sistema de Gerência de Banco de Dados Distribuídos. Rio de Janeiro : Campus, 1985.

[CER95] CERICOLA, Vicent Osvald. Oracle: Banco de Dados Relacionais e Distribuídos, Ferramentas para Desenvolvimento. São Paulo : Makron Books, 1995.

[CHU83] CHU, Shao Yong. Banco de Dados: Organização, Sistemas e Administração : São Paulo : Atlas, 1983.

[COU84] COUCEIRO, Luis Antônio Carneiro Cunha. Sistemas de Gerência de Banco de Dados distribuídos. Rio de Janeiro: Livors Técnicos e Ciêntificos, 1984.

[DAT85] DATE, C. J. . Banco de Dados: Fundamentos. Rio de Janeiro : Campus 1985.

[DAT88] DATE, C. J.. Banco de Dados : Tópicos Avançados. Rio de Janeiro: Campus, 1988.

[DAT89] DATE, C. J.. Guia para o Padrão SQL. Rio de Janeiro : Campus, 1989.

[DAT91] DATE, C. J.. Introdução a Sistemas de Banco de Dados. Rio de Janeiro : Campus, 1991.

[ENG97] ENGO, Frank. Como Programar em Delphi 3. São Paulo : Makron Books, 1997.

[FUR83] FURTADO, Antônio Luz. Organização de Banco de Dados. São Paulo : Campus, 1983.

[HUR90] HURSCH, Carolyn J. SQL Linguagem de Consulta Estruturada. Rio de Janeiro : LTC-Livros Técnicos e Cientifícos, 1990.

[KIN81] KIND, Judy. Evaluating Database Management systems. New York : Van Nostrand Rubhold, 1981.

[KOR95] KORT, Henry F. Sistemas de Banco de Dados. São Paulo : Makron Books, 1995.

[LEL78] LELLIS, João Vitor. Sistemas de Dados Distribuídos. Rio de Janeiro : PUC, 1978.

[MIC99] MICROSOFT, Características do SQL Server. Endereço Eletrônico : www.microsoft.com, 1999.

[REN94] RENAUD, Paul. Introdução aos Sistemas Cliente/Servidor. Rio de Janeiro : Infobook, 1994.

[SET89] SETZER, Valdemar Waingort. Banco de Dados: Conceitos, Modelos, Gerenciadores e Projetos Lógicos. São Paulo : PUC, 1989.

[SYB99] SYBASE, Características do Sybase. Endereço Eletrônico: www.sybase.com, 1999.

[VAS95] VASKEVITCH, David. Estratégia Cliente/Servidor. São Paulo : Berkeley, 1995.