



CENTRO UNIVERSITÁRIO DO TRIÂNGULO
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Gerenciamento de Dados em Banco de Dados Distribuídos

Samuel Lázaro de Oliveira Cunha

Uberlândia, Julho/2003.



CENTRO UNIVERSITÁRIO DO TRIÂNGULO
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Gerenciamento de Dados em Banco de Dados Distribuídos

Samuel Lázaro de Oliveira Cunha

Monografia apresentada ao Curso de
Ciência da Computação do Centro
Universitário do Triângulo - Unit, como
requisito básico à obtenção do grau de
Bacharel em Ciência da Computação,
sob a orientação do Prof. Hélio Rubens
Soares, Msc.

Uberlândia, Julho/2003.

Gerenciamento de Dados em Banco de Dados Distribuídos

Samuel Lázaro de Oliveira Cunha

Monografia apresentada ao Curso de Ciência da Computação do Centro Universitário do Triângulo - Unit, como requisito básico à obtenção do grau de Bacharel em Ciência da Computação.

Hélio Rubens Soares, Msc.
(Orientador)

Elmo Batista de Faria, Dsc.
Ronaldo Castro de Oliveira, Msc.
(Avaliadores)
Marcos Ferreira de Rezende, Dsc.
(Coordenador de Curso)

Uberlândia, Julho/2003.

Agradeço primeiramente a Deus por ter me dado os pais mais maravilhosos e carinhosos que possam existir, pois sem eles não conseguiria este sonho. Agradeço aos meus pais que sempre me apoiaram e me ajudaram. Meus agradecimentos também ao meu orientador Hélió Rubens e a todos os amigos que durante essa caminhada eu consegui fazer.

RESUMO

O objetivo deste trabalho é mostrar o funcionamento de uma estrutura de Bancos de Dados Distribuídos (BDD), considerando os aspectos de segurança, concorrência, integridade dos dados, além de fazer um comparativo entre a tecnologia centralizada e a tecnologia distribuída de dados. A utilização de computadores localizados em lugares diferentes deu origem às redes de computadores. Mais do que compartilhar informações, ao se conectar vários computadores em rede, tem – se uma melhoria na capacidade de processamento. É comum em um banco de dados que certos dados interessem a somente um lugar da rede e seria razoável que esses dados fossem guardados junto a esse lugar. É da descentralização do armazenamento desses dados que resulta em um BDD. O critério de localização das informações é influenciado pela estrutura de controle adotada na rede. Para dados utilizados com frequência por dois ou mais locais, cria-se cópias para cada lugar. Existem casos em que os dados armazenados estão de formas diferentes em cada lugar. Dessa maneira, obriga-se a fazer uma conversão nas informações antes de serem enviadas para os outros locais da rede. A base de dados pode ser mantido em um único local ou então em cada um dos lugares ligados pela rede. A principal vantagem de se distribuir dados em um sistema de banco de dados é compartilhar e acessar dados de uma maneira segura e eficiente. A principal desvantagem é o acréscimo de complexidade exigida para assegurar coordenação própria entre os locais. Um dos principais cuidados é com a sincronização dos bancos nos locais da rede, que deve garantir que um determinado local tenha todas as informações dos demais locais.

SUMÁRIO

1. INTRODUÇÃO.....	1
2. SISTEMAS DE BANCO DE DADOS	3
2.1 Tipos de SGBDs Distribuídos	6
2.2 Vantagens dos SBDDs	7
2.3 Desvantagens dos SBDDs	9
2.4 Áreas de Problemas nos Bancos de Dados Distribuídos	10
2.5 Conclusão.....	12
3. PROJETO DE BANCO DE DADOS DISTRIBUÍDOS	14
3.1 Estrutura do Banco de Dados.....	15
3.2 Controle dos Identificadores	18
3.3 Controle dos Cadastros	19
3.4 Atualização dos Bancos de Dados.....	21
3.5 Controle de Replicação de Dados	22
3.6 Conclusão.....	25
4. ESTUDO DE CASO.....	26
4.1 Cadastros	26
4.1.1 Cadastrar Usuários	27
4.1.2 Cadastrar Identificação	27
4.1.3 Cadastrar Cidades	27
4.1.4 Cadastrar Clientes	28
4.1.5 Cadastrar Vendedores.....	29
4.1.6 Cadastrar Fornecedores.....	29
4.1.7Cadastrar Representadas.....	30
4.1.8 Cadastrar Produtos.....	30
4.2 Atualizações e Exclusões	32
4.3 Vendas	33
4.3.1 Cadastrar Pedido	34
4.3.2 Alterar Pedido	36

4.3.3 Cancelar Pedido	36
4.4 Sincronização dos Bancos de Dados	36
4.4.1 Atualizando o Banco de Dados do Terminal	37
4.4.2 Atualizando o Banco de Dados do Servidor.....	40
4.4.3 Antes e Depois da Replicação	40
4.5 Conclusão.....	42
5 CONCLUSÃO	44
REFERÊNCIAS BIBLIOGRÁFICAS	46

1. INTRODUÇÃO

Sistema de bancos de dados distribuídos é a união de duas tecnologias distintas para processamento de dados, os sistemas de bancos de dados e as redes de computadores. Os sistemas de bancos de dados com o passar do tempo, mostraram ser algo além de um simples aplicativo para gerenciar dados de forma centralizada, como também um aplicativo para gerenciar através de uma rede de computadores, dados de uma forma distribuída.

O sistema de bancos de dados distribuídos nada mais é do que um conjunto de vários bancos de dados logicamente inter-relacionados, fisicamente separados em diferentes lugares, dispersos geograficamente, distribuídos por uma rede de computadores.

O objetivo deste trabalho é mostrar o funcionamento de uma estrutura de Bancos de Dados Distribuídos, considerando os aspectos de segurança, concorrência e integridade dos dados.

O capítulo 2 apresenta as principais características de um sistema de bancos de dados, os tipos de sistema de gerenciamento de banco de dados distribuídos, vantagens e desvantagens da distribuição dos dados e os principais problemas que podem existir com bancos de dados

distribuídos.

No capítulo 3 é descrito um projeto de SBDD, onde é apresentado um projeto de um sistema de representações. Este projeto contém um método de distribuição e sua estrutura contém diversos bancos de dados espalhados em diversos locais, entre computadores remotos (terminais) e um computador local (servidor).

O capítulo 4 é apresentado um estudo de caso apresentando como o projeto funciona utilizando o método de distribuição mostrado no capítulo 3.

No capítulo 5 é feita a conclusão do trabalho desenvolvido.

2. SISTEMAS DE BANCO DE DADOS

“Um sistema de banco de dados (SBD) é basicamente um sistema de manutenção de registros por computador” [8]. O objetivo dos SBDs é manter os registros e disponibilizar quando solicitados. Estes registros ficam gravados em arquivos com uma estrutura de um banco de dados.

Um banco de dados é uma grande quantidade de informações armazenadas de forma estruturada em um computador. Para ter acesso as informações gravadas dentro do banco de dados, é necessário um sistema de gerenciamento de banco de dados.

“Um sistema de gerenciamento de bancos de dados (SGBD) consiste em uma coleção de dados inter-relacionados e em um conjunto de programas para acessá-los” [2]. O objetivo de um SGBD é criar um ambiente adaptado para a recuperação e armazenamento das informações do banco de dados.

Os sistemas de banco de dados existem de duas formas: os sistemas de bancos de dados centralizados e os sistemas de bancos de dados descentralizados ou distribuídos. Em um banco de dados centralizado, todos os integrantes do sistema ficam em um mesmo computador ou *site* (um local onde algo se baseia). Os integrantes do sistemas são os

dispositivos para armazenamento, o software (aplicativos) de gerenciamento de banco de dados e todos os dados. Este tipo de banco de dados centralizado pode ser acessado por vários terminais diferentes conectados ao *site*, porém o SGBD e todos os dados estão situados em um único *site*.

Com o progresso, melhoria, nas áreas de bancos de dados e tecnologia de processamento em rede, a distribuição de sistemas de computadores em vários *sites* conectados em rede começaram a surgir. Começaram então a estudar uma nova forma de se trabalhar com dados de forma distribuída.

“Pode - se definir como bancos de dados distribuídos (BDD), como um conjunto de vários bancos de dados logicamente inter-relacionados, fisicamente separados em diferentes *sites*, dispersos geograficamente, distribuídos por uma rede de computadores” [1]. Segundo ÖZSU [1], KORTH [2] e CASANOVA [3], “um sistema de gerenciamento de bancos de dados distribuídos (SGBD Distribuídos) é conceituado como um sistema que possibilita o gerenciamento dos bancos de dados distribuídos e que transforma a distribuição dos dados de forma clara para os usuários, como se fosse um sistema centralizado”.

O termo “sistema de bancos de dados distribuídos” (SBDD) atribui ao conjunto entre os bancos de dados distribuídos e o SGBD distribuído. As figuras 2.1 e 2.2 ilustram a diferença de uma arquitetura de um SBDD de uma arquitetura centralizada.

A figura 2.1 mostra que para cada trabalho realizado por um *site*, as informações têm de passar pela rede e chegar ao *site* onde está o banco

de dados, podendo assim fazer com que o serviço seja lento, pois a rede pode ficar congestionada.

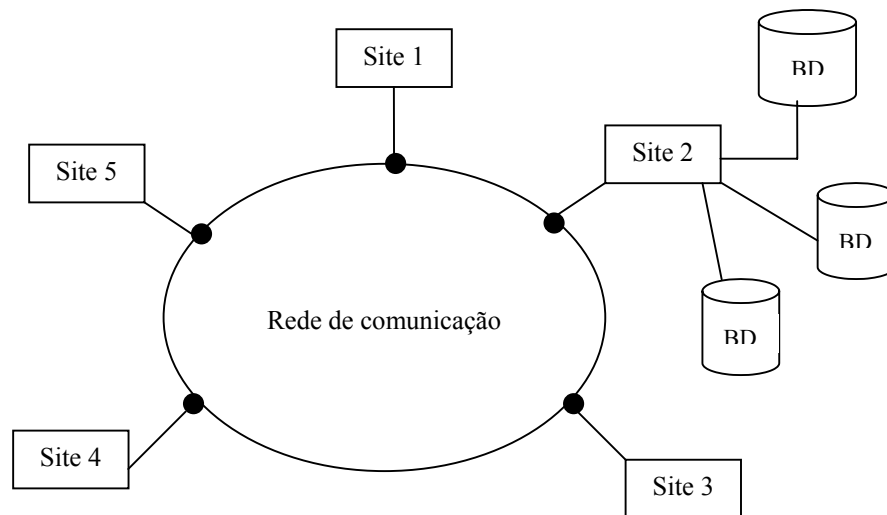


Figura 2.1 - Banco de dados central em uma

A figura 2.2 apresenta vários bancos de dados entre diferentes *sites*, e cada um dos *sites* possui um SGBD, onde trabalham de forma local. E por intermédio da rede é possível a comunicação entre os *sites*. Os SGBDs podem ser iguais em todos os sites ou diferentes em todos os sites.

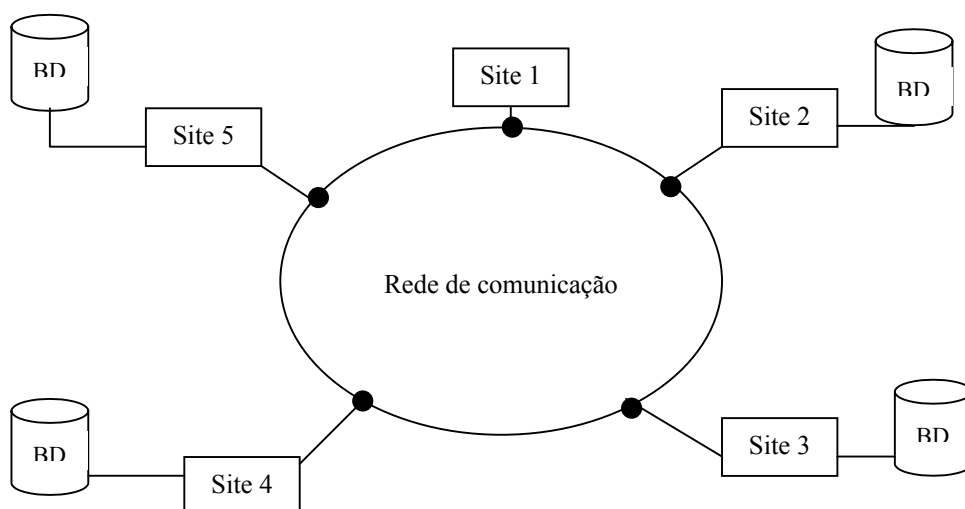


Figura 2.2 Ambiente de SBDD

2.1. TIPOS DE SGBDs DISTRIBUÍDOS

Segundo ÖZSU [1] e KORTH [2] existem dois tipos de SBDDs, o SBDD homogêneo e o SBDD heterogêneo. No SBDD homogêneo, ilustrado pela figura 2.3, os SGBDs locais são semelhantes em todos os *sites*, ou a estrutura do banco de dados é igual em todos os sites. Já no SBDD heterogêneo, ilustrado pela figura 2.4, pode existir dois ou mais SGBDs diferentes espalhados entre os *sites* ou a estrutura do banco de dados também é diferente.

“Os SGBDs homogêneos consistem em SGBDs locais que ofereçam interfaces idênticas ou pelo menos da mesma família (mesma empresa ou versões diferentes) e que forneçam os mesmos serviços aos usuários em diferentes sites.”[3]

“Os SGBDs distribuídos heterogêneos existem quando há necessidade de integrar sistemas já existentes.” [3]



Figura 2.3 SGBDs Homogêneos

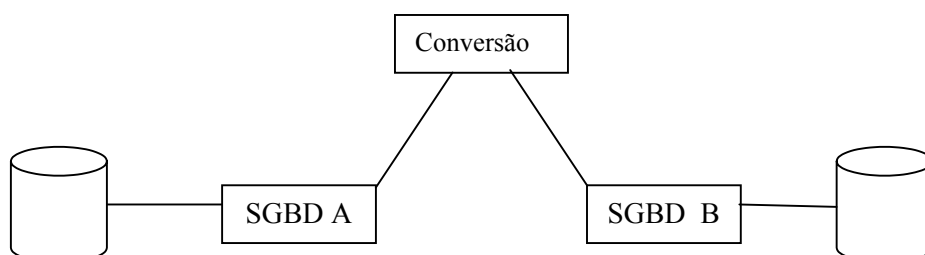


Figura 2.4 SGBDs Heterogêneos

Quando não existe homogeneidade entre os bancos de dados (tanto no SGBD quanto na estrutura), torna - se necessário fazer uma conversão entre os SBDDs. Em geral, esse mecanismo de conversão envolve uma forma canônica para facilitar a conversão de dados, bem como modelos de programas para converter instruções de manipulação de dados. Normalmente, essa heterogeneidade é introduzida quando se está construindo um SGBD distribuído a partir de vários SGBDs autônomos e centralizados. De fato, tais devem ser considerados complementares em relação aos SGBDs distribuídos.

“Existem dois tipos de distribuição dos dados no SBDD: particionar e replicar. Particionando, o banco de dados se separa em diversas partições disjuntas, e cada partição é colocada em um *site*. Replicando, pode ser totalmente ou parcialmente replicado. Totalmente replicado, cópias do banco de dados inteiro são armazenados em cada *site*. E parcialmente replicado, cada partição do banco de dados é armazenado em mais de um *site*” [1].

Existem diversos motivos que levaram ao desenvolvimento de “Sistemas de Bancos de Dados Distribuídos”. As vantagens para o desenvolvimento dos SBDDs são mostradas na seção 2.2, enquanto que na seção 2.3 são apresentadas algumas desvantagens. A seção 2.4 apresenta alguns cuidados a se terem na distribuição de dados.

2.2. VANTAGENS DOS SBDDs

Conforme ÖZSU ([1]), KORTH ([2]) e CASANOVA ([3]) existem diversas vantagens para se implantar um sistema de gerenciamento de bancos de dados distribuídos (SGBDD), das quais pode - se destacar:

- *Gerenciamento transparente de dados distribuídos.* Um sistema de bancos de dados distribuídos transparente “esconde” dos usuários os detalhes de implementação. A vantagem de um SGBD totalmente transparente é o alto nível de suporte que ele oferece para o desenvolvimento de aplicativos complexos.
- *Autonomia Local:* Utilizando – se de um SBDD é possível ter acesso a dados situados em *sites* próximos aos usuários que mais os utilizam e ao mesmo tempo, compartilhar dados armazenados em outros *sites*. Os sistemas podem ser relativamente independentes uns dos outros.
- *Integridade e Disponibilidade:* Os SGBDs distribuídos são planejados para melhorar a confiabilidade, pois têm componentes replicados e, portanto, eliminam pontos únicos de falha. A falha de um único *site* ou a falha de um único *link* (caminho de comunicação ou canal entre dois componentes ou dispositivos) de comunicação que torne um ou mais *sites* inacessíveis não é suficiente para deixar inativo o sistema inteiro.
- *Economia com hardware:* O custo com comunicação de dados diminui uma vez que os dados estão mais próximos dos usuários. Segundo ÖZSU [1], o ganho é relevante nas situações onde o volume de tráfego é tarifado (tarifas de telecomunicações). Os gastos com compra de servidores também podem ser reduzidos. A explicação para esta redução no custo acontece pelo fato de que computadores de menores valores e portes podem ser utilizados, tendo em vista que o volume de processamento é menor para os dados locais.
- *Melhoramentos de performance:* Um SBDD pode apresentar melhor performance que um sistema centralizado através da distribuição da carga de trabalho. Como cada site manipula apenas parte do banco de dados, o esforço pelos serviços da CPU (*Central Processing Unit*) e de

E/S (comunicações de entrada e saída) é bem menor do que os esforços no caso de banco de dados centralizados.

- *Expansibilidade:* Ao utilizar um SBDD, os custos associados ao aumento no número de *sites* são menores, quando comparados com os custos associados à expansão de um banco de dados centralizados.
- *Compartilhamento:* Em diversos *sites* geograficamente distribuídos e muito distantes entre si, a distribuição de dados se faz necessária já que mantê-los centralizados pode envolver um alto custo para uma empresa. Os dados distribuídos podem ser compartilhados por vários locais a um custo bem inferior.

2.3. DESVANTAGENS DOS SBDDs

Conforme ÖZSU ([1]), KORTH ([2]) e CASANOVA ([3]), existem algumas desvantagens também para se implantar um sistema de gerenciamento de bancos de dados distribuídos das quais pode-se citar:

- *Complexidade:* Os bancos de dados nos SBDD são mais complexos que os bancos de dados centralizados, pois além de existirem os problemas encontrados em um ambiente de bancos de dados centralizados, terá vários problemas não resolvidos (aspectos de modelagem, processamento, consultas, concorrência, sistema operacionais, entre outros). Estes aspectos serão discutidos na seção 2.4.
- *Custo com treinamento e aplicativos:* Existe um crescimento no custo ligados a treinamento, administração do ambiente operacional (ferramentas, técnicas, e pessoal) e desenvolvimento de aplicativos.

- *Distribuição de controle:* Mesmo sendo considerado como vantagem dos SBDDs anteriormente, a distribuição gera problemas de sincronização e coordenação. Portanto, o controle distribuído exige mais responsabilidade e cuidados já que pode se tornar facilmente uma obrigação.
- *Segurança:* Um dos mais importantes benefícios dos bancos de dados centralizados é a fiscalização sobre os acessos aos dados. Entretanto, em um sistema de bancos de dados distribuídos existe uma rede para a comunicação entre os *sítes*, e esta rede existe seus próprios requisitos de segurança. Sabe-se que manter a segurança das redes é uma tarefa séria e complexa. Desse modo, os problemas de segurança em sistemas de bancos de dados distribuídos são mais complexos do que os problemas nos sistemas de bancos de dados centralizados.
- *Falta de Experiência:* Como os sistemas de bancos de dados distribuídos já estão sendo muito utilizados, existe uma ausência de especialistas nas unidades responsáveis pelo gerenciamento de tecnologia de informação. Com isso, vários problemas podem ocorrer durante a implantação e manutenção de um sistema de bancos de dados distribuídos.

2.4. ÁREAS PROBLEMÁTICAS NOS BANCOS DE DADOS DISTRIBUÍDOS

Diversos problemas técnicos precisam ser resolvidos para alcançar o absoluto potencial dos SGBDs distribuídos, uma vez que os SGBD são muito mais complexos. Esta complexidade pode influenciar a estabilidade de um SBDD. Destacam-se alguns problemas técnicos para distribuição

de dados segundo ÖZSU ([1]), KORTH ([2]) e CASANOVA ([3]):

- *Projeto de bancos de dados distribuídos*: É o principal argumento a ser discutido ao se distribuir os dados por diversos *sítes*. Existem duas alternativas: particionar e replicar. A utilização de particionamento pode implicar estudos matemáticos para minimizar o custo de armazenamento do banco de dados pelos *sítes*, o tráfego de informações pela rede e a sincronização entre os *sítes*.
- *Processamento distribuído de consultas*: A distribuição das informações acelera a consulta, mas diminui a eficiência nas atualizações. O objetivo é otimizar os fragmentos, utilizando paralelismo para melhorar o desempenho nas consultas e atualizações.
- *Gerenciamento de diretório distribuídos*: Diretório ou catálogo do banco de dados, contém informações como estrutura e localização sobre os itens de informações no banco de dados. Um destes diretórios pode permanecer centralizado em um único local ou distribuído por vários *sítes*, onde pode haver uma cópia ou várias cópias do banco de dados e deverão estar sempre em constante atualização.
- *Controle distribuído da concorrência*: Envolve a sincronização de acessos entre os bancos de dados distribuídos, de forma que a integridade dos dados sejam mantidas. O problema do controle da concorrência em um contexto distribuído é um pouco diferente do que ocorre com a estrutura centralizada. Além de se preocupar com a integridade das informações de um único banco de dados, a consistência de várias cópias do banco de dados também é muito importante.

- Gerenciamento distribuído de impasses: A competição entre usuários pelo acesso a um conjunto de informações, pode resultar em impasse, se o mecanismo de sincronização se basear em bloqueios. As alternativas bem conhecidas de prevenção, anulação e detecção também se aplicam a SBDDs.
- Confiabilidade de SGBDs distribuídos: Foi mencionado anteriormente que uma das vantagens dos SBDDs era a maior confiabilidade e disponibilidade. É importante mencionar também os mecanismos para assegurar uma boa consistência dos bancos de dados e, também para detectar defeitos e se recompor deles. Nos casos dos SBDDs, ocorrendo um defeito em que vários *sítes* fiquem inoperantes e sem acessos, os bancos de dados existentes nos *sítes* que permanecem operacionais devem continuar estáveis e atualizados. Além disso, quando o sistema do computador ou da rede se recuperar da falha, o SBDD deve ser capaz de recuperar e manter atualizados os bancos de dados nos *sítes* em que ocorreram a falha.
- Suporte do sistema operacional: O suporte oferecido pelos sistemas operacionais para operações de bancos de dados não corresponde propriamente aos requisitos do software de gerenciamento dos bancos de dados. Os principais problemas são os sistemas de arquivos, o gerenciamento de memória, os métodos de acesso, a recuperação de falhas e o gerenciamento de processos.

2.5. CONCLUSÃO

A principal diferença entre bancos de dados (BD) centralizados e BD distribuídos é o posicionamento das bases de dados, onde no BD centralizado a base de dados está em um único local e no BD distribuído

as bases de dados estão espalhadas pelos sites da rede.

Existem alguns fatores importantes para se desenvolver um sistema de bancos de dados distribuídos, como por exemplo o gerenciamento transparente dos dados, a autonomia, a integridade, a economia com hardware, dentre outros.

Existem também algumas desvantagens para desenvolver um sistema de bancos de dados distribuídos, pois a complexidade é bem maior do que o sistema centralizado. O custo com treinamento para pessoas sem experiência e com aplicativos é alto. E a segurança é um fator importantíssimo, pois a rede do SBDD tem de ser muito bem protegida, já que a segurança de redes é muito complexa.

O próximo capítulo é descrito um projeto de um sistema de bancos de dados distribuídos, tratando de um método para distribuição dos dados.

3. PROJETO DE BANCO DE DADOS DISTRIBUÍDOS

Este capítulo destina-se à criação de um projeto de bancos de dados distribuídos envolvendo algumas técnicas para a sincronia dos dados entre os bancos espalhados por uma rede de computadores.

Este projeto é baseado em um sistema de representações comerciais, onde diversos representantes podem usar um sistema para controlar seus pedidos mediante um servidor central com uma base de dados. Para cada representante existe um terminal. Esses terminais podem ser *notebooks* (micro-computadores muito pequenos e leves, tão poderosos quanto os de mesa) e computadores PCs (Personal Computer- Computador Pessoal) comuns.

Este projeto visa fazer um controle nos pedidos realizados por diversos representantes, para que não haja uma duplicidade de dados no servidor de uma empresa. O servidor fica com o banco de dados central, e os terminais responsáveis pela administração do controle de replicação dos dados. Através dos terminais, os dados são replicados do servidor para seus respectivos discos rígidos. Para cada um dos terminais existe uma identificação, de modo que, quando uma atualização for feita no servidor, os pedidos realizados pelos terminais devem conter no código do pedido a identificação do terminal concatenado com o número do pedido. Com isso através de qualquer terminal ou do servidor, pode-se localizar de

qual terminal foi realizado o pedido. Esta situação é tratada a seguir na seção 3.2. Na seção 3.1 é mostrada a estrutura do banco de dados utilizado.

3.1. ESTRUTURA DO BANCO DE DADOS

É usado nesse projeto o sistema de SGBDs Homogêneo, onde que, para cada terminal e também o servidor, têm o mesmo SGBD e a mesma estrutura para o banco de dados. O SGBD utilizado é o FireBird, por ser gratuito à vários usuários, de fácil utilização e pela sua ótima capacidade de gerenciamento de dados. Mas em qualquer plataforma pode ser implementado o método desenvolvido.

O diagrama de entidade relacionamento (DER) apresentado na figura 3.1 mostra o que é usado na implementação do projeto de SBDDs. A estrutura do banco de dados possui as seguintes tabelas: **Fornecedor**, **Cliente**, **Produto**, **Item**, **FichaID**, **Usuario**, **IDTerminal**, **Pedido**, **Representada**, **Vendedor**, **Cidades**, **Atualizacao** e **VendedorCidade**.

- Tabela **Fornecedor**: Contém as informações referentes aos fornecedores cadastrados.
- Tabela **Clientes**: Contém as informações referente a todos os clientes cadastrados.
- Tabela **FichaID**: Contém cadastradas as identificações dos clientes, se forem registradas como empresas. Por exemplo, empresa matriz.
- Tabela **Produto**: Contém as informações de todos os produtos que existem em uma determinada representada.
- Tabela **Pedido**: Contém todos os pedidos realizado pelo

representante.

- Tabela **Item**: Contém todos os produtos a serem emitidos em um pedido.

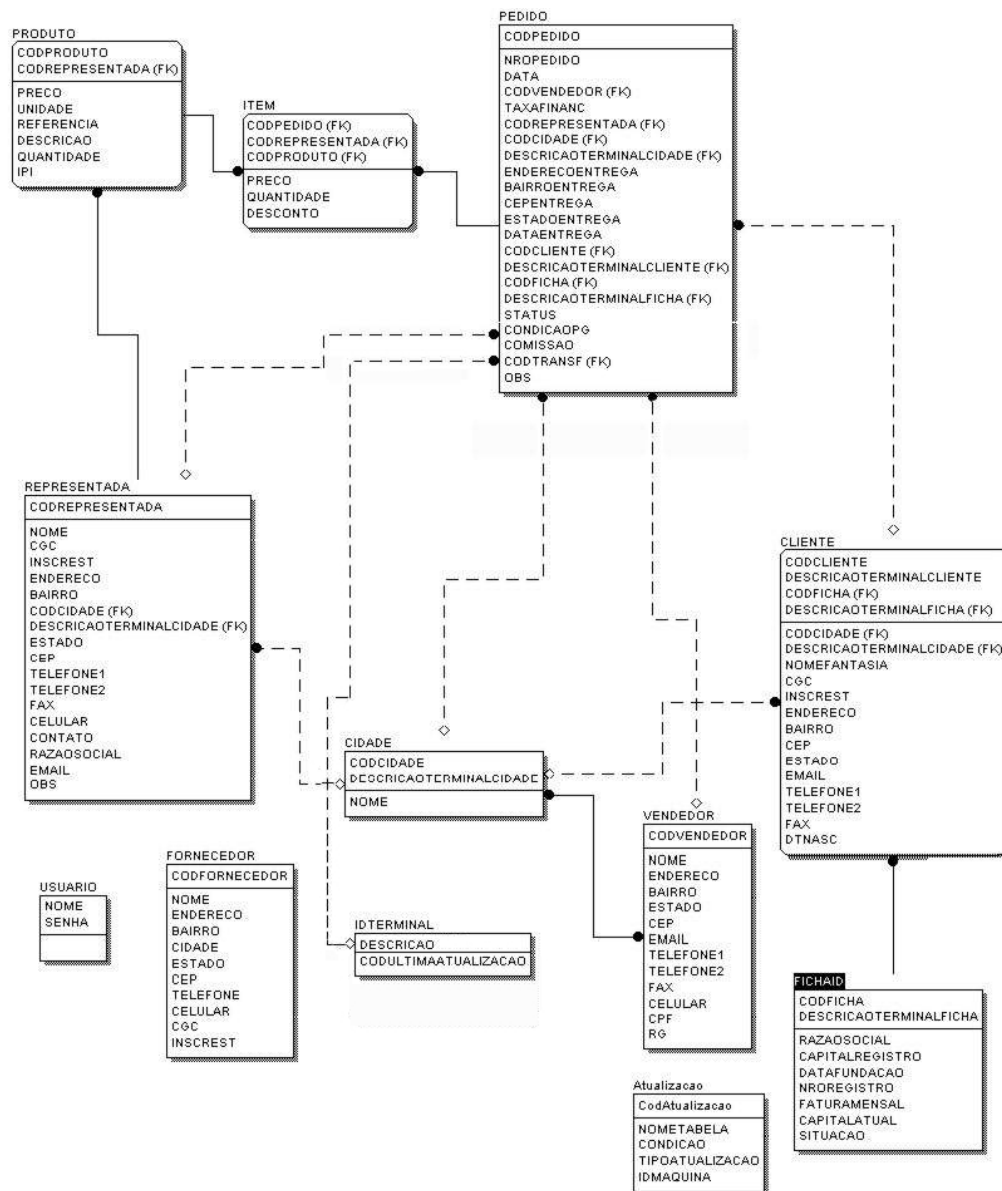


Figura 3.1 - DER Lógico do Banco de Dados

- Tabela **Representada**: Contém as informações cadastrais das representadas.

- Tabela **Vendedor**: Contém as informações dos vendedores da empresa de representação.
- Tabela **Cidade**: Contém todas as cidades onde um determinado vendedor faz visitas.

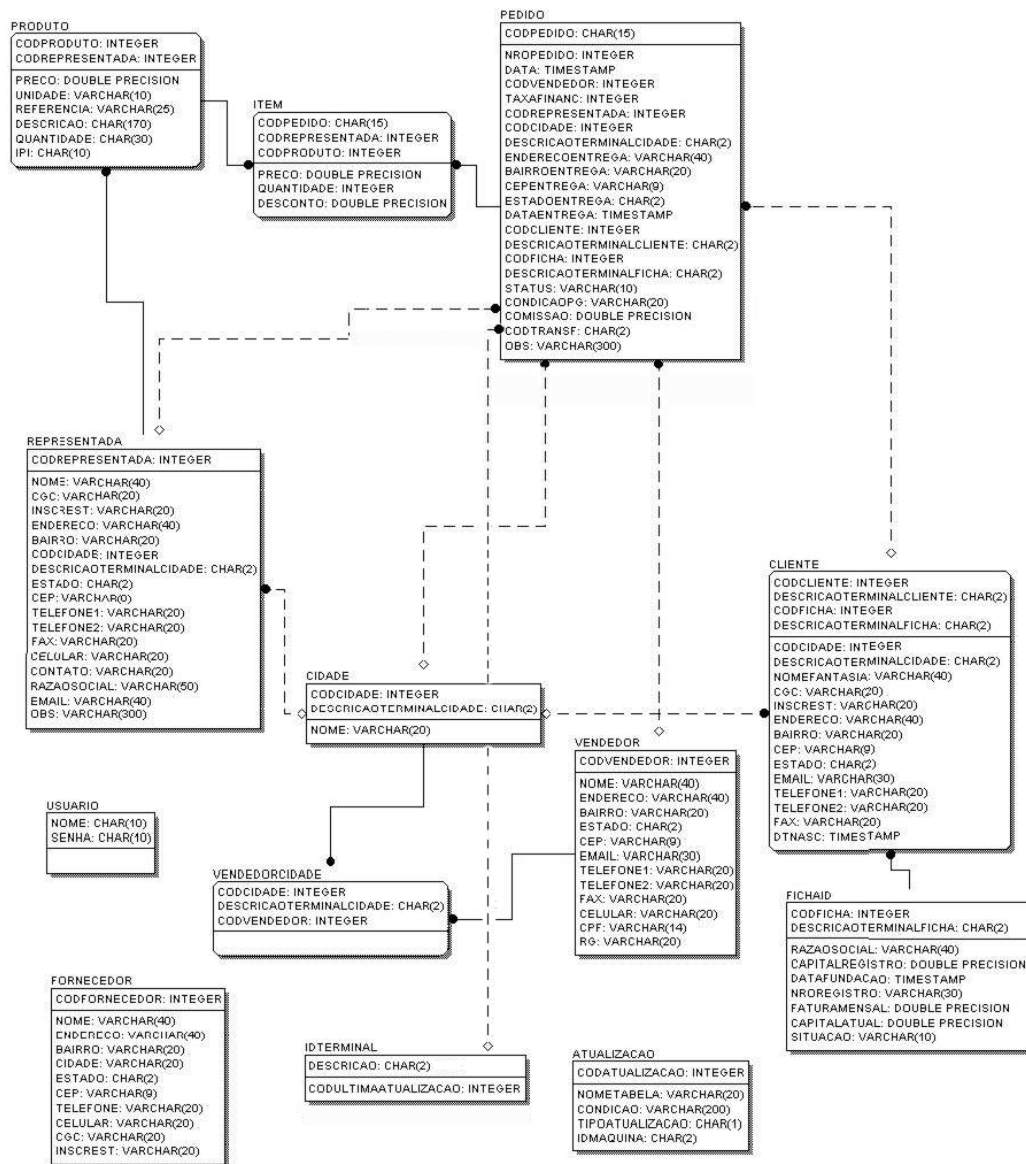


Figura 3.2 - DER Físico do Banco de Dados

- Tabela **VendedorCidade**: Contém informações de ligação entre a

tabela **Cidade** e a tabela **Vendedor**, pois pode haver em uma cidade vários vendedores (representantes) e um vendedor representar várias cidades.

- Tabela **Usuário**: Contém a identificação do usuário, esta tabela é responsável para iniciar o programa, onde cada vendedor tem um nome e uma senha de acesso ao sistema.
- Tabela **IDTerminal**: Contém as informações armazenadas dos terminais, dito anteriormente.
- Tabela **Atualizacao**: Esta tabela é a principal do projeto, ela é responsável pelo controle de replicação e atualização dos bancos.

A figura 3.2 apresenta o modelo de dados físico, onde é demonstrado os tipos de dados que cada atributo das tabelas.

3.2. CONTROLE DOS IDENTIFICADORES

O DER da seção 3.1, representado pelas figuras 3.1 e 3.2, contém atributos específicos para controlar os índices dos registros, sem que eles sejam duplicados na tabela.

Na tabela **Pedido**, existe um atributo de nome *CodTransf* onde é gravado a identificação do terminal (tabela **IDTerminal**, atributo *Descricao*). O valor de *CodPedido* é igual a concatenação de *CodTrans* com o próximo número do pedido (*NroPedido*) a ser gerado neste terminal. Por exemplo, suponha que a identificação do terminal seja igual a 'T1' e o próximo número do pedido a ser gerado na tabela igual a '12345'. O valor a ser gravado no atributo *CodPedido* ao fazer um pedido no sistema, é igual a 'T112345'. Assim, na hora de atualizar o servidor, não ocorre o risco de

ter dados com o mesmo código, já que cada terminal tem uma identificação diferente.

Na tabela **Cidade**, existe um atributo de controle da chave primária chamado *DescricaoTerminalCidade*. A informação que fica armazenada neste atributo é a identificação do terminal. Por exemplo, *CodCidade* igual a 1 e *DescricaoTerminalCidade* igual a T1. Assim, na hora de atualizar os bancos, não ocorre duplicação de chave primária, pois o que ocorre no sistema é que para cada tabela **Cidade** dos terminais, o atributo *CodCidade* recebe valor sequencial. Se a chave primária fosse somente o atributo *CodCidade*, ao atualizar todos os bancos ocorreria um erro de duplicidade. Assim sendo, o atributo *DescricaoTerminalCidade* faz o papel de distinguir um código do outro, mantendo a integridade dos dados.

Essa forma de controle citada anteriormente para controlar a integridade dos dados se aplica também nas tabelas **FichaID** e **Cliente**. Nestas tabelas também existem um atributo de controle, *DescricaoTerminalFicha* e *DescricaoTerminalCliente* respectivamente.

3.3. CONTROLE DOS CADASTROS

Os cadastros existentes no projeto contém algumas restrições para os terminais. No servidor é possível fazer todos os cadastros existentes, podendo também fazer alterações e exclusões. Já nos terminais só podem ser feitos cadastros, alterações e exclusões em clientes, cidades, fichas e principalmente os pedidos, pois um representante geralmente não tem autorização para cadastrar produtos, representadas, fornecedores e vendedores em uma empresa. Assim, estes cadastros são exclusivos para quem faz o gerenciamento do servidor na empresa.

Ao fazer um cadastro nas tabelas **FichaID**, **Cliente**, **Pedido** e **Cidade**, tabelas de comum ação (inserir, atualizar e excluir) para servidor e terminal, é gravado automaticamente no registro em que se está cadastrando, a identificação da máquina nos atributos: *DescricaoTerminalFicha*, *DescricaoTerminalCliente*, *CodTransf* e *DescricaoTerminalCidade* respectivamente. Essa identificação faz parte da chave primária destas tabelas junto com seus respectivos códigos sequenciais.

Para cada inserção, atualização ou exclusão que fizer no sistema, é gravado automaticamente na tabela **Atualizacao**, um registro contendo o nome da tabela em que a ação está sendo exercida, a condição (para que seja possível identificar qual registro está sendo inserido, modificado ou excluído no outro banco de dados em que se deseja sincronizar), o tipo de atualização (“I” para inserir, “U” para atualizar e “D” para excluir) e a identificação da máquina (terminal ou servidor). A figura 3.3 demonstra como fica a tabela **Atualizacao** do terminal após uma ação de inserção e outra de atualização na tabela **Cidade**. A partir desses cadastros o sistema poderá sincronizar os bancos.

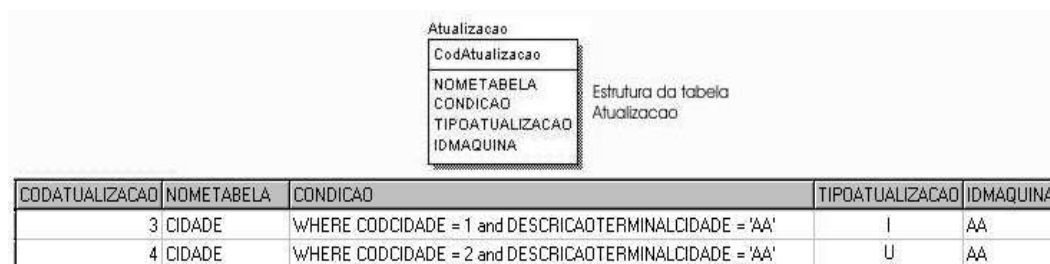


Diagrama da estrutura da tabela **Atualizacao**:

Atualizacao				
CodAtualizacao				
NOMETABELA				
CONDICAO				
TIPOATUALIZACAO				
IDMAQUINA				

Estrutura da tabela Atualizacao

CODATUALIZACAO	NOMETABELA	CONDICAO	TIPOATUALIZACAO	IDMAQUINA
3	CIDADE	WHERE CODCIDADE = 1 and DESCRICAO TERMINALCIDADE = 'AA'	I	AA
4	CIDADE	WHERE CODCIDADE = 2 and DESCRICAO TERMINALCIDADE = 'AA'	U	AA

Figura 3.3 – Demonstração do conteúdo da tabela **Atualizacao** do banco do terminal

Na seção 3.4 é descrito o processo de atualização dos bancos através da tabela **Atualizacao**.

3.4 ATUALIZAÇÃO DOS BANCOS DE DADOS

Para a atualização do banco de dados, tanto no servidor como nos terminais, existe uma tabela muito importante para esse processo, a tabela **Atualizacao**. Nessa tabela é armazenado o que deve ser feito no servidor e no terminal que chamou a rotina de atualização.

Conforme ilustrado na figura 3.3 da seção 3.3, na tabela **Atualizacao** de cada banco é gravado no atributo *NomeTabela* o nome da tabela a ser atualizada. Isso acontece caso o sistema execute as funções *insert* (comando para inserção de dados no banco), *update* (atualização de dados já existente no banco) ou *delete* (exclusão de dados no banco).

No atributo *Condicao*, é gravado a condição que será necessária para a atualização, por exemplo “*where codCidade = 2 and DescricaoTerminalCidade = 'AA'*”.

No atributo *TipoAtualizacao* é gravado o que deve ser feito no banco em que ocorrerá a atualização. Se tiver gravado no atributo *TipoAtualizacao* a letra ‘I’, deve ser inserido no outro banco o registro em que apontar a condição (atributo *Condicao*). Se for a letra ‘U’, deve ser atualizado o registro em que a condição apontar. E se for a letra ‘D’, deve ser excluído o registro em que a condição estiver apontando.

No atributo *IDMaquina*, é armazenado a identificação da máquina que executou determinada função.

Com esses procedimentos feitos, o sistema tem condições de fazer o

controle de replicação dos dados evitando que os dados sejam duplicados.

Para cada uma dessas ações, existe um código na tabela **Atualizacao**. Quando uma replicação é feita do servidor para um terminal, grava-se o ultimo código da tabela **Atualizacao** no atributo *codUltimaAtualizacao* da tabela **IDTerminal** do terminal. Isso ocorre porque quando precisar replicar novamente, o sistema do terminal irá começar a replicar a partir desse código, evitando que os registros já atualizados anteriormente sejam lidos novamente. Enquanto que no servidor, após a replicação, o sistema apaga todos os registros da tabela **Atualizacao** do banco do terminal, fazendo com que na próxima vez que for replicar, o banco esteja somente com o que ainda não foi atualizado.

Para o cadastro de produtos, fornecedores, representada e vendedor, apenas é possível fazer esse cadastro no servidor.

3.5 CONTROLE DE REPLICAÇÃO DE DADOS

Para controlar a replicação de dados no sistema, duas funções principais fazem parte da rotina, *ReplicaRemoto* e *ReplicaLocal*.

A função *ReplicarRemoto* é responsável por atualizar o banco de dados dos terminais a partir do servidor, onde todos os registros das tabelas que não estão no terminal e estão no servidor, são replicado para o terminal. Dessa forma o banco de dados do terminal está completo e atualizado.

Primeiramente, o sistema faz uma busca na tabela **IDTerminal** do banco

do terminal obtendo o conteúdo do atributo *codUltimaAtualizacao* e gravando em uma variável *X*.

X = Select codUltimaAtualizacao from IDTerminal

Em seguida, o sistema precisa fazer uma seleção na tabela **Atualizacao** do banco do servidor, buscando todos os registros em que o *codAtualizacao* seja maior do que o valor gravado na variável *X* e o atributo *IDMaquina* seja diferente da identificação do terminal. Após isso, o sistema faz uma varredura em cada um dos itens selecionados e faz uma nova seleção, buscando os registros no banco do servidor de acordo com cada condição apontada no atributo *Condicao*. A seguir está sendo demonstrado como o sistema faz essa seleção:

*Select * from [atributo NomeTabela da tabela **Atualizacao** do banco do Servidor] [atributo Condicao da tabela **Atualizacao** do banco do Servidor]*

CODATUALIZACAO	NOMETABELA	CONDICAO	TIPOATUALIZACAO	IDMAQUINA
3	CIDADE	WHERE CODCIDADE = 1 and DESCRICAOterminalCIDADE = 'SS'	I	SS
4	CIDADE	WHERE CODCIDADE = 2 and DESCRICAOterminalCIDADE = 'SS'	U	SS

Figura 3.4 –Conteúdo da tabela **Atualizacao** do Servidor

Baseando se na figura 3.4, a seleção seria feita da seguinte forma:

*Select * from **Cidade** where CodCidade = 1 and
DescricaoTerminalCidade = 'SS'*

Após selecionar o registro, o sistema verifica qual informação contém no

atributo *TipoAtualizacao*. Se o conteúdo deste atributo for:

- “I” - o sistema primeiramente faz uma consulta para verificar se já existe o registro na tabela de nome igual ao conteúdo do atributo *NomeTabela* no banco do terminal. Se não existir, o sistema insere o registro selecionado do banco do servidor no banco do terminal. Caso exista, o sistema faz uma alteração no conteúdo do atributo *TipoAtualizacao* para ‘U’.
- “U” - o sistema busca no banco do terminal o registro na tabela em que o atributo *NomeTabela* apontar e *Condicao*. Após selecionado o registro, o sistema faz as alterações igualando o registro do banco do terminal com o banco do servidor.
- “D” - o sistema busca no banco do terminal o registro na tabela em que o atributo *NomeTabela* apontar e *Condicao*. Após selecionado o registro, o sistema vai apaga-lo.

Com esses procedimentos o banco do terminal fica atualizado com o banco do servidor.

A função *ReplicarLocal* atualiza o banco de dados do servidor a partir do terminal, onde todos os registros das tabelas que não estão no servidor e estão no terminal são replicado para o servidor. Desse modo o banco de dados do servidor fica completo e também atualizado.

O sistema faz da mesma forma que na função *ReplicarRemoto* para atualizar o banco do servidor, só que ao invés de *CodAtualizacao* for maior do que o *CodUltimaAtualizacao*, o sistema considera que o

CodUltimaAtualizacao seja igual a zero. Assim, todos os registros que estiverem na tabela **Atualizacao** do terminal, são atualizados no banco do servidor. Da mesma forma, o sistema lê os registros na tabela **Atualizacao** e executa todas as ações em que o atributo *TipoAtualizacao* informar ('I', 'U' e 'D').

Após executar as funções *ReplicarRemoto* e *ReplicarLocal*, os dados da tabela **Atualizacao** do terminal tem de serem excluídos, para que da próxima vez não seja atualizados os dados antigos.

Desta forma todos os registros que estiverem em um banco, estão também no outro, deixando os bancos idênticos.

3.6 CONCLUSÃO

O tipo do sistema usado no projeto é o sistema de SGBDs Homogêneo, onde não é preciso converter nenhuma estrutura do banco de dados e nem mudar as configurações dos SGBDs quando diferentes.

O método utilizado para o controle dos dados é pouco complexo, pois é necessário atributos e tabelas especiais para garantir a integridade dos dados e um bom controle para replicação dos dados. É um método eficiente e de baixo custo, o que proporciona um ótimo benefício para pequenas, médias e até grandes empresas.

No capítulo 4 é apresentado um estudo de caso envolvendo as técnicas e métodos descritos nesse capítulo.

4. ESTUDO DE CASO

Este capítulo descreve um estudo de caso sobre as técnicas e métodos mostrados no capítulo 3. Esse estudo de caso é demonstrado por um projeto desenvolvido, utilizando Delphi e FireBird, de um sistema de representações. O nome do projeto é “Representação” e tem como funcionalidade auxiliar empresas e representantes, numa sincronia entre suas informações.

4.1 CADASTROS

Nesta seção é descrito como ocorrem os cadastros existentes no sistema. A figura 4.1 ilustra o menu Cadastro com todos os cadastros existentes e também as alterações dos cadastros, quando estes estiveram incorretos.



Figura 4.1 – Menu Cadastros

4.1.1 CADASTRAR USUÁRIOS

O cadastro de usuários somente serve para ter acesso ao sistema Representação.

4.1.2 CADASTRAR IDENTIFICAÇÃO

A identificação é um pseudônimo que recebe os terminais (notebooks) e também o servidor. O tamanho dessa identificação é de apenas dois caracteres, como por exemplo "AA". Essa identificação é necessária para se fazer o controle de atualizações dos bancos de dados entre os terminais e o servidor, esse controle é descrito na seção 4.4.

4.1.3 CADASTRAR CIDADES

Primeiramente, para se fazer os cadastros deve-se cadastrar as cidades, pois as cidades são utilizadas para o cadastramento de clientes, representada e pedido.

Depois de ter informado o nome da cidade, este é gravado na tabela **Cidade** no atributo *Nome*. Para o atributo *CodCidade* é gerado um código por auto incremento. E para o atributo *DescricaoTerminalCidade* é gravado a identificação do terminal ou o servidor. Na tabela **Atualizacao** é executado o seguinte comando:

```
"Insert into Atualizacao (CodAtualizacao, NomeTabela, Condicao, TipoAtualizacao, IDMaquina) values (gen_id(GEN_ATUALIZACAO, 1), "Cidade", "Where CodCidade = (new.codcidade as varchar(9) " and
```

DescricaoTerminalCidade = new.DDescricaoTerminalCidade, "I", :vat)

onde:

- *CodAtualizacao* – é um auto incremento da própria tabela;
- *NomeTabela* – é igual a *Cidade*;
- *Condicao* – é igual a “where codCidade = ao código da nova cidade inserida and *DescricaoTerminalCidade* = a identificação da máquina”;
- *TipoAtualizacao* – é igual “I”;
- *IDMachina* – é igual a identificação da máquina.

Esse registro é necessário para quando for atualizar os bancos, pois através desse registro o sistema interpreta que o registro na tabela ***Cidade*** tem de ser inserido na tabela ***cidade*** da outra máquina.

4.1.4 CADASTRAR CLIENTES

Esse módulo é utilizado para se cadastrar os clientes de um representante em seu computador. A figura 4.2 ilustra a tela do sistema onde o módulo de cadastro funciona. Em primeiro lugar, é preciso cadastrar a “Ficha” do cliente. Esta ficha é uma identificação extra do cliente. Se o cliente é filial de uma outra empresa, esta utiliza a ficha da empresa matriz. Caso já exista a ficha para aquele cliente, basta selecionar a ficha desejada e prosseguir com o restante do cadastro. Caso a ficha não existir no banco de dados do sistema, tem de se cadastrar uma nova ficha.

Tanto o cadastro de fichas como o cadastro de clientes, são gravados nos atributos *CodFicha* e *CodCliente* os códigos de identificação, que são gerados por auto incremento. Já nos atributos *DescricaoTerminalFicha* e

DecricaoTeminalCliente é gravado a identificação da máquina. Também para essas duas tabelas, conforme demonstrado anteriormente em cadastro de cidades, ao inserir um registro, é inserido na tabela **Atualizacao** um registro para controlar a atualização dos bancos, com o nome das tabelas e com as determinadas condições, códigos e identificação da máquina.

Figura 4.2 – Cadastro de Clientes

4.1.5 CADASTRAR VENDEDORES

O cadastro de vendedor é também semelhante aos cadastros anteriores citados, ele também possui um auto incremento para gerar seu código e também registra na tabela **Atualizacao** conforme o exemplo citado em cadastrar cidades. Esse cadastro somente é possível de se realizar no servidor, pois é no servidor que se organiza as funções dos terminais para seus representantes. No sistema dos terminais, a opção no menu Cadastrar - 'Vendedor' é bloqueada.

4.1.6 CADASTRAR FORNECEDORES

Esse módulo serve para o cadastramento de fornecedores. Basta informar

os dados do fornecedor e gravar. Esse módulo, assim com o vendedor, só é disponibilizado para cadastro no servidor.

Semelhante ao cadastro de vendedor, o processo de cadastramento é o mesmo, o código do fornecedor é gerado a partir de um auto incremento no atributo *CodFornecedor* e os demais registros são gravados nos respectivos atributos. Na tabela ***Atualizacao*** o mesmo é feito, atributo:

- *NomeTabela* – é igual a *Fornecedor*;
- *Condicao* – é igual “where codFornecedor = novo código do fornecedor”;
- *TipoAtualizacao* – é igual a ‘I’;
- *IDMaquina* – é igual a identificação da máquina.

4.1.7 CADASTRAR REPRESENTADAS

Ao se cadastrar as representadas (somente pelo servidor), da mesma forma dos cadastros anteriores, é gerado o código da representada por auto incremento, em seguida grava-se na tabela ***Atualizacao*** o registro contendo o nome da tabela, condição, o tipo da atualização e a identificação da máquina. Reforçando que esse processo de inserção de dados na tabela ***Atualizacao*** é de muita importância, pois é através dessa tabela que será possível uma ótima replicação dos dados.

4.1.8 CADASTRAR PRODUTOS

Depois de cadastrada a representada pode e cadastrar os produtos, já

que para cada produto cadastrado, deve-se ter uma representada associada. Esse módulo é bastante simples, basta informar as informações do produto, quantidade em estoque, preço e valor do IPI. A figura 4.3 mostra a tela do módulo de cadastro de produtos.

Ao gravar, o sistema grava na tabela **Produto** o código do produto que também é auto incremento, o código da representada selecionada e as demais informações. Também nesse módulo grava-se um registro na tabela **Atualizacao**, contendo nome da tabela, a condição, o tipo de atualização e também a identificação da máquina.

Figura 4.3 – Cadastro de Produtos

Após o cadastramento de alguns dados, algumas tabelas ficam como no exemplo ilustrado pela figura 4.4 e pela figura 4.5 das tabelas **Produto** e **Cidade**:

	CODPRODUTO	PRECO	UNIDADE	CODREPRESENTADA	REFERENCIA	DESCRICAO	QUANTIDADE	IPI
▶	1	3000	PC		1 10001	Pentium 4 2.0 ghz	5	0
	1	4200	PC		2 20001	NoteBook Pentium 4 1.8 ghz	3	0
	2	1552	PC		1 10002	Pentium 4 1.2 ghz	10	0
	2	3100	PC		2 20002	PC Pentium 4 2.4 ghz	2	0

Figura 4.4 – Conteúdo da tabela **Produto**

	CODCIDADE	DESCRICAO	TERMINALCIDADE	NOME
▶	1	AA		Perdizes
	2	AA		Uberlândia

Figura 4.5 – Conteúdo da tabela **Cidade**

4.2 ATUALIZAÇÕES E EXCLUSÕES

Os módulos de alterações são bem simples. No menu Cadastrar do sistema, mostrada na figura 4.1, existe um item do menu que é ‘Alterações’. Esse item do menu Cadastrar é também um sub-menu e contém as chamadas dos módulos de alterações.

- Alteração e exclusão em Fichas – Ao alterar o registro, o sistema faz a alteração no banco de dados e insere na tabela **Atualizacao** um registro contendo o nome da tabela (**FichaID**), a *condição* (*where codFicha = a ficha selecionada*), o tipo de atualização que nesse caso é a letra “U” e a identificação da máquina. Caso queira excluir, o sistema apaga esse registro da tabela **FichaID** e insere na tabela **Atualizacao** um registro semelhante ao registro anterior, só que, no atributo *TipoAtualizacao*, ao invés de conter a letra “U” será a letra “D”.
- Alteração e exclusão em Clientes –Depois de selecionado o cliente, escolha entre alterar ou excluir e o sistema faz o mesmo processos de alteração e exclusão da ficha.
- Alterações e exclusões em *Vendedores*, *Fornecedores*, *Produtos* e *Representadas* – Em todos esses módulos, o processo de escolha do registro, forma de alteração e exclusão são idênticas aos processos de

fichas e clientes. Porém existe um detalhe importante, também só é feito no servidor.

- Alterações e exclusões em Cidades – Após alterar o nome da cidade, o sistema vai da mesma forma que nas demais atualizações, inserir um registro na tabela **Atualizacao**, onde nome da tabela igual a **Cidade**, condição igual a “where codCidade igual ao código da cidade selecionado” e tipo de atualização igual a “U”. Para excluir utiliza-se do mesmo processo, selecione a cidade e exclui. A inserção na tabela **Atualizacao** também é feita, só que com o tipo de atualização igual a “D”.

Após feitas alguns cadastros, alterações e exclusões, a tabela **Atualizacao** contém alguns itens. Na figura 4.6 ilustra a situação da tabela:

CODATUALIZACAO	NOMETABELA	CONDICAO	TIPOATUALIZACAO	IDMAQUINA
1	FICHAID	WHERE CODFICHA = 1 and DESCRICAOTERMINALficha = 'AA'	I	AA
2	FICHAID	WHERE CODFICHA = 2 and DESCRICAOTERMINALficha = 'AA'	I	AA
3	CIDADE	WHERE CODCIDADE = 1 and DESCRICAOTERMINALCIDADE = 'AA'	I	AA
4	CIDADE	WHERE CODCIDADE = 2 and DESCRICAOTERMINALCIDADE = 'AA'	I	AA
5	CLIENTE	WHERE CODCLIENTE = 1 and DESCRICAOTERMINALCiente = 'AA' e I		AA
6	CLIENTE	WHERE CODCLIENTE = 2 and DESCRICAOTERMINALCiente = 'AA' e I		AA

Figura 4.6 – Conteúdo da tabela **Atualizacao**

4.3. VENDAS

Nesta seção é descrito como funciona o módulo de pedidos existente no sistema. A figura 4.7 ilustra o menu ‘Vendas’.



Figura 4.3.1 – Menu Vendas

4.3.1 CADASTRAR PEDIDO

No menu 'Vendas' existe um sub-menu 'Pedidos'. Em 'Pedidos' existe 'Criar Pedido' e 'Alterar Pedido'. Clicando em 'Criar Pedido' abre a tela de cadastros de pedidos ilustrado pela figura 4.8.

O pedido a ser gerado tem um número de pedido sequencial, onde esse número serve para controle da empresa e dos representantes. Para se fazer o pedido, primeiramente deve-se escolher a ficha do cliente a ser feito o pedido. Depois de escolhido a ficha do cliente, deve-se escolher o cliente daquela ficha. Após feito isso, preenche o pedido com o restante das informações.

A interface 'Cadastros de Pedidos' apresenta os seguintes campos e seções:

- Campos de Cadastro:**
 - Ficha: (dropdown)
 - Cliente: (dropdown)
 - Status: (dropdown)
 - Endereço de Entrega: (text)
 - Bairro de Entrega: (text)
 - UF de Entrega: (dropdown)
 - CEP de Entrega: (text)
 - Cidade de Entrega: (dropdown)
 - Data para Entrega: (text)
 - Data do Pedido: (text)
 - Condição de Pagamento: (text)
 - Vendedor: (dropdown)
 - Comissão (%): (text)
 - OBS: (área de texto grande)
 - Representada: (dropdown)
 - Produtos da Representada: (dropdown)
- Itens do Pedido:** (tabela com 9 colunas: Referência, Descrição, Unidade, Preço / Und, Quantidade, Preço Total / Und, Desconto, IPI, Total)
- Barra de Resumo:**
 - NÚMERO DO PEDIDO: (campo de texto)
 - TOTAL: (campo de texto com valor 0,00)
 - TOTAL COM DESC. E IPI: (campo de texto com valor 0,00)
 - Botões: Fechar Pedido, Gravar, Limpar

Figura 4.8 – Cadastros de Pedidos

Para inserir os itens dos pedidos, primeiro é necessário escolher a

representada em que o produto se associa. Para cada produto de outra representada, deve-se escolher a representada correspondente e inserir o produto na lista de itens.

Quando estiver tudo certo para o fechamento do pedido, basta clicar no botão 'Gravar' para registrar o pedido. Ao gravar o pedido o sistema concatena a identificação da máquina com o número do pedido para ser o valor do código do pedido. Por exemplo, o número do pedido igual a 10002 e a identificação da máquina igual a 'AA', gerando um valor para o código do pedido igual a 'AA10002'. Esse procedimento é para não ocorrer duplicidade nos dados quando for atualizar os banco de dados.

Os produtos selecionados para o pedido são gravados todos na tabela **Item**. Juntamente com o registro dos produtos, também é gravado o código do pedido.

Do mesmo modo dos outros cadastros citados anteriormente, o pedido também insere na tabela **Atualizacao** um registro contendo o nome da tabela (**Pedido**), a condição (*where codPedido = concatenação da identificação da máquina com o número do pedido*), tipo de atualização ('I') e a identificação da máquina.

Os itens do pedido também são gravados na tabela **Atualizacao**, os registros, nome da tabela (**Item**), a condição (*where CodPedido = concatenação da identificação da máquina com o número do pedido and CodProduto = ao produto selecionado no pedido and CodRepresentada = ao código da representada do produto*), tipo de atualização ('I') e a identificação da máquina.

Depois de todos esse procedimentos de inserção, a tabela **Atualizacao** fica da seguinte forma, ilustrada pela figura 4.9:

CODATUALIZACAO	NOMETABELA	CONDICAO	TIPOATUALIZACAO	IDMAQUINA
3	PEDIDO	WHERE CODPEDIDO = 'AA1'	I	AA
4	PEDIDO	WHERE CODPEDIDO = 'AA2'	I	AA

Figura 4.9 – Conteúdo da tabela **Atualizacao** após inserção de pedidos

4.3.2 ALTERAR PEDIDO

Ao efetuar as alterações, também é inserido um registro na tabela **Atualizacao**, onde todos dados são iguais aos mencionados acima na inserção do pedido, somente com uma diferença, o tipo de atualização será igual a “U”, o que significa atualização.

4.3.3 CANCELAR PEDIDO

Não sendo diferente dos outro módulos, ao cancelar o pedido o sistema insere na tabela **Atualizacao** os mesmo dados da inserção ou da alterações, sendo que o tipo de atualização será igual a “D”.

4.4 SINCRONIZAÇÃO DOS BANCOS DE DADOS

Esta seção é a principal do projeto e a mais importante, pois é descrito como ocorre a sincronia e atualizações das tabelas no banco de dados do terminal e no banco de dados do servidor. A figura 4.10 ilustra o menu *Sincronizar BD* com as opções *Replica Servidor*, *Replica Terminal* e *Replica Servidor / Terminal*.



Figura 4.10 – Menu Sincronizar BD

A opção *Replica Servidor / Terminal* ajusta tanto o banco do terminal quanto o do servidor. A opção *Replica Servidor* ajusta somente o banco do terminal e a opção *Replica Terminal* ajusta somente o banco do servidor.

É descrito a seguir como é o funcionamento desses procedimentos, separadamente.

4.4.1 ATUALIZANDO O BANCO DE DADOS DO TERMINAL

A sincronia dos bancos parte dos terminais. Os terminais ligados a uma rede de computadores, são quem solicitam a sincronia. Primeiramente o sistema busca no banco de dados do servidor, na tabela ***Atualizacao***, todos os registros em que o atributo *IDMaquina* seja diferente da identificação do terminal e o *CodAtualizacao* seja maior do que o atributo *CodUltimaAtualizacao* (atributo da tabela ***IDTerminal*** do banco do terminal onde é gravado o ultimo código da ultima atualização). Essa condição é necessária para que o processo de atualização não busque dados que já foram atualizados. Dessa forma a sincronia dos dados se torna mais rápida.

Por exemplo, a descrição do terminal é 'AA' e o código do último registro

atualizado neste terminal é 8. O sistema busca na tabela **Atualizacao** do servidor todos os registros em que o atributo *IDMaquina* é diferente de 'AA' e *CodAtualizacao* maior do que 8. O resultado da consulta é ilustrado pela figura 4.11.

CODATUALIZACAO	NOMETABELA	CONDICAO	TIPOATUALIZACAO	IDMAQUINA
9	VENDEDOR	WHERE CODVENDEDOR = 1	I	SS
10	VendedorCidade	WHERE CODCIDADE = 2 and DESCRICAO TERMINALCidade = 'SS'	I	SS
11	REPRESENTADA	WHERE CODREPRESENTADA = 1	I	SS
12	REPRESENTADA	WHERE CODREPRESENTADA = 2	I	SS
13	PRODUTO	WHERE CODProduto = 1 and CodRepresentada = 1	I	SS
14	PRODUTO	WHERE CODProduto = 2 and CodRepresentada = 1	I	SS
15	PRODUTO	WHERE CODProduto = 1 and CodRepresentada = 2	I	SS
16	PRODUTO	WHERE CODProduto = 2 and CodRepresentada = 2	I	SS
17	FICHAID	WHERE CODFICHA = 1 and DESCRICAO TERMINALFicha = 'AA'	I	SS
18	FICHAID	WHERE CODFICHA = 2 and DESCRICAO TERMINALFicha = 'AA'	I	SS
19	CIDADE	WHERE CODCIDADE = 1 and DESCRICAO TERMINALCidade = 'AA'	I	SS
20	CIDADE	WHERE CODCIDADE = 2 and DESCRICAO TERMINALCidade = 'AA'	I	SS
21	CLIENTE	WHERE CODCLIENTE = 1 and DESCRICAO TERMINALCiente = 'AA'	I	SS
22	CLIENTE	WHERE CODCLIENTE = 2 and DESCRICAO TERMINALCiente = 'AA'	I	SS
23	Fornecedor	WHERE CODFornecedor = 1	U	SS
24	Fornecedor	WHERE CODFornecedor = 2	U	SS
25	FICHAID	WHERE CODFICHA = 1 and DESCRICAO TERMINALFicha = 'SS'	U	SS
26	FICHAID	WHERE CODFICHA = 2 and DESCRICAO TERMINALFicha = 'SS'	U	SS
27	CIDADE	WHERE CODCIDADE = 1 and DESCRICAO TERMINALCidade = 'SS'	U	SS
28	CIDADE	WHERE CODCIDADE = 2 and DESCRICAO TERMINALCidade = 'SS'	U	SS
29	CLIENTE	WHERE CODCLIENTE = 1 and DESCRICAO TERMINALCiente = 'SS'	U	SS
30	CLIENTE	WHERE CODCLIENTE = 2 and DESCRICAO TERMINALCiente = 'SS'	U	SS

Figura 4.11 – Conteúdo selecionado no módulo de atualização

Depois de selecionados os registro na tabela **Atualizacao**, o sistema monta uma nova consulta ao banco do servidor. Essa consulta tem a finalidade de verificar se o registro em que a *Condicao* (atributo da tabela **Atualizacao**) apontar existe. Essa consulta é feita através do seguinte código em SQL (*structured query language - linguagem de consulta estruturada*):

SELECT * FROM ' + ('NOMETABELA') + ('CONDICAO')

onde *NOMETABELA* e *CONDICAO* são os atributos da tabela **Atualizacao** buscados na seleção anterior no servidor.

Após ter verificado no banco do servidor se existe o registro, o sistema faz uma outra verificação. Essa nova busca só acontece se o registro apontado na tabela **Atualizacao** do servidor no atributo *TipoAtualizacao* for igual a “I”, buscando no banco do terminal se há aquele registro. Se já existir, o sistema substitui o valor do atributo *TipoAtualizacao* para “U”, assim não acontece de inserir o mesmo item, e dessa forma apenas atualizando o registro encontrado.

Após essa verificação o sistema monta os comandos para fazer a atualização das tabelas de acordo com o registro que estiver gravado no atributo *TipoAtualizacao*. Esses comandos em SQL são montados da seguinte maneira:

- Se o registro for “I”, inserir – *Insert into ('NomeTabela') (todos os atributos da tabela separados por vírgula) values ('todos os registros que tiver sendo apontado na tabela de mesmo nome do servidor, também separados por vírgula).* Exemplo: *insert into Cidade (CodCidade, DescricaoTerminalCidade, Nome) values (1, AA, “Uberlândia”)*
- Se o registro for “U”, alterar – *Update ('NomeTabela') set (todos os atributos da tabela com valores iguais aos da tabela do servidor) ('Condicao').* Exemplo: *update Cidade set CodCidade = 2, DescricaoTerminalCidade = ‘BB’, Nome = “Perdizes” where CodCidade = 1 and DescricaoTerminalCidade = ‘AA’.*
- Se o registro for “D”, excluir – *Delete from ('NomeTabela') ('Condicao').* Exemplo: *delete from cidade where CodCidade = 2 and DescricaoTerminalCidade = ‘BB’.*

O sistema repete esse procedimento com todos os registros que estiverem na tabela **Atualizacao** do banco do servidor.

4.4.2 ATUALIZANDO O BANCO DE DADOS DO SERVIDOR

Da mesma forma em que o sistema atualiza o banco de dados dos terminais, ele atualiza o servidor. Todos os procedimento que foram utilizados para ajustar o banco do terminal são usados com o servidor. A única diferença é que o sistema não busca a partir do ultimo código gravado. Ele atualiza todos os registros encontrados na tabela **Atualizacao** do banco do terminal e , após atualizar o sistema, apaga todos os registro encontrados na tabela **Atualizacao** do terminal. Essa exclusão é necessária para que se ganhe tempo, já que partindo do terminal somente o servidor irá ser atualizado, o contrário que ocorre com o servidor que recebe atualização de vários terminais.

4.4.3 ANTES E DEPOIS DA REPLICAÇÃO

É demonstrado nas figuras de 4.12 até 4.23, como os bancos de dados do terminal e do servidor estavam antes da replicação e depois da replicação. Esta demonstração considera apenas algumas tabelas.

Tabela Cidade:

CODCIDADE	DESCRICAO	TERMINALCIDADE	NOME
1	AA		Perdizes
2	AA		Patrocinio

Figura 4.12 – (Banco do terminal) antes da atualização

CODCIDADE	DESCRICAO	TERMINALCIDADE	NOME
1	SS		São Paulo
2	SS		Uberlândia

Figura 4.13 – (Banco do servidor) antes da atualização

CODCIDADE	DESCRICAO	TERMINALCIDADE	NOME
1	AA		Perdizes
1	SS		São Paulo
2	AA		Patrocinio
2	SS		Uberlândia

Figura 4.14 – (Banco do terminal e servidor) depois da atualização

Tabela Cliente:

CODCLIENTE	DESCRICAO	TERMINALCLI	CODFICHA	DESCRICAO	TERMINALFIC	CODCIDADE	DESCRICAO	TERMINALCID	NOMEFANTASIA
1	AA		1	AA		1	AA		Samuel Lázaro de Oliveira
2	AA		2	AA		2	AA		Joao Sistemas

Figura 4.15 – (Banco do terminal) antes da atualização

CODCLIENTE	DESCRICAO	TERMINALCLI	CODFICHA	DESCRICAO	TERMINALFIC	CODCIDADE	DESCRICAO	TERMINALCID	NOMEFANTASIA
1	SS		1	SS		1	SS		PC Informatica Ltda
2	SS		2	SS		2	SS		War sistemas e informatica

Figura 4.16 – (Banco do servidor) antes da atualização

CODCLIENTE	DESCRICAO	TERMINALCLI	CODFICHA	DESCRICAO	TERMINALFIC	CODCIDADE	DESCRICAO	TERMINALCID	NOMEFANTASIA
1	AA		1	AA		1	AA		Samuel Lázaro de Oliveira I
1	SS		1	SS		1	SS		PC Informatica Ltda
2	AA		2	AA		2	AA		Joao Sistemas
2	SS		2	SS		2	SS		War sistemas e informatica

Figura 4.17 – (Banco do terminal e servidor) depois da atualização

Estas duas tabelas, **Cidade** e **Cliente**, podem ser alteradas tanto no terminal como no servidor, já as tabelas **Produto** e **Fornecedor** só podem ser alteradas no servidor.

Tabela Produto:

CODPRODUTO	PREÇO	UNIDADE	CODREPRESENTADA	REFERENCIA	DESCRIÇÃO
<null>	<null>	<null>	<null>	<null>	<null>

Figura 4.18 – (Banco do terminal) antes da atualização

CODPRODUTO	PREÇO	UNIDADE	CODREPRESENTADA	REFERENCIA	DESCRIÇÃO
1	3000	PC	1	10001	Pentium 4 2.0 ghz
1	4200	PC	2	20001	NoteBook Pentium 4 1.8 ghz
2	1552	PC	1	10002	Pentium 4 1.2 ghz
2	3100	PC	2	20002	PC Pentium 4 2.4 ghz

Figura 4.19 – (Banco do servidor) antes da atualização

CODPRODUTO	PREÇO	UNIDADE	CODREPRESENTADA	REFERENCIA	DESCRIÇÃO
1	3000	PC	1	10001	Pentium 4 2.0 ghz
1	4200	PC	2	20001	NoteBook Pentium 4 1.8 ghz
2	1552	PC	1	10002	Pentium 4 1.2 ghz
2	3100	PC	2	20002	PC Pentium 4 2.4 ghz

Figura 4.20 – (Banco do terminal e servidor) depois da atualização

Tabela Fornecedor:

CODFORNECEDOR	NOME	ENDERECO	BAIRRO	CIDADE
<null>	<null>	<null>	<null>	<null>

Figura 4.21 – (Banco do terminal) antes da atualização

CODFORNECEDOR	NOME	ENDERECO	BAIRRO	CIDADE
1	José da Silva	Rua rua sei la	sei la	Uberlandia
2	Microtec	nao sei	tb nao sei	São Paulo

Figura 4.22 – (Banco do servidor) antes da atualização

CODFORNECEDOR	NOME	ENDERECO	BAIRRO	CIDADE
1	José da Silva	Rua rua sei la	sei la	Uberlandia
2	Microtec	nao sei	tb nao sei	São Paulo

Figura 4.23 – (Banco do terminal e servidor) depois da atualização

4.5 CONCLUSÃO

O projeto descrito neste capítulo possui um método eficiente e de uma complexidade pequena para implementação do controle dos dados em uma replicação de bases de dados distribuídas, pois ao contrário de se usar um método onde o sistema sai buscando todas as tabelas dos bancos, item por item, para encontrar qual item teria de ser alterado, inserido ou apagado, levariam muito mais tempo do que o método utilizado nesse projeto. Com este método o sistema busca somente os dados a serem trabalhados, ou seja, o sistema busca somente pelos itens que estiver no atributo Condicao da tabela **Atualizacao** e na tabela em que o atributo NomeTabela apontar, fazendo com que diminua o tempo gasto. Gastando menos tempo o sistema tem um melhor desempenho.

5. CONCLUSÃO

Os sistemas de bancos de dados sempre foram importantes na história da computação. Durante um bom tempo estes sistemas funcionaram somente de forma centralizada, onde os integrantes do sistema (dispositivos para armazenamentos, software de gerenciamento de dados e os próprios dados) ficam em um mesmo local (um computador).

Com o avanço da tecnologia na área da informática, surge então o processamento em rede, possibilitando que uma nova tecnologia na área de sistemas de banco de dados fosse criada, o sistema de bancos de dados distribuídos (SBDD). Esta tecnologia se diferencia da centralizada no aspecto de processamento das informações, pois é feita entre vários bancos de dados localizados em locais diferentes, ligados por uma rede de computadores.

O SBDD transforma a distribuição dos dados de forma clara para quem opera o sistema, dando a sensação de se estar trabalhando em um ambiente centralizado.

Existem vários métodos e sistemas de gerenciamento de bancos de dados (SGBD) para controlar um SBDD. Logicamente existem métodos melhores do que outros. No estudo de caso desenvolvido neste trabalho,

a utilização do método escolhido tem como objetivo mostrar a existência de métodos mais eficientes.

O método escolhido faz todo o processo de sincronia entre os bancos de dados de forma rápida. Este método utiliza de atributos e uma tabela de controle para melhor fazer a sincronia. Ao invés sair buscando item por item em todas as tabelas para saber qual item tem de ser atualizado, inserido ou excluído, o sistema utiliza dos registros que estão armazenados nessa tabela e somente faz a sincronia destes registros. Por exemplo, um banco que contém doze tabelas. Imagina-se que em cada tabela tivesse dez mil registros e deseja-se atualizar somente cem registros. Para o sistema fazer uma sincronia, ele vai ler cento e vinte mil registros para atualizar apenas cem. Com o método desenvolvido no projeto, o sistema irá ler apenas os cem que se deseja atualizar, tornando o sistema mais eficiente.

Este método foi implementado em um sistema, foi testado e comprovado que sua eficiência na sincronia dos dados é rápida e segura.

Com este trabalho foi demonstrado os problemas que podem ocorrer para se implantar um sistema de bancos de dados distribuídos. As vantagens e as desvantagens da distribuição de dados. A criação de um método (técnica) para sincronizar bancos de dados. E a aplicação do método de sincronização utilizando bancos de dados distribuídos de forma real.

Com um bom método de sincronização, é capaz de tornar uma empresa mais informatizada, com recursos mais baratos. Sendo assim, as empresas poderão expandir seus negócios em diferentes localidades mantendo suas bases de dados sempre em sincronia.

Como trabalhos futuros, o estudo das teorias, ferramentas e métodos de conversão de bancos de dados e SGBDs para implantação de sistemas de bancos de dados distribuídos. Criação de métodos e ferramentas para o processo de conversão dos mesmos.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] ÖZSU, M. Tamer; VALDURIEZ, Patrick. Principles of Distributed Database Systems. tradução [da 2^a ed. americana] Vandenberg D. de Souza. Rio de Janeiro : Campus, 2001.

- [2] KORTH, Henry F., Sistema de Banco de Dados, tradução [da 2^a ed. ver] Maurício Heihachiro Galvan Abe. São Paulo : Makron Books, 1995.

- [3] CASANOVA, Marco Antônio ; MOURA, Arnaldo Vieira, Princípios de Sistemas de Gerência de Banco de Dados Distribuídos, Rio de Janeiro : Campus, 1985.

- [4] SALEMI, Joe, Banco de Dados Cliente/Servidor, tradução [2^a ed. original] Cláudio Costa, Rio de Janeiro : Infobook, 1995.

- [5] GIMENEZ, Luís Júnior. “Informática em Concursos Públicos”, www.terravista.pt/ilhadomel/2388/glosbdd.html#inicio, janeiro, 2000. Acesso em 03/09/2002.

- [6] FONSECA, Décio. “Aulas do Professor Décio - UFPE”,

www.cin.ufpe.br/~bd/aulasDecio/aulas.html, 1998. Acesso em 03/09/2002.

[7] SOUZA, Luís Fernando Cardeal de. “III Wola – Workshop Interno do LaSiD”, www.lasid.ufba.br/eventos/wola99/resumos/fernando.html. S.D. Acesso em 03/09/2002.

[8] DATE, C. J.. “Introdução a sistemas de banco de dados / C. J. Date”, tradução (da 4ª edição original) de Contexto traduções, Rios de Janeiro. Campus, 1991.