UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUIZ OTÁVIO GIORDANI

ESTUDO SOBRE TRANSPARÊNCIA DE LOCALIZAÇÃO, NOMEAÇÃO E REPLICAÇÃO EM SISTEMAS DE GERÊNCIA DE BANCO DE DADOS DISTRIBUÍDOS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Professor Murilo S. De Camargo, Dr.

Florianópolis, Novembro de 2002.

ESTUDO SOBRE TRANSPARÊNCIA DE LOCALIZAÇÃO, NOMEAÇÃO E REPLICAÇÃO EM SISTEMAS DE GERÊNCIA DE BANCO DE DADOS DISTRIBUÍDOS

LUIZ OTÁVIO GIORDANI

Esta dissertação foi julgada adequada para obtenção do título de "Mestre em Computação", Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Fernado A. Ostuni Gauthier, Dr.

Coordenador do Curso

Banca Examinadora:

_____

Prof. Murilo S. de Camargo , Dr.

Orientador

_____

Prof. Rosvelter J. Coelho da Costa, Dr.

_____

Prof. Roberto Willrich, Dr.

.

DEDICATÓRIA

Dedico este trabalho a meus pais Otávio e Enercy,
a minha espôsa Maria Angela e a meus filhos
Luiz Otávio, Laura e Luiz Guilherme.

AGRADECIMENTO

Agradeço ao Prof. Dr. Murilo S. Camargo e
a equipe de professores do Departamento de
Informática e Estatística da UFSC.

SUMÁRIO

LISTA DE ABREVIATURAS

ASE	Aplication Server Enterprise
BD	Banco de Dados
CODASYSL	Conference on Data Systems Languages
CPU	Central Processor Unit
DBA	Database Administrator
DBS	Database System
DBMS	Database Management System
DDBMS	Distributed Database Management System
EDI	Eletronic Data Interchange
GCS	Global Conceptual Schema
Gb	Gigabyte
IBM	International Business Machine
Inc.	Incorporated
I/O	Input/Output
FDBS	Federated Database System
LCS	Local Conceptual Schema
Mb	Megabyte
MDBS	Multidabatase System
Mhz	Megahertz
ODBC	Open Database Connect
QUEL	Query Language
QEP	Query Execute Plan
RAM	Random Access Memory
SGBD	Sistema de Gerência de Banco de Dados
SGBDD	Sistema de Gerência de Banco de Dados Distribuído
SQL	Structured Query Language
SQL-92	Structured Query Language - 92
SQL3	Structured Query Language - 3
OLAP	On Line Aplication Program

LISTA DE SÍMBOLOS

<	Menor que
>	Maior que
>=	Maior ou igual
<=	Menor ou igual
π	Projeção
σ	Seleção
$\wedge$	E
$\bowtie$	Junção
$\times$	Produto Cartesiano
$\equiv$	Equivalente
$\cup$	União
$\cap$	Intersecção
$\leftarrow$	Atribuição
θ	Predicado de seleção

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Os SGBDD - Sistemas de Gerência de Banco de Dados Distribuídos resultam da união de duas tecnologias: A tecnologia dos SGBDs - Sistemas de Gerência de Banco de Dados; e da tecnologia das Redes de Computadores. Um SGBD tem o objetivo de gerenciar o armazenamento e a recuperação dos dados, escondendo do usuário a forma com que os dados são armazenados e recuperados. As Redes de Computadores interligam computadores e periféricos, utilizando protocolos de comunicação.

A arquitetura de informática composta por um ou mais servidores e por diversas estações de trabalho, interligadas por uma rede de computadores - local ou de longa distância, onde haja distribuição de dados, funções ou de contrôle entre os componentes, constitui um *sistema distribuído*. Esta arquitetura permite uma otimização do ambiente de informática, explorando a capacidade de comunicação e de processamento dos equipamentos e implementando paralelismo na execução de transações.

Um ambiente *de banco de dados distribuído* é composto por um sistema distribuído, onde cada local tem seu software gerênciador de banco de dados que é responsável por uma ou mais base de dados locais. Os Sistemas de Gerência de Banco de Dados Distribuídos são aplicativos de software cujo objetivo é gerenciar estes ambientes, de forma que os diversos bancos de dados locais sejam interligados formando um ambiente distribuído. As aplicações dos SBDD envolvem áreas de negócios que tenham como característica a distribuição geográfica de função e/ou de dados.

Este estudo investiga os SGBD Ingres versão 2.0, DB2 versão 7.1, Sybase ASE versão 11.9 e Oracle versão 8.1.7, analisando o modo como implementam um ambiente de banco de dados distribuído. Realiza dois estudos de caso, sendo que o primeiro estudo investiga a implementação das propriedades de transparência de localização e nomeação e verificando a performance na execução de consultas distribuídas. O segundo estudo estuda a propriedade de transparência de replicação, e configurando os SGBD DB2 e Oracle de modo a estudar como estes aplicativos implementam um ambiente replicado.

ABSTRACT

The DDBMS - Distributed Database Management Systems are result of the union of two technologies: The technology of the DBMS - Database Management Systems; e of the technology of the computer networks. A DBMS has the objective to manage the storage and the recovery of the data, hiding of the user the form with that the data are stored and recouped. The computer networks establish connection computers and peripherals, using communication protocols.

The architecture of the distributed database systems are composed for one or more servers and multiples stations of work, linked for a computer network - local or of long distance, where it has distribution of data, functions or of control between the components, it constitutes a distributed system. This architecture allows to a optimization of the computer environment, exploring the capacity of communication and processing of the equipment and implementing parallelism in the execution of transactions.

An environment of database distributed is composed for a distributed system, where each place has its management software of database that is responsible for one or more local database. The Distributed Database Management Systems are applicatory of software whose objective is to manage these environments, of form that the diverse local data bases are linked forming a distributed environment. The applications of the DDBS involve business-oriented areas that have as characteristic the geographic distribution of function and/or data.

This study it investigates the DBMS Ingres version 2.0, DB2 version 7.1, Sybase ASE version 11.9 and Oracle version 8.1.7, analyzing the way as they implement an environment of data base distributed. It carries through two studies of case, being that the first study it investigates the implementation of the properties of localization transparency and nomination and verifying the performance in the execution of distributed consultations. As the study it studies the property of response transparency, and configuring the DBMS DB2 and Oracle in order to study as these applicatory ones implement an talked back environment.

1 INTRODUÇÃO

Um banco de dados distribuído (BDD) é uma coleção de múltiplos, logicamente inter-relacionados bancos de dados locais, distribuídos sobre uma rede de computadores. Tem como suposições implícitas:

- Os dados são armazenados em um certo número de sites;
- Cada site consiste em uma unidade computacional autônoma;
- Os dados são logicamente relacionados, não sendo uma coleção de arquivos distribuídos.

As aplicações dos sistemas de banco de dados distribuídos (SBDD) envolvem a utilização em áreas de negócios que tenham como característica a distribuição geográfica de função e de dados. Entre elas, citamos:

- Manufatura, principalmente quando envolve diversas plantas de produção;
- Comando e controles militares;
- Instituições financeiras;
- Companhias aéreas;
- Cadeias de hotéis;
- Empresas comerciais e industriais com matriz e filiais;
- Organizações que possuam estrutura organizacional descentralizada.

Um sistema de gerência de banco de dados distribuído (SGBDD) é o sistema que permite a gerência dos bancos de dados distribuídos e que torna a distribuição transparente ao usuário. O termo sistema de banco de dados distribuído (SBDD) é usado para referenciar a combinação de BDD e SGBDD (ÖZSU, 1999).

Uma possível arquitetura de um banco de dados distribuído é composta por um conjunto de sites de consulta (possivelmente completo) e por um conjunto incompleto de sites de dados. Os sites de dados tem a capacidade de armazenar os dados, enquanto que os sites de consulta não. Os sites de consulta somente executam as aplicações de interface com o usuário, de modo a facilitar o acesso aos dados (ÖZSU, 2000). A questão do conjunto de sites de consulta ser possivelmente completo refere-se ao fato de que todos os sites podem realizar consultas, sendo que não necessariamente o façam. A

questão relativa a um conjunto incompleto de sites de dados se dá pelo fato de que os dados são armazenados em computadores servidores e as consultas realizadas em computadores clientes (fig.1.1).

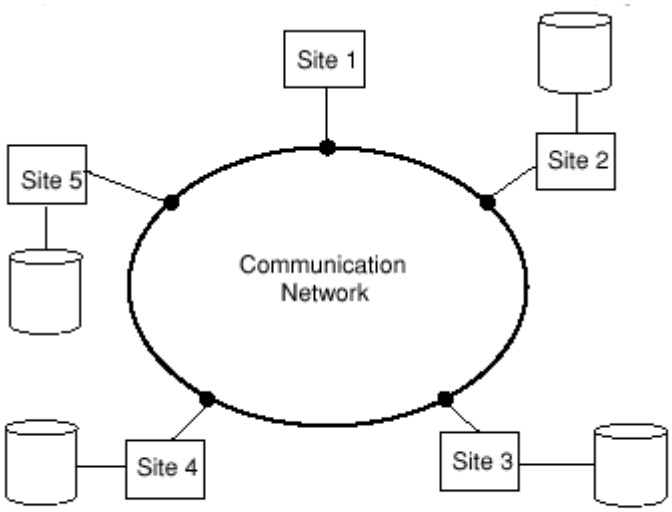


Figura 1.1. Ambiente de Banco de Dados Distribuído
Fonte: (OZSU 2000)

Na figura 1.1, o site 1 é um site de consulta e os sites 2, 3, 4 e 5 são sites de dados e, possivelmente, sites de consulta.

Caso os diversos sistemas de banco de dados de cada site sejam do mesmo tipo, fornecedor e versão, o sistema é dito como *homogêneo*. Se os bancos de dados possuem alguma forma de heterogeneidade, diferença de fornecedor ou de versão, são referenciados como *sistemas multidabase (multidatabase system)* ou *sistema de banco de dados federado (federated database system).* Se a funcionalidade do SGBD e dos dados é executada por um computador multiprocessado, ele é referenciado como um *sistema de banco de dados paralelo (parallel database system).*

Existem diferentes modelos de arquitetura de distribuição para o desenvolvimento de SGBD distribuídos, variando de sistemas cliente/servidor até sistemas ponto-a-ponto (peer-to-peer). Na arquitetura cliente/servidor, o site de consulta corresponde ao cliente enquanto que o site de dados corresponde ao servidor. Na arquitetura peer-to-peer, não há distinção entre clientes e servidores. As arquiteturas diferem com respeito a localização onde cada função do SGBD é provida (ÖZSU, 2000). Nos sistemas cliente/servidor, o servidor é responsável pela gerência do armazenamento dos dados, pelas transações e pelo processamento e otimização das

consultas. O cliente executa a aplicação do usuário e a interface com o banco de dados, além disto, possui um módulo do SGBD chamado de *cliente (client)* que é responsável pela gerência dos dados e das transações que estão na memória cache do cliente. A arquitetura cliente/servidor mais simples é o sistema *multiple-client/single-server*, onde o banco de dados é armazenado em um único servidor e acessado por diversos clientes. Uma arquitetura mais complexa é o sistema *multiple-client/multiple-server*, onde os dados estão localizados em mais de um servidor.

Nos sistemas peer-to-peer não há distinção entre clientes e servidores, sendo que cada site tem a mesma funcionalidade. Isto é provido separando os módulos que atendem as requisições dos usuários dos módulos que gerenciam o armazenamento dos dados.

Os sistemas de bancos de dados distribuídos comerciais são desenvolvidos por empresas de software e concorrem no mercado divulgando suas características e evidenciando suas qualidades com relação aos concorrentes. Estes sistemas buscam implementar as características e funcionalidades descritas em trabalhos acadêmicos e em pesquisas desenvolvidas por institutos de pesquisa e universidades. Os trabalhos de pesquisa descrevem as características funcionais, caracterizando e definindo os requisitos operacionais de cada componente de um banco de dados distribuído. Sob o ponto de vista mercadológico, os desenvolvedores implementam estas funcionalidades de modo particular e buscam preservar as informações referentes ao modo como foi realizada a implementação, tendo em vista obter diferenciais competitivos com relação a produtos concorrentes. Assim, o modo como foram implementados os requisitos funcionais de um sistema constitui-se em segredo de indústria, onde, o acesso por parte de usuários e pesquisadores é vetado. Analisando as necessidades dos usuários destes sistemas, os critérios de avaliação e de seleção de um ou outro software se dão por meio de reportagens, avaliações realizadas por empresas e institutos de pesquisa e instituições de ensino, sendo que, quanto maior a quantidade e qualidade da informação referente a estes produtos, melhor são os critérios sobre os quais será baseada uma decisão. Ao usuário cabe buscar a melhor alternativa que atenda suas necessidades e, de modo mais específico, que aproxime-se mais das características demonstradas neste estudo. A

analise dos SGBD comerciais permite observar como estes sistemas implementam estas especificações e permite realizar análises sobre as implementações.

O objetivo desta dissertação é realizar uma análise sobre os sistemas de banco de dados Ingres versão 2.0, DB2 versão 7.1, Sybase ASE versão 11.9 e Oracle versão 8.1.7, implementando um ambiente de banco de dados distribuído, configurando os sistemas e analisar o modo como são atendidos os critérios de transparência de localização e transparência de nomeação das bases de dados. Para analisar a transparência de localização será realizada uma análise dos SGBD e desenvolvido um aplicativo para inserir e realizar consultas sobre bases de dados distribuídas, verificando o desempenho de cada sistema de gerência de banco de dados. Também será desenvolvida uma análise verificando a implementação da replicação de tabelas sobre as bases de dados distribuídas. Como objetivos específicos, propõem-se a:

- Instalar e configurar os SGBD sob o sistema operacional Red Hat Linux 6.2;
- Configurar um ambiente de banco de dados distribuídos, verificando as propriedades de transparência de localização e de nomeação das bases de dados;
- Desenvolver um aplicativo de software que realize consultas sobre as bases de dados distribuídas e analisar os resultados com relação a:
    1. Configuração do ambiente de distribuição;
    2. Implementação de transparência de localização e de nomeação;
    3. O desempenho dos bancos de dados na realização das consultas.
- Analisar os sistemas de banco de dados em um ambiente de replicação de dados, verificando:
    1. Configuração dos SGBDs para o ambiente de replicação;
    2. Implementar um ambiente replicado e verificar a replicação dos dados entre os sites.

1.1 JUSTIFICATIVA

Este estudo tem o objetivo de analisar quatro sistemas de banco de dados distribuídos comerciais, utilizando como critério para a seleção os fatores:

1. O SGBD Ingres pelo fato de ser oriundo de um projeto universitário, desenvolvido na Universidade da Califórnia, nos fins dos anos 70, tornando-se uma referência nos estudos dos sistemas de banco de dados.
2. Os três SGBD com maior participação no mercado de banco de dados e com suporte para o ambiente Linux.

A tabela 1.1 demonstra a participação do mercado dos bancos de dados. Baseado nestes critérios, foram selecionados os SGBD: Oracle, IBM DB2 e Sybase.

Tabela 1.1. Quotas de Mercado dos DBMS

Empresa	Quota de mercado 2000	Quota de mercado 1999
Oracle	33,8%	31,4%
IBM	30,1%	29,9%
Microsoft	14,9%	13,1%
Sybase	3,2%	3,3%
Informix	3%	5%
Outros	15%	17,3%
Totalidade do mercado	100%	100%

Valores em percentagem Nota: exclui transacções por EDI Fonte: BGC

Fonte: (COMPUTERWORD 2001)

O desenvolvimento de aplicativos para sistemas de banco de dados distribuídos (SBDD) envolve o acesso a bases de dados distribuídas interligadas por uma rede de comunicação. O usuário realiza consultas sobre bases de dados locais, sobre dados remotos, bem como sobre o conjunto de todos os dados. Como exemplo, pode-se citar o de uma empresa varejista comercial, que possui estabelecimentos em diversas cidades, e a operação de compra de um produto qualquer. O cliente dirige-se a um dos estabelecimentos comerciais e, em contato com o vendedor informa o desejo de realizar a compra. O vendedor, para isto, consulta a base de dados do estabelecimento para conhecer o preço, condições de venda e disponibilidade de estoque da mercadoria, realiza, portanto, uma consulta local. No caso da mercadoria não estar disponível neste estabelecimento, o vendedor passa a ter a necessidade de buscar a mercadoria em

outros estabelecimentos - realizando uma busca remota aos dados. Da mesma forma, um analista de negócios ou um administrador da empresa busca saber o montante de vendas da empresa ou outros dados estatísticos. Neste caso, necessita realizar uma busca sobre todas as bases de dados, realiza uma consulta sobre todas as bases de dados. Este exemplo demonstra a complexidade inerente ao negócio, e o desenvolvedor da aplicação deve implementar no aplicativo de software estas funcionalidades.

Sendo o desenvolvedor um usuário do banco de dados, a consulta será expressa em linguagem SQL, através de uma instrução SELECT. A questão é:

1. Como são implementadas as consultas relacionais as bases de dados locais, remotas e sobre todas as bases?
2. Como é implementada a distribuição dos dados e de que modo isto se torna transparente ao usuário?
3. Como são nomeadas as tabelas locais e remotas e como é implementada a transparência de nomeação?
4. Existem diferenças, a nível de aplicativo de usuário, no uso de um ou outro sistema de banco de dados - uma aplicação escrita para um determinado sistema pode ser reaproveitada para outro?
5. Como é implementada a replicação dos objetos do banco de dados, especificamente no que ser refere aos SGBD citados acima?

Analisar, portanto, como os SBDD resolvem estas questões para o usuário do sistema. A resposta se dá através da implementação de *transparência de distribuição e transparência de nomeação* e *replicação dos dados*. Estas características permitem ao usuário desconhecer os detalhes da implementação da distribuição e nomeação e existência de cópias dos dados.

Este estudo analisa estas questões sob o ponto de vista do desenvolvedor (usuário) do banco de dados, contribuindo no conhecimento destes sistemas, dos bancos de dados distribuídos e auxilia a tomada de decisão na definição por um determinado SGBD. As analises realizadas refletem a realidade do ambiente de negócios das empresas, servindo como uma referência para estudo e análise dos SGBDD e deste tipo de ambiente de informática.

1.2 MOTIVAÇÃO

O processamento distribuído corresponde melhor à estrutura organizacional encontrada nas empresas. Muitas das aplicações da tecnologia da computação é distribuída por natureza. O comércio eletrônico sobre a Internet, aplicações multimídia, notícias por demanda, imagens médicas e sistemas de controle de manufaturas são exemplos destas aplicações. Sob um aspecto mais global, o processamento distribuído está melhor apto a resolver os grandes e complicados problemas encontrados hoje, através da variação da regra “dividir e conquistar” (ÖZSU, 1999).

Esta realidade pode ser mensurada através da análise dos requisitos de negócios das empresas. Um sistema que implemente uma estrutura de software que corresponda à estrutura organizacional da empresa fornece recursos para uma melhor administração do ambiente de informática e aproxima os recursos de informática dos usuários. A distribuição de servidores entre prédios e/ou departamentos é uma forma de adequar a estrutura de informática à estrutura organizacional. Aliando a esta distribuição de hardware a distribuição física e lógica dos dados, obtém-se um ambiente computacional idêntico à estrutura funcional do negócio, fornecendo um ambiente onde os recursos computacionais necessários à administração do negócio correspondem à estrutura organizacional da empresa. Somando-se a estes fatores a oferta de sistemas de software, sistemas operacionais e aplicativos, de uso livre, tem-se uma combinação que torna muito atraente o uso deste ambiente computacional. O sistema operacional Linux, em todas as suas variantes, é distribuído na forma de software livre, e constitui uma alternativa para diminuir os custos de informática. Este fato faz com que uma série de aplicativos de software tenham o seu custo de aquisição diminuído, ou ofertados também de modo gratuito. Sendo este um motivo que contribui para a adoção de um ambiente de banco de dados distribuído.

Necessário salientar que, sob os aspectos demonstrados acima, o conhecimento e a aplicação dos sistemas de banco de dados distribuídos é um requisito para os profissionais da informática.

1.3 Limitações da Pesquisa

Esta pesquisa implementa um ambiente de banco de dados distribuído homogêneo, com distribuição de dados, funções e controle, visto que os bancos de dados são do mesmo fabricante, mesma versão, e localmente autônomos. A estação cliente, onde é executado o aplicativo que implementa as consultas e manutenções sobre a base de dados, utiliza aplicativos cliente e drivers de ODBC fornecidos pelo desenvolvedor do sistema. O ambiente de avaliação é *peer-to-peer* na distribuição e *single-client/multiple-server* com relação ao cliente.

1.4 Estrutura da Dissertação

O capítulo 1 apresenta os objetivos e a motivação para a realização deste estudo.

O capítulo 2 realiza uma revisão da literatura dos sistemas de banco de dados distribuídos, classificando-os e descrevendo suas características e especificações funcionais.

O capítulo 3 desenvolve a revisão bibliográfica com relação ao processamento e otimização de consultas nos bancos de dados distribuídos, demonstrando a estrutura funcional e as questões inerentes a serem implementadas nos sistemas de gerência de banco de dados distribuídos.

O capítulo 4 investiga os sistemas de banco de dados Ingres versão 2.0, DB2 versão 7.1, Sybase ASE versão 11.9 e Oracle versão 8.1.7, descrevendo as características e os aspectos operacionais. Demonstra a instalação e configuração destes sistemas sob o sistema operacional Red Hat Linux 6.2.

O capítulo 5 implementa as análises dos SGBD. A primeira análise tem como objetivo avaliar a distribuição de dados entre três bases de dados distribuídas, demonstrando o modo como foi implementada a distribuição e analisa os resultados obtidos. A segunda análise estuda a implementação da replicação de dados entre duas bases de dados nos sistemas de gerência de banco de dados DB2 e Oracle.

2 Tecnologia dos Bancos de Dados Distribuídos

Introdução

Neste capítulo é feita a revisão bibliográfica da tecnologia dos sistemas de banco de dados distribuídos, estudando seus requisitos e modo de implementação.
Os tópicos pesquisados referem-se a análise das vantagens e desvantagens na utilização, conceito dos sistemas de banco de dados distribuídos, suas propriedades, classificação e implementações. São apresentados o conceito, as vantagens, as funções que os BDD devem prover e mostradas as arquiteturas de implementação, as propriedades, a classificação e os tipos de distribuição, fragmentação e replicação dos dados. Caracteriza os tipos de sistemas de banco de dados distribuídos, em específico os BDD client/server, distributed database e multidatabase e apresenta a sua taxonomia.

Tem o objetivo de fornecer o embasamento teórico de modo a conceituar e classificar o objeto de estudo desta dissertação. O desenvolvimento de aplicativos para a gerência de banco de dados distribuídos é fundamentado nas teorias dos sistemas de banco de dados e nos estudos realizados sobre suas implementações.

2.1 Conceito

A tecnologia dos bancos de dados distribuídos (BDD) é a união de duas tecnologias distintas no processamento de dados: a tecnologia dos sistemas de banco de dados e a tecnologia das redes de computadores.

Os BDD tem sido propostos como alternativa ao armazenamento de dados centralizados, tendo como principais vantagens (ELMASRI & NAVATHE, 2000):

- **Gerência dos dados distribuídos com diferentes níveis de transparência:** Um BDD deve ter *distribution transparent* no sentido de esconder os detalhes de onde o arquivo (tabela, relação) é fisicamente armazenado. Os seguintes tipos de transparência são possíveis (ELMASRI & NAVATHE, 2000):

- **Transparência de Rede ou Distribuição - distribution or network transparency***:* Refere-se a transparência do usuário com relação aos detalhes da distribuição e da rede. É dividida em dois modos: *location transparency* e *naming transparency*. A *Transparência de localização* - refere-se ao fato de que os comandos usados para realizar a tarefa independem da localização dos dados e do sistema onde o comando está sendo executado; *Transparência de nomeação* - implica na atribuição de um nome único a cada objeto no banco de dados, e, a partir disto, os objetos sejam acessados sem a necessidade de especificações adicionais.
- **Transparência de Replicação - replication transparency:** Implica na realização de cópias dos dados em diversos sites, de modo a obter maior disponibilidade, desempenho e segurança. A transparência de replicação faz com que o usuário desconheça a existência das cópias.
- **Transparência de Fragmentação - fragmentation transparency:** A fragmentação dos dados é a sua divisão com o objetivo de aumentar a disponibilidade, o desempenho e a segurança do sistema. São possíveis dois tipos de fragmentação: *Fragmentação Horizontal* - quando as relações de distribuição são feitas em conjuntos de tuplas. *Fragmentação Vertical* - quando a distribuição é feita em subconjuntos de colunas. Uma consulta global deve ser transformada em uma série de consultas locais. A transparência de fragmentação torna transparente ao usuário os detalhes da existência de fragmentos.

- **Incremento de Disponibilidade e Segurança:** Estas são as vantagens mais comuns dos bancos de dados distribuídos. A *segurança (realibity)* - é definida como a probabilidade do sistema estar em execução num momento de tempo; a *disponibilidade (avalibity)* - é probabilidade do sistema estar continuamente disponível durante um intervalo de tempo. Quando os dados são distribuídos sobre diversos sites, caso um destes falhe, os restantes continuam a operar. Somente os dados e o software existente no site da falha não podem ser acessados.

- **Incremento de Desempenho:** A fragmentação de um banco de dados faz com que os dados estejam localizados onde são mais necessários. A *localização dos dados (data location),* reduz a disputa por CPU e serviços de I/O e reduz os tempos de acesso envolvidos nas redes wan (wide area networks). Quando um grande banco de dados é distribuído sobre diversos sites, pequenas bases de dados passam a existir em cada site. Como resultado, consultas e transações locais têm melhor desempenho do que se as mesmas fossem submetidas a um banco centralizado. Além disto, consultas podem ser executadas em paralelo, através da sua execução em diversos sites ou pela subdivisão de uma consulta em diversas subconsultas.
- **Facilidade de Expansão:** Nos ambientes distribuídos, a expansão do sistema em termos de adição de dados, número de processadores é muito mais fácil que em ambientes centralizados, visto que implica na adição de sites e bases de dados.

A distribuição traz um incremento na complexidade do design do sistema e na sua implementação. De modo a explorar as vantagens da distribuição, um software de gerência de bancos de dados distribuídos deve prover as funções (ELMASRI & NAVATHE, 2000):

- **Armazenar a localização dos dados:** Sendo a habilidade em conhecer a distribuição dos dados, fragmentação e replicação, através da expansão do catálogo do banco de dados.
- **Processar consultas distribuídas:** A habilidade de acessar sites remotos e transmitir consultas e dados sobre diversos sites, através de uma rede de comunicação.
- **Gerenciar transações distribuídas:** Sendo a habilidade de desenvolver estratégias de execução para consultas e transações que acessam dados em mais de um site, em sincronizar o acesso aos dados distribuídos e em manter a integridade de todo o banco de dados.
- **Gerenciar a replicação dos dados:** Consistindo em estar apto a decidir qual copia dos dados replicados será acessada e em manter a consistência entre as cópias dos dados replicados.

- **Recuperação distribuídas do banco de dados:** Que é a habilidade de recuperação a partir de falhas em sites individuais ou a partir de novos tipos de falhas como as falhas de comunicação.
- **Segurança:** Onde transações distribuídas deve ser executadas com uma gerência de segurança sobre os dados e sobre privilégios de acesso/autorização dos usuários.
- **Gerência de diretório distribuída:** Um diretório contém informações sobre o banco de dados. Um diretório pode ser global para todo o banco de dados e local para cada site. A localização e distribuição dos diretórios são questões relativas ao design e política do sistema.

O termo distribuição pressupõe uma questão: O que pode ser distribuído? Segundo ÖZSU (1999) três elementos podem ser distribuídos:

1. **Distribuição de funções.** De modo que várias funções de um sistema computacional podem ser delegadas para vários componentes de hardware ou de software.
2. **Distribuição de dados.** Onde os dados utilizados pelas aplicações podem ser distribuídos entre sites.
3. **Distribuição de controle.** O controle da execução de várias tarefas pode ser distribuída ao invés de ser executada por um único computador.

A idéia básica dos bancos de dados paralelos é implementar passos em paralelo sempre que possível, de modo a implementar o desempenho (RAMAKRISHNAN & GEHRKE, 2000). Um *sistema multiprocessado* é um sistema com dois ou mais processadores compartilhando recursos de memória (ÖZSU, 1999). Um *sistema de computação distribuída* ou *distributed computing system* consiste em um número de elementos de processamento, não necessariamente homogêneos, interconectados por uma rede de computadores e que cooperam para a execução de certas tarefas designadas (RAMAKRISHNAN & GEHRKE, 2000) (ÖZSU, 1999). Um sistema de banco de dados paralelo procura implementar o desempenho através da execução de varias operações em paralelo, como leitura de dados, construção de índices e avaliação de consultas. Embora os dados possam ser armazenados de modo distribuído num sistema,

a distribuição é dirigida por considerações de desempenho (ELMASRI & NAVATHE, 2000).

Três arquiteturas básicas são propostas na construção de bancos de dados paralelos (RAMAKRISHNAN & GEHRKE, 2000).

1. **Sistemas shared-memory.** Consistem em múltiplas CPUs interconectadas por uma rede de comunicação e que compartilham uma memória principal.
2. **Sistemas shared-disk.** Cada CPU possui sua própria memória principal e acessam as memórias secundárias através de uma rede de comunicação.
3. **Sistemas shared-nothing.** Nesta implementação, cada CPU tem memórias principal e secundária própria, onde duas CPUs não podem acessar a mesma área de armazenamento, e a comunicação entre as CPUs é feita através da rede de comunicação.

Quando os sistemas compartilham a memória principal e também memória secundária, são chamados de *sistemas shared memory* ou *tightly coupled* (ELMASRI & NAVATHE, 2000). Quando estes sistemas compartilham a memória secundária, mas cada um possui sua própria memória principal, são chamados de *shared disk* ou *loosely coupled* (ELMASRI & NAVATHE, 2000). O termo *coupled* refere-se ao grau de acoplamento entre os componentes do sistema. Se a comunicação é feita sobre uma rede de computadores, existe um *weak coupling* (acoplamento fraco) entre os elementos de processamento. Se os componentes forem compartilhados, ocorre um *strong coupling* (acoplamento forte) dos elementos.

O termo *shared-everything* é utilizado para referenciar o modelo que permite cada processador acessar qualquer recurso (memória principal e secundária, e periféricos) (ÖZSU, 1999). Desta forma, as arquiteturas *shared-disk* e *shared-memory* são arquiteturas shared-everything. Estas arquiteturas permitem os processadores comunicarem-se sem sobrecarga de mensagens sobre a rede (ELMASRI & NAVATHE, 2000).

Do ponto de vista dos sistemas de banco de dados distribuídos, todos estes modos de distribuição são necessários e importantes (ÖZSU, 1999). Os sistemas de gerência de banco de dados que utilizam estas arquiteturas são chamados de *parallel database management systems (sistemas de gerência de banco de dados paralelos*), uma

vez que utilizam tecnologia de processamento paralelo (ELMASRI & NAVATHE, 2000).

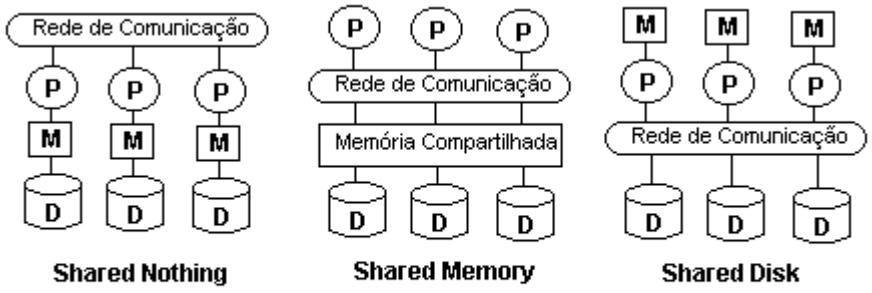


Figura 2.1. Arquiteturas Físicas para Sistemas de Banco de Dados Paralelos
Fonte: (RAMAKRISHNAN & GEHRKE 2000)

É necessário distinguir um *sistema de banco de dados paralelo* de um *sistema de banco de dados distribuído*. Um *parallel database system* procura prover desempenho através da implementação de operações em paralelo, como leitura de dados, construção de índices e analise de consultas (ELMASRI & NAVATHE, 2000). Um *distributed database system* armazena os dados em diversos sites, sendo cada site gerenciado por um DMBS capaz de operar de modo independente dos outros sites (ELMASRI & NAVATHE, 2000). Um banco de dados paralelo, construído sob uma arquitetura shared-nothing, aproxima-se do conceito de banco de dados distribuídos. Nos sistemas multiprocessados shared-nothing existe simetria e homogeneidade nos nós da rede, não sendo isto verdadeiro no ambiente dos bancos de dados distribuídos, onde a heterogeneidade de hardware e sistemas operacionais em cada nó é comum (ELMASRI & NAVATHE, 2000). Os bancos de dados paralelos são caracterizados pelo uso de máquinas multiprocessadas e múltiplos periféricos de armazenamento de dados, com varias funções de interconexão (ATZENI et al., 2000). Num banco de dados distribuído, cada servidor possui sua própria identidade e armazena dados que estão “funcionalmente” associados a este servidor (ATZENI et al., 2000).

2.2 Bancos de Dados Distribuídos

Os *sistemas de banco de dados distribuídos* tem como característica o armazenamento dos dados distribuídos nos sites, sendo que cada site é gerenciado por um SGBD independente dos demais (RAMAKRISHNAN & GEHRKE, 2000). Esta distribuição dos dados deve ser feita de modo transparente ao usuário. Um banco de dados distribuído é uma coleção de múltiplos, logicamente inter-relacionados bancos de dados, distribuídos sobre uma rede de computadores (ÖZSU, 1999). Um sistema de gerência de banco de dados distribuídos - SGBDD é um software (sistema) que permite gerenciar a distribuição dos bancos de dados, tornando a distribuição transparente ao usuário (ÖZSU, 1999). Os bancos de dados distribuídos trazem as vantagens da computação distribuída para o domínio da gerência dos bancos de dados (ELMASRI & NAVATHE, 2000). Uma das mais importantes tendências em banco de dados é o aumento no uso de técnicas de paralelismo e de distribuição de dados. Como propriedades principais, possuem (RAMAKRISHNAN & GEHRKE, 2000):

- **Independência na distribuição dos dados:** Os usuários devem realizar consultas sem a necessidade de referenciar onde as relações, ou cópias ou fragmentos de relações, estão armazenadas.
- **Atomicidade das transações distribuídas:** Os usuários devem escrever transações que acessem ou atualizem dados em diversos sites do mesmo modo que realizariam as transações sobre os dados locais. Assim, os efeitos da transação realizada entre os sites deve permanecer atômica.

A arquitetura dos bancos de dados distribuídos é classificada, segundo ÖZSU (1999), conforme três aspectos: a *autonomia* dos sistemas locais, a *distribuição* dos SGBD, e a *heterogeneidade* dos SGBD. Os sistemas que são constituídos por múltiplos sistemas de banco de dados, nos quais os FDBMS são um tipo específico, que podem ser caracterizados ao longo de três dimensões ortogonais: distribuição, heterogeneidade e autonomia (SHETH & LARSON 1990). Um federated database system é uma coleção de cooperativos mas autônomos sistemas de bancos de dados (SHETH & LARSON, 1990).

A *autonomia* refere-se à distribuição de controle, não dos dados. Indica o grau que cada sistema de gerência de banco de dados local opera de modo independente (ÖZSU, 1999). Um sistema de banco de dados que participa de um sistema de banco de dados federado pode manifestar diversos tipos de autonomia: *design, comunicação, execução e associação* (SHETH & LARSON, 1990). A *autonomia de design* refere-se à habilidade do sistema de banco de dados de escolher seu próprio design, com respeito a qualquer aspecto, incluindo (SHETH & LARSON, 1990):

- Gerência dos dados;
- Representação e nomeação dos dados;
- De conceituação ou interpretação semântica dos dados;
- Dos critérios de serialização e restrições de integridade;
- Da funcionalidade do sistema;
- De associação e compartilhamento com outros sistemas;
- De implementação.

A *autonomia de comunicação* refere-se à habilidade de decidir o meio de comunicar-se com outros sistemas. Um componente com autonomia de comunicação é apto a decidir quando e como responder a uma requisição de outro sistema de banco de dados (SHETH & LARSON, 1990). A *autonomia de execução* refere-se à habilidade de executar transações locais sem a interferência de operações externas e de decidir a ordem de execução das operações externas (SHETH & LARSON, 1990). A *autonomia de associação* implica na habilidade de decidir o meio e o quando compartilhar a funcionalidade e recursos. Incluindo a habilidade associar-se ou desassociar-se da federação e a habilidade de um DBS de participar de uma ou mais federações (SHETH & LARSON, 1990).

A *distribuição* refere-se à distribuição dos dados (ÖZSU, 1999). Os dados podem ser distribuídos entre diversos bancos de dados. Estes bancos de dados podem ser um único sistema de computação ou diversos sistemas de computação, co-localizados ou distribuídos geograficamente, mas interconectados por um sistema de comunicação (SHETH & LARSON, 1990). Os modos de distribuição podem ser agregados em duas classes: distribuição *client/server*, e distribuição *peer-to-peer*. Na distribuição client/server há a distinção entre equipamentos clientes e servidores,

ocorrendo uma distribuição de funções. Na distribuição *peer-to-peer* não há distinção de máquinas clientes e servidoras, cada equipamento tem um sistema de gerência de banco de dados completo e comunica-se com outros equipamentos para executar consultas e transações. A *heterogeneidade* ocorre de vários modos, partindo da heterogeneidade de hardware e nos protocolos de rede, indo até as variações nos gerenciadores dos dados (ÖZSU, 1999).

A heterogeneidade nos sistemas de bancos de dados podem ser divididas entre aquelas que diferenciam o sistema de gerência de banco de dados e aquelas que diferenciam a semântica dos dados (SHETH & LARSON, 1990). A heterogeneidade nos sistemas de bancos de dados ocorre como conseqüência da evolução tecnológica dos sistemas - desta forma, os sistemas mais recentes incorporam inovações tecnológicas que não estão presentes em versões anteriores. Como, também, pelo fato de que as organizações podem necessitar de sistemas diferentes para propósitos diferentes.

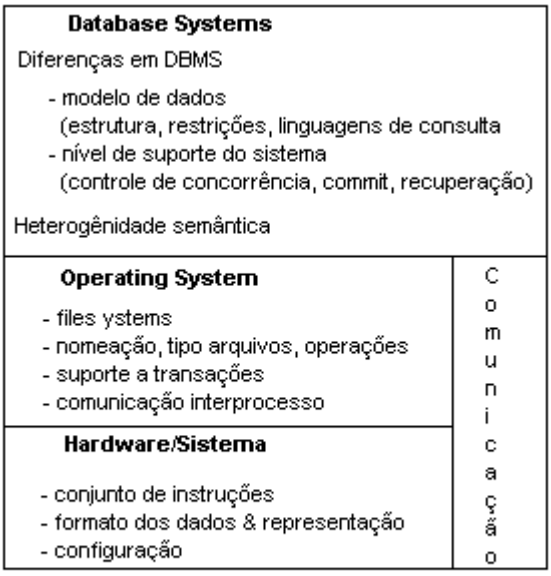


Figura 2.2. Tipos de Heterogênidade
Fonte: (SHETH & LARSON 1990)

Assim, um sistema é mais adequado para uma aplicação do que outro.

- **Heterogeneidade em SGBD:** Cada sistema de gerência de banco de dados tem um modelo de dados utilizado para definir a estrutura dos dados e as restrições. Esta representação e os aspectos da linguagem conduzem a heterogeneidade (SHETH & LARSON, 1990).

- **Diferenças na estrutura ou nos modelos de dados:** Modelos de dados diferentes provêem diferentes primitivas estruturais (SHETH & LARSON, 1990). Os bancos de dados possuem uma variedade de modelos de dados, como, por exemplo, o modelo relacional e o modelo orientado a objeto. Isto faz com que seja necessário um esquema de consultas globais, baseado em um mecanismo inteligente de processamento de consultas, responsável pela representação dos dados, nomes de atributos e relações conforme os diferentes tipos de bancos de dados.
- **Diferenças nas restrições ou regras de consistência:** Dois modelos de dados podem suportar diferentes restrições. Por exemplo, o conjunto de tipos de dados no esquema CODASYL pode ser parcialmente modelado com restrições de integridade no esquema relacional. CODASYL, entretanto, suporta inserções e restrições de contenção que não são detectadas somente pelas restrições de integridade referencial - ou de modo único (SHETH & LARSON, 1990). As regras de consistência são implementadas em cada banco e variam conforme cada sistema. O esquema de consultas globais do banco de dados distribuído deve lidar e resolver os conflitos entre as diferentes regras existentes, nos diversos bancos de dados que fazem parte do sistema de distribuição.
- **Diferenças na linguagens de consulta:** Linguagens diferentes são utilizadas para manipular dados representados em diferentes modelos de dados. Mesmo que dois DBMS suportem o mesmo modelo de dados, a diferença na linguagem de consulta (ex: SQL e QUEL), ou versões da linguagem SQL diferentes, suportadas por dois sistemas de gerência de banco de dados relacionais, contribuem para a heterogeneidade (SHETH & LARSON, 1990). Os diferentes sistemas têm seus próprios conjuntos de tipos de dados, operadores, funções, etc. Além disto, implementam variações na linguagem de consulta e manutenção dos dados, como SQL, SQL-92, SQL3, etc. O modelo distribuído deve administrar esta variedade de modelos e resolver os conflitos.

- **Heterogeneidade Semântica:** A heterogeneidade semântica ocorre quando existem diferenças no propósito, interpretação ou intenção de uso do mesmo dado ou de dados relacionados (SHETH & LARSON, 1990). A heterogeneidade semântica entre os componentes de um sistema de banco de dados (DBSs) é o maior obstáculo no projeto do esquema global dos bancos de dados heterogêneos (ELMASRI & NAVATHE, 2000). A *autonomia na construção* dos componentes dos bancos de dados distribuídos e a liberdade na escolha de parâmetros de construção afetam a complexidade dos FDBS. As questões ligadas a isto são (ELMASRI & NAVATHE, 2000):
  - *O universo do discurso de como cada dado é descrito:* considerando dois bancos de dados com mesmos nomes de atributos podem conter dados que representem informações distintas. Por exemplo, à taxa de juros armazenada em uma tabela de consumidores, uma referindo-se a praticada no Brasil e outra à praticada nos Estados Unidos. Apesar do dado ser o mesmo - taxa de juros, a informação é válida somente para o local onde é aplicada.
  - *Atribuição de nomes e representação:* a representação dos dados, o nome dos elementos dos dados e a estrutura do modelo de dados pode ser especificada nos bancos de dados locais, ocasionando diferenças na representação global.
  - *A compreensão, o significado e a interpretação dos dados:* esta é a principal causa da existência da heterogeneidade semântica. Refere-se a como os dados são compreendidos e interpretados.
  - *Transações e regras de consistência:* referindo-se ao critério utilizado para a serialização, compensação de transações e outras políticas utilizadas na realização de transações.
  - *Resumo de derivações:* esta questão está relacionada aos serviços suportados pelo sistema, como agregação, sumarização e outras operações de processamento de dados.

As diferenças na autonomia, distribuição e a heterogeneidade faz com que seja possível construir alternativas de implementação, variando o grau destas alternativas. A figura 2.3 demonstra a variação. As dimensões são identificadas como: A (autonomia),

D (distribuição) e H (heterogeneidade). As alternativas de autonomia são representadas por: 0, representando uma integração forte, 1 representando sistemas semi-autônomos e 3 representando total isolamento. Para a distribuição, 0 representa um sistema não distribuído, 1 representa sistemas client/server e 2 distribuição peer-to-peer. Com relação a heterogeneidade, 0 representa sistemas homogêneos e 1 representa sistemas heterogêneos.

Baseados na variação destes três fatores (ÖZSU, 1999) classifica a arquitetura dos bancos de dados distribuídos em três: sistemas *client/server*, *distributed databases* e *multidatabase systems.*

Com relação a heterogeneidade, os bancos de dados distribuídos podem ser classificados em dois tipos: *homogêneos* e *heterogêneos.* Se os dados são distribuídos mas todos os servidores executam o mesmo software de gerência de banco de dados , temos um *homogeneous distributed database system* (RAMAKRISHNAN & GEHRKE, 2000).

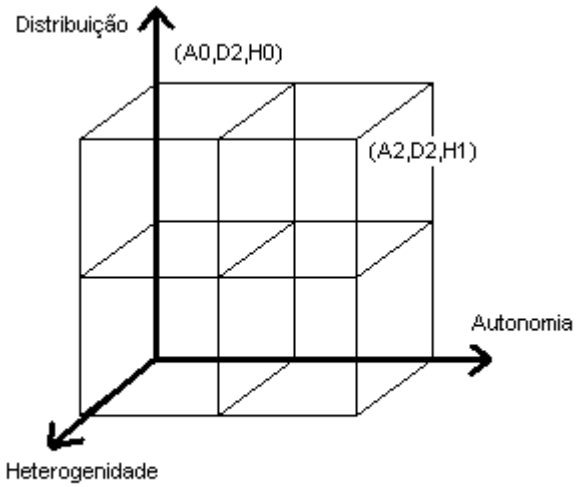


Figura 2.3. Alternativas de Implementação
Fonte: (ÖZSU 1999)

Se os diferentes sites estão sob controle de sistemas de gerência de banco de dados diferentes, essencialmente autônomos, interconectados de modo a permitir o acesso aos dados em múltiplos sites, temos um *heterogeneous distributed database system* (RAMAKRISHNAN & GEHRKE, 2000). Quando todos os servidores utilizam o mesmo sistema de gerencia de banco de dados, o database é chamado de *homogeneous*,

caso contrário, é chamado de *heterogeneous* (ATZENI et al. 2000). Se todos os servidores usam software idêntico e todos os usuários utilizam software idêntico, o sistema de gerência de banco de dados é chamado de *homogeneous;* caso contrário, é chamado de *heterogeneous* (ELMASRI & NAVATHE, 2000).

As alternativas de distribuição permitem dividir a base de dados em unidades lógicas chamadas *fragmentos*, e que podem ser armazenados em diferentes sites (ELMASRI & NAVATHE, 2000). Também permitem a *replicação* dos dados sobre diversos sites. A *fragmentação* consiste em dividir a relação em relações menores e armazená-los em diferentes sites (RAMAKRISHNAN & GEHRKE, 2000). A *fragmentação horizontal* consiste em subconjuntos de linhas da relação original. A *fragmentação vertical* consiste em subconjuntos das colunas da relação original (RAMAKRISHNAN & GEHRKE, 2000). A *replicação* lida com armazenar cópias de uma relação ou de uma relação fragmentada (RAMAKRISHNAN & GEHRKE, 2000). Uma relação pode ser armazenada em um ou mais sites. Isto é feito com o objetivo de (RAMAKRISHNAN & GEHRKE, 2000):

- **Incrementar disponibilidade dos dados:** Uma vez que, caso um site torne-se indisponível, os dados podem ser localizados em outro. Do mesmo modo, caso ocorra falha de comunicação, os dados permanecem disponíveis através da cópia local dos dados
- **Aumentar a velocidade na execução das consultas:** Sendo que uma consulta é executada mais rapidamente utilizando uma cópia local dos dados do que se fosse realizada em um site remoto.

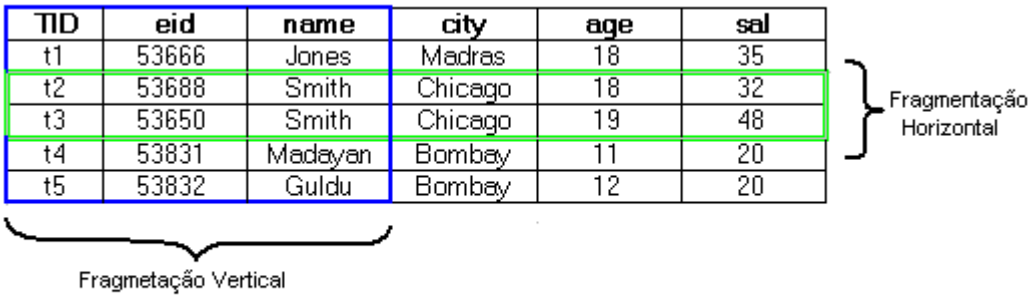


Figura 2.4. Fragmentação Horizontal e Vertical
Fonte: (RAMAKRISHNAN & GEHRKE 2000)

A distribuição tem implícito o aumento na complexidade no projeto do sistema e na implementação. Para usufruir das vantagens oferecidas por esta arquitetura, é necessário que os sistemas de gerência de banco de dados distribuídos resolvam alguns problemas adicionais aos encontrados nos bancos de dados centralizados (ÖZSU, 1999):

- **Distributed Database Design - Projeto de Distribuição do Banco de Dados:** esta questão é relativa ao modo como os dados são distribuídos e a manutenção e consulta dos dados. As alternativas básicas são o *particionamento (partitioned)* e *a replicação (replicated).* No particionamento os dados são divididos em partições e localizadas em diferentes sites. Na replicação os dados são duplicados em diversos sites. A replicação pode ser *totalmente replicada (fully replicated ou duplicated),* ou *parcialmente replicada (partially replicated ou duplicated).* A replicação total ocorre quando todo o banco de dados é armazenado em cada site. A replicação parcial é o armazenamento das partições em mais de um site, mas não em todos. A replicação dos dados também pode ser implementada fazendo-se uso da *fragmentação.*
- **Distributed Query Processing - Processamento de Consultas Distribuídas:** o processamento de consultas consiste da análise das consultas e a sua conversão em uma série de operações de manipulação dos dados. Esta questão é maior nos bancos de dados distribuídos, pois envolve a decisão de como implementar uma estratégia de execução para cada consulta sobre a rede e qual o custo disto (em termos de tempo e desempenho). Os fatores a serem considerados são a distribuição dos dados, o custo de comunicação e a ausência de informações sobre a disponibilidade local dos sites.
- **Distributed Directory Manangement - Gerência de Diretórios Distribuídos:** um diretório contém as informações sobre os dados do banco de dados (descrição e localização). O diretório pode ser global - para todo o banco de dados distribuído, ou local - para cada site. Pode ser centralizado ou distribuído entre diversos sites e, ainda, pode ter uma cópia ou diversas

cópias. Isto requer uma política de gerência da distribuição, localização e de projeto do banco de dados.

- **Distributed Concurrency Control - Controle de Concorrência Distribuído:** o controle de concorrência envolve a sincronização do acesso aos bancos de dados distribuídos, de modo que a integridade do banco de dados seja mantida. Nos bancos de dados distribuídos isto envolve a consistência de múltiplas bases de dados e múltiplas cópias das bases de dados - no uso de replicação. A condição de que todas as cópias dos diversos bancos de dados tenham o mesmo valor de atributo é chamada de *consistência mútua.*
- **Distributed Deadlock Manangement - Controle de Deadlock Distribuído:** a disputa entre os usuários pelo acesso aos recursos do sistema pode resultar na ocorrência de deadlock. Um sistema de banco de dados distribuído deve prover um mecanismo de prevenção, recuperação e detecção aplicável a todas as bases distribuídas.
- **Reability of Distributed DBMS - Confiabilidade em DBMS Distribuídos:** a disponibilidade e confiabilidade são uma das maiores vantagens dos bancos de dados distribuídos, isto implica em prover um mecanismo de consistência que detecte e recupere falhas em um ou mais sites. E que, na ocorrência de falha em um dos sites, o restante permaneça operante e consistente. Além disto, no momento em que o site ou a rede de comunicação volte a operar, o sistema deve providenciar mecanismos de recuperação e atualização dos dados entre os sites que permaneceram ativos e os sites onde ocorreu a falha.
- **Operating System Suport - Suporte do Sistema Operacional:** a implementação (instalação) dos sistema de banco de dados distribuídos sobre os sistemas operacionais faz com que ocorra um gargalo no desempenho. O suporte provido pelos sistemas operacionais às operações dos bancos de dados não corresponde de maneira apropriada às requisições do software de gerência do banco de dados. Os problemas mais comuns referem-se a sistemas mono-processados (single-processor), gerência de memória, sistema

de arquivos, métodos de acesso a arquivos, recuperação de falhas e gerenciamento de processos. Nos sistemas de bancos de dados distribuídos o problema torna-se mais complexo pelo fato de lidar com múltiplas camadas do software de rede.

- **Heterogeneous Databases - Bancos de Dados Heterogêneos:** quando os bancos de dados dos diversos sites não são homogêneos, em termos de modelo de dados (estrutura lógica dos dados) e mecanismos de acesso (linguagem de consulta), é necessário prover mecanismos de tradução entre os sistemas de banco de dados. Estes mecanismos envolvem uma forma canônica (conjunto de regras) que facilitem a tradução dos dados, bem como programas e modelos para tradução de instruções de manipulação dos dados.

Estas questões aumentam a complexidade dos bancos de dados distribuídos. A complexidade pode estar localizada, também, a nível de hardware, pelo fato de que, ao contrário dos sistemas centralizados, um banco de dados distribuído tem:

- Múltiplos computadores, chamados de sites ou nós.
- Cada site interligado por uma rede de comunicação, que pode ser local (LAN) ou remota (WAN), bem como com diferentes topologias de rede.

2.2.1 Sistemas Client/Server

Os sistemas cliente/servidor foram introduzidos no início dos anos 90. O princípio básico é a distinção das funcionalidades necessárias e a divisão destas funções em duas: funções servidores e funções clientes. Um sistema client/server tem um ou mais processos clientes e um ou mais processos servidores, e um processo cliente pode enviar uma consulta para um processo servidor (RAMAKRISHNAN & GEHRKE, 2000). O paradigma cliente/servidor é um modelo de interação entre processos de software, onde os processos são subdivididos em clientes (que requerem serviços) e servidores (que oferecem serviços) (ATZENI et al., 2000). Os clientes são responsáveis pelas questões relativas interface dos usuários e os servidores gerenciam os dados e executam as transações (ATZENI et al., 2000). A figura 2.5 mostra a arquitetura dos sistemas cliente/servidor.

Existem diversos tipos de arquiteturas cliente/servidor. O caso mais simples é onde somente um servidor é acessado por diversos clientes. É chamado de *multiple client-single server* (ÖZSU, 2000). Uma arquitetura mais sofisticada é onde existem diversos servidores no sistema, chamado de *multiple client-multiple server* (ÖZSU, 2000).

Não necessariamente os processos clientes e servidores devam ser alocados em máquinas diferentes (ATZENI et al., 2000). As razões para o uso da arquitetura client/server, segundo (ATZENI et al., 2000) são:

- As funções client e server são identificadas no contexto do banco de dados. O programador da aplicação é responsável pela escrita, controle e gerência do software de modo a fazer o cliente responder a demandas específicas do usuário. O administrador do banco de dados (DBA) tem a responsabilidade de gerenciar e planejar os dados no servidor, compartilhados por vários clientes, de modo a garantir a melhor serviço para todos os processos clientes.
- A partir da divisão funcional de processos e tarefas, o uso de computadores diferentes para clientes e servidores é, particularmente, conveniente. O computador dedicado ao cliente deve ser adequado para interação com o usuário - computadores pessoais, providos com ferramentas de produtividade pessoal (planilhas, editores, etc.), bem como com interfaces “user-friendly”. A capacidade do computador servidor depende dos serviços que deve prover. Basicamente deve ter grande capacidade de memória e de armazenamento.
- A linguagem SQL oferece um paradigma de programação ideal para a “service interface”. Deste modo, os clientes formulam as consultas em SQL e as enviam para o servidor. O servidor processa a consulta e remete o resultado para o cliente. Analisado pelo lado da rede, somente a informação necessária ao cliente irá trafegar, representando uma fração dos dados. Em adição, a portabilidade e interoperabilidade da linguagem SQL permite a construção de aplicações que envolvam diferentes sistemas servidores.

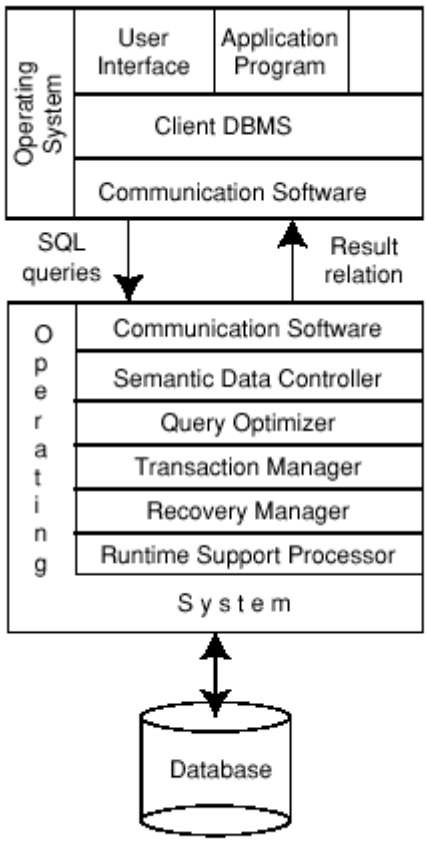


Figura 2.5. Arquitetura Client/Server
Fonte: (ÖZSU 1999)

2.2.2 Sistemas Distributed Database

Na arquitetura client/server as transações envolvem um único servidor. Quando mais de um servidor é envolvido, é chamado de *distributed database* (ATZENI et al., 2000). Conforme (RAMAKRISHNAN & GEHRKE, 2000) a arquitetura cliente/servidor não permite que uma consulta seja distribuída para múltiplos servidores pelo fato de um processo cliente não ser capaz de dividir a consulta em sub-consultas a serem executas em sites diferentes e juntá-las de modo a atender a consulta. Os sistemas *collaborating server* são uma alternativa à arquitetura client/server, consistindo de uma coleção de servidores de banco de dados, cada um capaz de executar transações sobre os dados locais, cooperando na execução de transações entre os diversos servidores. Para isto, quando um servidor recebe uma consulta que requer o acesso a dados de outros

servidores, ele gera um conjunto de sub-consultas a serem executadas pelos outros servidores, juntando os resultados e respondendo a consulta original. Isto é possível separando os módulos que atendem as requisições dos usuários, dos módulos que gerenciam os dados.

A funcionalidade deste sistema é feita definindo em cada site um *local internal schema (LIS)* - que contém a descrição da organização física dos dados. Como os dados do banco de dados distribuídos podem estar fragmentados e replicados, é necessário um *local conceptual schema (LCS)* - que lida com a fragmentação e a replicação. A definição geral dos dados é descrito num *global conceptual schema (GCS)* que descreve a estrutura lógica de todos os sites, constituindo-se na união dos esquemas LIS e LCS de todos os sites. As aplicações dos usuários acessam o banco de dados através de *external schemas (ESs)* definidos acima do esquema conceptual global (ECS) (ÖZSU, 1999).

Os detalhes funcionais deste modelo utilizam um *global directory/dictionary (GD/D)* que permite o mapemento global requerido. Os mapeamentos locais são executados por um *local directory/dictionary (LD/D).* Os componentes deste modelo são divididos em dois: um componente chamado de *user processor;* e outro componente chamado de *data processor.* O primeiro é encarregado de manipular as requisições dos usuários, enquanto que o segundo encarrega-se do armazenamento (ÖZSU, 1999).

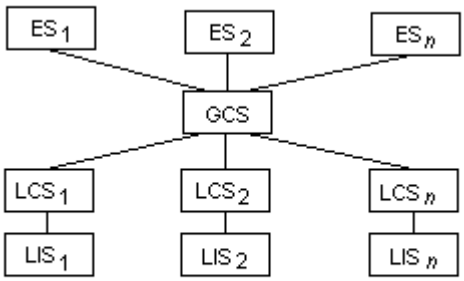


Figura 2.6. Arquitetura de Referência dos Bancos de Dados Distribuídos
Fonte: (ÖZSU 1999)

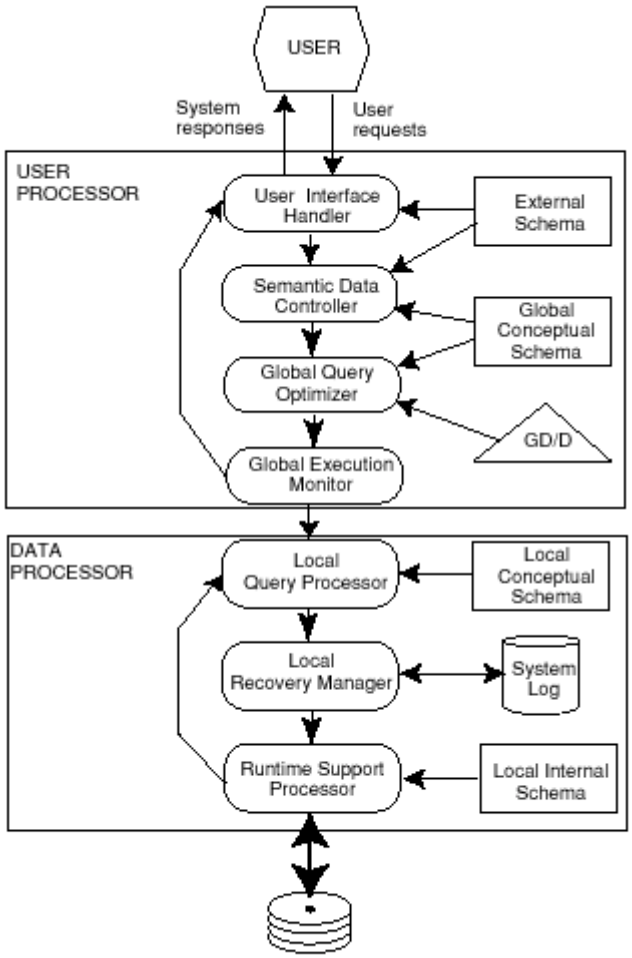


Figura 2.7. Funcionalidade dos DBMS Peer-to-Peer
Fonte: (ÖZSU 1999)

O componente *user processor* consiste de quatro elementos (ÖZSU, 1999):

1. Uma *user interface handler*, responsável pela interpretação dos comandos do usuário.
2. Um *semantic data controller* que, através de restrições de integridade e autorizações definidas no esquema conceptual global, checa se a consulta do usuário pode ser processada.
3. Um *global query optimizer and decomposer* que determina a estratégia de execução de modo a minimizar o custo de execução, e traduz as requisições globais em consultas locais. Utiliza os esquemas globais, locais e o diretório global para isto.

4. Um *distributed execution monitor* que coordena a execução distribuída da requisição. Este elemento também é chamado de *distributed transaction mananger.*

O componente *data processor* é composto por três elementos:

1. O *local query optimizer*, que atua como um *access path selector (selecionador do modo de acesso)*, responsável pela escolha da melhor forma de acessar os dados.
2. Um *local recovery manager* que é responsável por assegurar que o banco de dados local permaneça consistente.
3. Um *run-time support processor* que acessa a base de dados conforme os comandos gerados pelo otimizador de consultas.

As razões para o desenvolvimento de soluções distribuídas na gerência de dados são pragmáticas. Esta arquitetura responde a demanda, estruturando os dados em uma arquitetura que corresponde a forma como os dados são usados e produzidos, uma vez que as empresas são estruturalmente distribuídas (ATZENI et al., 2000).

2.2.3 Sistemas Multidatabase

A diferença entre um sistema multidabase de um sistema de banco de dados distribuído estão no nível de autonomia e são refletidas na sua arquitetura (ÖZSU, 1999). A diferença fundamental está na definição do esquema conceptual global. No caso dos bancos de dados distribuídos, o GCS define a visão de todo o banco de dados, enquanto que no caso dos bancos de dados distribuídos multidatabase, representam a coleção de alguns dos bancos de dados locais, com os quais o banco de dados local compartilha dados. Desta forma, a definição *global database* é diferente num banco de dados MDBSs do que num DBMSs (ÖZSU, 1999)**.**

A arquitetura *middleware* é projetada de modo a permitir que uma consulta seja transportada para múltiplos servidores, sem requerer que todos os bancos de dados sejam capazes de gerenciar estratégias de execução multi-site. Isto é especialmente atrativo quando lida-se com integração de sistemas legados, onde suas capacidades básicas não podem ser estendidas (RAMAKRISHNAN & GEHRKE, 2000). A idéia é

que é necessário somente um servidor de banco de dados que seja capaz de gerenciar consultas e transações entre diversos servidores; os servidores remanescentes somente necessitam manipular consultas e transações locais. (RAMAKRISHNAN & GEHRKE, 2000). Conforme ÖZSU (1999) a definição de *global database* é diferente nos MDBSs do que nos DBMSs distribuídos. Nos DDBMS o database global é a união de todos os bancos de dados locais, enquanto que nos MDBSs o mesmo é formado por um subconjunto desta mesma união. A modelo de arquitetura baseada em componentes (compenent-based) de um multi-DBMS é significantemente diferente do DBMS distribuído. A diferença fundamental é a existência de DBMS completos, cada um gerenciando um banco de dados diferente. O MDBS prove um nível de software que é executado sobre os DBMSs individuais e provê os usuários com facilidades de acesso a vários bancos de dados (ÖZSU, 1999).

2.2.3.1 Modelo Utilizando um Global Conceptual Schema

Em um sistema de banco de dados multidatabase o GCS é definido pela integração dos esquemas conceituais locais ou dos esquemas externos (ÖZSU, 1999). Entretanto, os usuários do SGBD local definem suas próprias visões no banco de dados local e não precisam trocar suas aplicações caso não necessitem acessar dados de outro banco de dados (ÖZSU, 2000). A maior diferença no design do GCS nos multi-DBMS dos logicamente integrados bancos de dados distribuídos é no modo de mapear os esquemas conceituais locais para o esquema global. Nos multidabase o mapeamento é feito de modo reverso (ÖZSU, 1999). Isto se deve pela heterogeneidade encontrada nos sistemas multidatabase onde um modelo de dados canônico não é encontrado que seja base para a definição do GCS (ÖZSU, 1999). Desta forma, duas alternativas de implementação existem: unilingual e multilingual. A alternativa *unilingual* requer que os usuários utilizem modelos de dados e linguagem possivelmente diferentes nos bancos de dados locais e globais.

A característica que identifica os sistemas unilinguais é que qualquer aplicação que acesse dados de múltiplos databases o façam por meio de uma visão externa definida no esquema conceptual global.

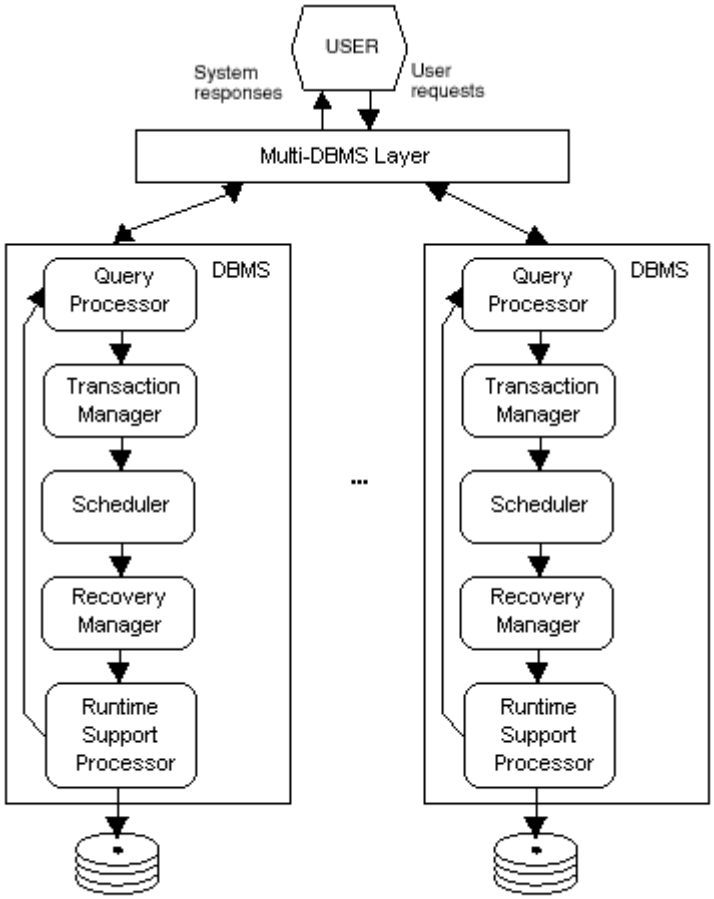


Figura 2.8. Componentes de um MDBS
Fonte: (ÖZSU 1999)

Isto faz com que os usuários do banco de dados global sejam diferentes dos usuários que acessem somente o banco de dados local, utilizando um modelo de dados diferente e uma linguagem diferente. Então, uma aplicação deve ter um *local external schema (LES)* definida no esquema conceptual local e um *global external schema (GES)* definido no esquema conceptual global (ÖZSU 1999). A figura 2.9 apresenta esta arquitetura.

A arquitetura *multilingual* tem como filosofia básica permitir que cada usuário acesse o banco de dados global (ou dados de outros bancos de dados) utilizando um esquema externo, sendo definido utilizando a linguagem do SGBD local. Assumindo que a definição é puramente local, a consulta feita conforme um esquema particular é manipulada exatamente como qualquer consulta nos SGBDs centralizados.

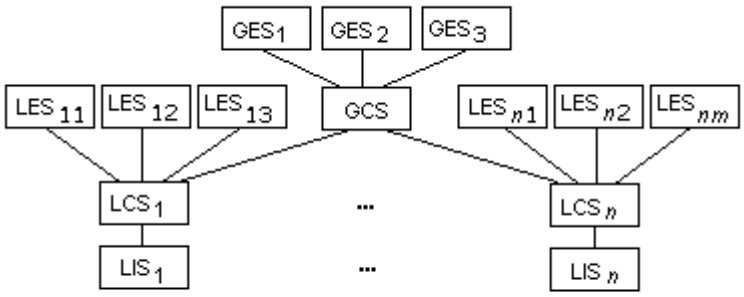


Figura 2.9. Arquitetura MDBS com GCS
Fonte: (ÖZSU 1999)

As consultas sobre o banco de dados global são feitas na linguagem do SGBD local, requerendo processamento extra de modo a mapeá-la para o esquema conceptual global (ÖZSU, 1999). O enfoque multilingual torna as consultas aos bancos de dados mais fácil sob a visão do usuário. É, no entanto, mais complicada devido ao fato de lidar com traduções de consultas em tempo de execução (ÖZSU, 1999).

2.2.3.2 Modelo Sem um Global Conceptual Schema

A existência de um esquema conceptual global em sistemas multidabase é uma questão controversa (ÖZSU, 1999). A arquitetura é composta por dois níveis: o nível do sistema local e o nível multidatabase. O nível do sistema local (local system level) consiste em um número de SGBDs, que fornecem ao nível multidatabase partes do banco de dados local, que são compartilhados com usuários de outros bancos de dados. Este compartilhamento é feito com base no esquema conceptual local ou definido no esquema externo local. Caso exista heterogeneidade, cada um destes esquemas, LCS_I , pode utilizar um modelo de dados diferente (ÖZSU, 1999).

As visões externas são construídas sobre o nível multidatabase, onde cada visão pode ser definida em um esquema conceptual local ou sobre múltiplos esquemas conceituais. Deste modo, a responsabilidade de prover acesso a múltiplos (e possivelmente heterogêneos) bancos de dados é delegada ao mapeamento entre os esquemas externos e os esquemas conceituais locais. Esta é a principal diferença dos modelos que usam um esquema conceptual global, onde esta responsabilidade é feita sobre o mapeamento

entre o esquema conceptual global e o esquema conceptual local. Esta transferência de responsabilidade tem um conseqüência prática. O acesso a múltiplos bancos de dados é provido por meio de uma poderosa linguagem na qual as aplicações dos usuários são escritas (ÖZSU 1999). A figura 2.10 apresenta a arquitetura de um multidatabase system sem um global conceptual schema.

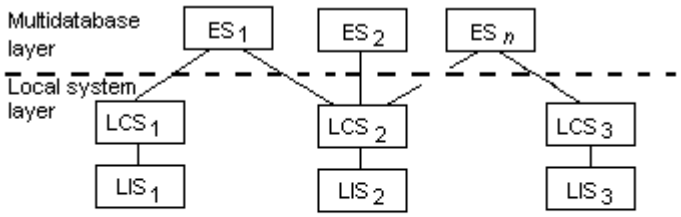


Figura 2.10. Arquitetura MDBS Sem um GCS
Fonte: (ÖZSU 1999)

2.2.3.3 Federated Database Systems

Com relação a autonomia dos bancos de dados, temos SGBDD que são "vistos como" pelo usuário como um SGBD centralizado. Existe um único esquema conceptual e todo o acesso ao sistema é obtido através do site que é parte do DDBMS, significando que não existe autonomia local. Em outro extremo, encontra-se um tipo de DBMS chamado de *federated DDBMS*. Neste sistema, cada servidor é um independente e autônomo DBMS centralizado, que tem seus próprios usuários locais, transações locais, e DBA e, por conseguinte um alto grau de autonomia. O termo *federated database system (FDBS)* é utilizado quando existe alguma visão global ou esquema da federação de banco de dados que é compartilhado pelas aplicações (ELMASRI & NAVATHE, 2000). Um federated database system (FDBS) é uma coleção de cooperativos mas autônomos bancos de dados (DBSs) (SHETH & LARSON, 1990). OS DBS componentes são integrados em diversos graus. O software que provê o controle e a coordenação dos DBSs componentes é chamado de *federated database management system (FDBMS)*. O DBMS de um DBS componente, ou o componente DBMS, pode ser centralizado ou distribuído ou outro FDBMS (SHETH & LARSON, 1990). Um típico esquema de cinco níveis da arquitetura de um FDBS é demonstrada na figura

2.11. Nesta arquitetura, o *local schema - esquema local*, é o esquema conceptual de um banco de dados componente, e o *esquema do componente - component schema*, é derivado da tradução do esquema local em um esquema canônico ou modelo comum de dados - *common data model (CDM*) para o FDBS. O esquema de tradução do esquema local para o esquema do componente é acompanhado da geração de um mapeamento, de modo a transformar os comandos de um esquema do componente em comandos no esquema local correspondente. O esquema exportado - *export schema*, representa um subconjunto de esquemas do componente que estão disponíveis no FDBS. O esquema federado - *federated schema*, é o esquema global ou visão, resultante da integração de todos os esquemas exportados compartilhados. O esquema externo - *external schema*, define o esquema para um grupo de usuários ou uma aplicação, numa arquitetura em três níveis (ELMASRI & NAVATHE, 2000).

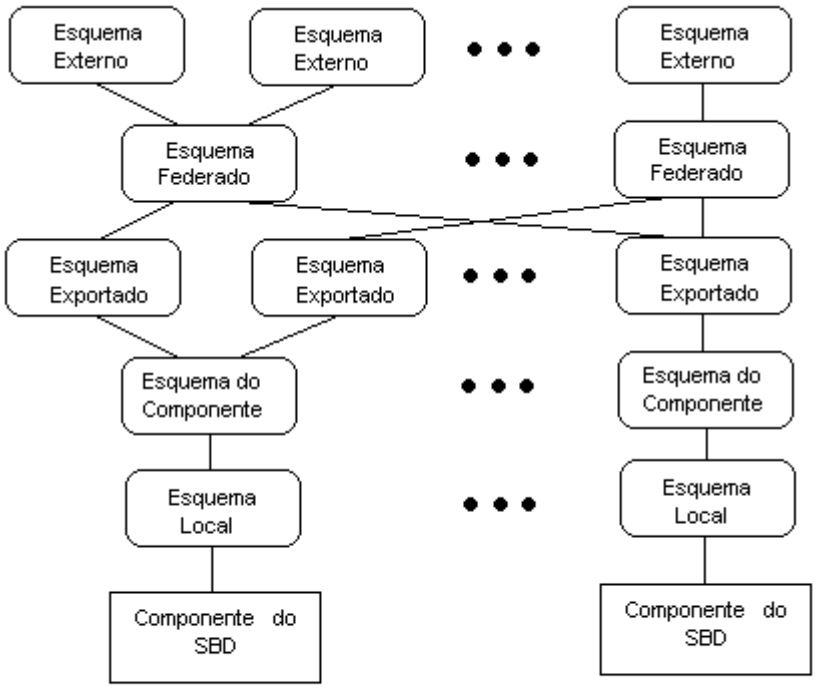


Figura 2.11. Níveis dos Federated Database Systems
Fonte: (ELMASRI & NAVATHE 2000)

2.3 TAXONOMIA DOS SISTEMAS MULTIDATABASE

Um SGBD *distribuído* consiste em um único SGBD gerenciando múltiplos bancos de dados. Estes bancos de dados podem residir em um único ou em múltiplos sistemas de computação que podem ser diferentes em hardware, software do sistema e suporte à comunicação (SHETH & LARSON, 1990).

Um *multidatabase system* (MDBS) suporta operações em múltiplos DBSs componentes. Cada componente é gerenciado por um SGBD. Um sistema de banco de dados componente de um MDBS pode ser centralizado ou distribuído e pode residir em um ou em múltiplos computadores conectados por um subsistema de comunicação. Um multidatabase system é chamado de homogêneo se todos os componentes são os mesmos, de outra forma, é chamado de heterogêneo (SHETH & LARSON, 1990).

Diferentes arquiteturas e tipos de FDBSs são criados pelos diferentes níveis de integração dos sistemas de banco de dados componentes e pelos diferentes níveis de serviços (federação) globais (SHETH & LARSON, 1990).

Um MDBS é classificado em dois tipos, baseado na autonomia dos seus componentes: *nonfederated databases systems* e *federated database systems.* Um *nonfederated database system* é uma integração de DBMSs componentes que não são autônomos. Possuem um único nível de gerência e todas as operações são feitas de modo uniforme. Nestes sistemas não há distinção entre usuários locais e usuários globais (SHETH & LARSON, 1990).

Um *federated database system* consiste de DBS componentes que são autônomos e participantes da federação, permitindo um controlado e parcial compartilhamento dos seus dados. Os componentes cooperam em diferentes graus de integração, não existindo um controle centralizado pelo fato de que seus componentes controlam o acesso aos seus dados (SHETH & LARSON, 1990).

Uma federação é construída através de uma seletiva e controlada integração dos seus componentes. Um *loosely coupled* FDBS sempre suporta múltiplos esquemas federados. Um *tightly coupled* FDBS pode ter um ou mais esquemas federados. Um *tightly coupled* FDBS é dito como sendo *sigle federation* se suporta a criação e o gerenciamento de um único esquema federado. Um *tightly coupled* FDBS é dito como

sendo *multiple federation* se permite a criação e o gerenciamento de múltiplos esquemas federados (SHETH & LARSON, 1990). A figura 2.12 apresenta a taxonomia dos sistemas multidatabase.

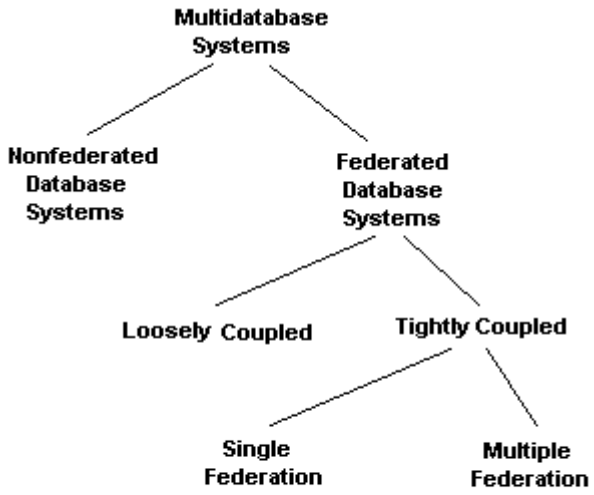


Figura 2.12. Taxonomia dos Sistemas Multidatabase
Fonte: (SHETH & LARSON 1990)

CONCLUSÃO

Os sistemas de banco de dados distribuídos são compostos da união da tecnologia dos bancos de dados e das redes de computadores. E devem gerênciar os dados com diferentes níveis de transparência, sendo: *transparência de rede*, dividindo-se em *transparência de localização* - onde o usuário não necessita conhecer os detalhes de onde a base de dados está localizada e *transparência de nomeação*, que é a atribuição de um nome único a cada objeto do banco de dados (tabelas, views, etc); *transparência de replicação*, que é relativa a realizar cópia dos dados sobre diversos sites; *transparência de fragmentação*, dividida em *fragmentação horizontal* - que consiste na divisão das tabelas em subconjuntos de tuplas, e *fragmentação vertical* que é a divisão da tabela em subconjuntos de colunas. E devem implementar disponibilidade, segurança, desempenho e facilidade de expansão do sistema.

O software de gerência dos bancos de dados distribuídos deve prover funções para: *armazenar a localização os dados* - referindo-se a necessidade de conhecer a

distribuição, fragmentação e replicação dos dados; *processar consultas distribuídas* - implicando em acessar e transmitir dados de sites remotos através de uma rede de comunicação; *gerenciar as transações distribuídas* - que se refere em desenvolver estratégias de execução de transações em mais de um site; *gerenciar a replicação dos dados* - que consiste em decidir qual copia dos dados será acessada e em manter a consistência entre as cópias; *recuperação distribuída do banco de dados* - que é a capacidade de recuperar-se de falhas em sites individuais ou de falhas de comunicação; *segurança* - onde as transações distribuídas devem ser executadas sobre privilégios de acesso dos usuários; e *gerência de diretório distribuída* - referindo-se a conter informações relativas a localização e distribuição dos diretórios.

A distribuição pode ser realizada sobre três elementos: funções, dados e controle. Sendo que três arquiteturas básicas são propostas: Sistemas *shared-memory* - onde múltiplas cpus compartilham uma memória principal, *sistemas shared-disk* - onde cada cpu compartilham dispositivos de memória secundária, e *sistemas shared-nothing* - caracterizados pela comunicação entre as cpus por uma rede de comunicação.

Os sistemas de banco de dados distribuídos tem como característica o armazenamento dos dados distribuído nos sites, sendo cada site gerenciado por um SGBD independente. Suas propriedades principais são: *independência na distribuição dos dados* - que consiste na realização de transações sem a necessidade de referenciar onde as relações estão armazenadas, e *atomicidade nas transações distribuídas* - que é a realização de transações sobre diversos sites da mesma forma que seriam realizadas sobre os dados locais.

A arquitetura dos bancos de dados distribuídos é classificada sob três aspectos: autonomia, distribuição e heterogeneidade. A *autonomia* refere-se a distribuição do controle e indica o grau em que cada SGBD local opera e pode ser de design, comunicação, execução e associação. A *distribuição* refere-se a forma de distribuição dos dados e podem ser agregados em duas classes: distribuição cliente/servidor e distribuição peer-to-peer. A distribuição cliente/servidor ocorre a distinção entre equipamentos cliente e equipamentos servidores, realizando a distribuição de funções. Nos sistemas peer-to-peer cada sistema é um SGBD completo que comunica-se com outros sistemas para a execução de transações. A *heterogeneidade* pode ocorrer de

vários modos, iniciando da heterogeneidade de hardware e indo até variações nos gerenciadores dos dados. A heterogeneidade dos SGBD é dividida entre as que diferenciam o sistema de gerência de banco de dados e das que diferenciam a semântica dos dados. Sendo que a heterogeneidade nos sistema decorre da evolução tecnológica dos SGBD. Os diferentes graus de autonomia, distribuição e heterogenidade permite a classificação da arquitetura dos bancos de dados em três sistemas: sistemas *client/server, distributed databases* e *multidatabase systems.*

O sistema cliente/servidor parte do princípio de dividir as funções em funções clientes e funções servidoras. Onde um processo cliente envia um consulta a um processo servidor, ocorrendo uma interação entre processos.

O sistema distributed database é uma arquitetura peer-to-peer, onde cada site tem a mesma funcionalidade. Sendo capaz de executar transações sobre diversos sites e coletando as respostas de cada servidor, atendendo a consulta original. A funcionalidade deste sistema é feita:

a) definindo-se em cada site local um LIS - esquema interno local, que contem a descrição dos dados;

b) definindo-se um LCS - esquema conceitual local, que lida com a fragmentação e replicação dos dados;

c) a definição de um GCS - esquema conceitual global, que descreve a estrutura lógica de todos os sites.

O sistema multidatabase diferencia-se de um sistema distributed database no nível de autonomia dos sites. Nestes sistemas, uma arquitetura midleware é projetada de modo que uma consulta seja transportada para diversos servidores sem requerer que todos os bancos de dados sejam capazes de gerenciar estratégias de execução multi-site. Este sistema utiliza dois modelos de esquemas conceituais globais. Um modelo é definido com o uso de um GCS, sendo formado pela integração dos esquemas locais, Duas forma de implementação são propostas: a alternativa *unilingual* requer que o acesso aos múltiplos bancos de dados seja feita por um esquema conceitual global e utilizando um modelo de dados e uma linguagem de consulta diferente, sendo necessário mapear as consultas locais na sintaxe das consultas globais. A alternativa *multilingual,* permite o acesso as bases de dados utilizando a linguagem do banco de dados local.

Outro modelo definido é feito sem a definição de um GCS, neste caso a arquitetura é composta por um nível local e um nível multidatabase. O nível local fornece ao nível multidatabase partes do banco de dados local que são compartilhados com os usuários dos bancos de dados com base num esquema externo local (ES).

Um caso particular de sistemas multidatabase são os Federated Database Systems. Neste sistema, cada banco de dados é um sistema de gerência de banco de dados autônomo e independente, com alto grau de autonomia. Nesta arquitetura, o esquema local é traduzido para um esquema componente, que gera um esquema exportado. O conjunto dos esquemas exportados gera o esquema federado.

3 Processamento de Consultas

Introdução

Neste capítulo é estudado o processamento de consultas em banco de dados relacionais. São descritos os modos como as consultas são realizadas, otimizadas e implementadas e apresentados os algoritmos básicos para o processamento de consultas. São descritos o processo de otimização de consultas, os algoritmos utilizados e os tipos de otimização nos bancos de dados centralizados e nos banco de dados distribuídos. São mostradas as otimizações baseadas em regras herísticas, por seletividade e estimativa de custo e otimização semântica. Apresenta o processamento de consultas e otimização em sistemas de banco de dados distribuídos.

O estudo desenvolvido neste capítulo tem o objetivo de fundamentar e apresentar o processo de realização de consultas nos sistemas de banco de dados distribuídos, visto que este processo é inerente ao uso dos mesmos e necessário para avaliar os recursos que os SGBDD devem provêr na distribuição, fragmentação e replicação dos dados.

3.1 Processamento de Consultas em Banco de Dados

Os bancos de dados relacionais utilizam uma linguagem de consulta aos dados de alto nível, que esconde do usuário os detalhes da organização física dos dados, possibilitando um aumento significativo na produtividade e facilidade no desenvolvimento de aplicações. A construção de consultas a base de dados é feita por meio de uma linguagem não-procedural, de alto nível, e onde o usuário não tem a necessidade de especificar como o procedimento será executado. A consulta expressada em uma linguagem de alto nível, como a linguagem SQL, deve primeiro ser analisada e validada (ELMASRI & NAVATHE, 2000). A consulta será submetida a um módulo do sistema de gerência do banco de dados (DBMS) chamado *processador de consultas (query processor).* O processador de consultas analisa, otimiza e executa a consulta de modo que seja executada de modo mais rápido e com menor custo (de tempo e

processamento). O processo de analise da consulta é feito em etapas. A primeira etapa é o *scanner (exame),* que tem o objetivo de identificar os pontos da linguagem, como comandos sql, nomes de atributos e relacionamentos. A partir disto, a etapa - chamada de *parser (análise),* checa a sintaxe e determina se a mesma foi construída de acordo com as regras de gramática da linguagem de consulta. Após, a consulta passa para a fase de *validated* (validação), que é feita através da checagem da validade e significância semântica de todos os atributos e nomes de relação no esquema do banco de dados particular, onde é iniciada a consulta. A partir disto é feito uma representação interna da consulta, através de uma estrutura em árvore, chamada *query tree (árvore de consulta),* e a representação da mesma em uma estrutura gráfica chamada *query graph (gráfico de consulta).* É elaborado uma *execution strategy (estratégia de execução),* que busca extrair os resultados da consulta do banco de dados. Uma consulta pode ser executada de diversas formas. A estratégia de execução deve envolver uma etapa de escolha de um modo, entre os diversos possíveis, de realizar a consulta. Esta fase é chamada de *query optimization (otimização da consulta).* Após a otimização da consulta é produzido um plano de execução - chamado *code generator (gerador de código),* que codifica o plano de execução e o submete ao runtime do banco de dados para o seu processamento.

A busca pela melhor forma de otimização pode consumir um tempo de processamento que, possivelmente seria maior do que o dispendido na própria execução da consulta, visto que envolve a análise de diversas formas ou modos de realizar a consulta. Este fato faz com que a busca pela otimização seja feita de modo a buscar uma estratégia razoável e eficiente *(reasonably efficient strategy).* As duas técnicas principais de otimização são baseadas em *regras heurísticas (heuristic rules)* e *estimativa (systematically estimating).* A técnica baseada em heurística busca ordenar as operações na estratégia de execução da consulta. A ordenação se dá através do conhecimento heurístico de regras que proporcionem o melhor desempenho na maioria dos casos, mas não há garantia de que seja a melhor solução para todos os casos. A técnica baseada em estimativas visa avaliar sistematicamente o custo da estratégia de execução e escolher o plano de execução de menor custo. As duas técnicas são, geralmente, combinadas no otimizador de consultas.

Segundo ÖZSU (1999), o objetivo da otimização é encontrar uma estratégia mais adequada e, talvez mais importante, prevenir estratégias ruins. A seleção de uma mais adequada requer a predição dos custos de execução sobre alternativas candidatas à execução da consulta. A expressão de custo é uma carregada combinação de custos de I/O, CPU e comunicação (ÖZSU, 1999). A otimização da consulta refere-se ao processo de produzir um plano de execução (QEP) que represente uma estratégia de execução para a consulta. O plano selecionado tem o objetivo de minimizar o custo. O otimizador é um módulo de software que realiza a otimização da consulta e possui três componentes:

1. **Search Space:** Os planos de execução são abstraídos através de árvores de operadores que definem a ordem em que as operações serão executadas. Estes planos são enriquecidos com informações adicionais como o melhor algoritmo para cada operação. Para uma determinada árvore, o espaço de busca pode ser definido como um conjunto de árvores de operações equivalentes, produzidas por meio de regras de transformação.
2. **Search Strategy:** A estratégia mais popular utilizada pelos otimizadores de consulta é a *dynamic programing*, que é determinística. A estratégia determinística busca a construção de planos, iniciando da relação base, juntando uma ou mais relações a cada passo, até a obtenção de um plano completo. A programação dinâmica constrói todos os planos possíveis e, após, seleciona o melhor. Outra estratégia determinística constrói somente um plano de execução. A programação dinâmica é, também, exaustiva e assegura que o melhor plano seja encontrado. Isto pode tornar-se muito dispendioso - em termos de custo. Outra estratégia de busca é a *randomized***.** Esta estratégia concentra a busca em sobre alguns pontos particulares. Isto não garante que o melhor plano seja encontrado, mas previne um alto custo de otimização, em termos de consumo de memória e tempo. Este método constrói um ou mais planos iniciais, e, através da aplicação de um plano randômico de transformação, aprimora a consulta.
3. **Distributed Cost Model:** Um modelo de custo inclui funções de modo a predizer o custo das operações, estatísticas, base de dados e formulas, de

modo a avalizar o tamanho dos resultados intermediários. Um modo de custo para uma estratégia de execução distribuída pode ser expressada com respeito a tempo total ou tempo de resposta. O tempo total é a soma de todos os componentes de tempo; o tempo de resposta é o tempo dispendido a partir do início até o término da consulta. Uma fórmula geral pode ser especificada como:

$$\textbf{Total\_time} = T_{CPU} * \#insts + T_{I/Os} + T_{MSG} * T_{TR} * \# bytes$$

Onde os dois primeiros tempos T_{CPU} e $T_{I/O}$ são tempos de processamento locais. T_{CPU} é o tempo de instruções da CPU, $T_{I/O}$ é o tempo de entrada e saída em disco. T_{MSG} e T_{TR} são tempos de comunicação, sendo T_{MSG} o tempo de inicializar e receber uma mensagem e T_{TR} o tempo para transmitir dados de um site para outro. Nos ambientes distribuídos a topologia da rede influencia grandemente a proporção entre estes componentes. Numa rede wide - como a Internet, o tempo de comunicação é um fator dominante. Numa rede local, há um balanceamento entre estes componentes.

3.2 Algoritmos Básicos para a Execução de Consultas

Segundo ELMASRI & NAVATHE (2000), um sistema de gerência de banco de dados (SGBD) utiliza algoritmos para implementar os diferentes tipos de operações relacionais. As operações incluem operações básicas e avançadas de álgebra relacional, bem como a combinação destas. A cada operação, um ou mais algoritmos são utilizados, visto que, um determinado algoritmo pode ser aplicável a uma estrutura de dados particular e são utilizados por outros algoritmos envolvidos na operação. Desta forma, as operações relacionais combinam diversos algoritmos (ELMASRI & NAVATHE, 2000).

- **External Sorting:** este algoritmo é aplicável em grandes arquivos e registros armazenados em disco e que não são possíveis de serem armazenados na memória principal (ELMASRI & NAVATHE, 2000). Um algoritmo de sort externo utiliza uma estratégia de sort-merge, iniciando pela classificação

(sort) de sub-arquivos do arquivo principal e juntando-os (merge), criando um arquivo classificado.

- **Implementação de Operações de Seleção:** existem diversas opções para a execução de operações de seleção: alguns dependem do arquivo ter um caminho de acesso específico e outros são aplicáveis somente a certos tipos de condições de seleção (ELMASRI & NAVATHE, 2000). Os principais algoritmos de implementação de operações de seleção são (ELMASRI & NAVATHE, 2000):
  - **Métodos de Busca de Operações de Seleção Simples:**
    1. *Linear search ou Busca linear:* este método lê cada registro do arquivo e testa se o mesmo atende as condições de seleção.
    2. *Binary serch ou Busca binária:* este método é utilizado se a condição de seleção envolver uma operação de igualdade em uma chave (key). A busca binária é realizada sem a necessidade de classificação, pelo fato de que a chave definida encontra-se ordenada e armazenada no arquivo de índices.
    3. *Busca por primary index (hash key) ou chave primária:* neste caso, a operação de seleção deve envolver uma condição de igualdade sobre uma chave primária (hash key). A busca desta forma é mais eficiente pelo fato de ser possível a localização através do arquivo de índices.
    4. *Utilizando chave primária para recuperar múltiplos registros:* se a comparação for uma operação lógica (>, <, >=, <=) sobre uma chave primária, a busca pelo intervalo de chaves que satisfaçam a condição é mais eficiente.
    5. *Utilizando uma chave clustering para recuperar múltiplos registros:* uma chave clustering é o valor de um atributo que ordena um arquivo, mas que se referencia a diversos valores iguais (diferente da chave primária que requer que o valor do atributo seja único). Uma chave clustering ordena os registros em blocos (block cluster) e gera um arquivo de chaves que refere-se ao bloco como um todo, e não a cada registro. Se a operação de seleção envolver uma comparação de

igualdade em um atributo que não seja chave (non-key attribute) com uma chave clustering, a busca é feita pelo uso desta chave.

6. *Utilizando uma chave secundária:* este método é utilizado para procurar os registros pelo valor da chave, visto que a classificação é feita sobre o arquivo de índices.

- **Métodos de Busca para Operações de Seleção Complexas.** Uma operação de seleção é considerada complexa quando envolver um conjunção, como por exemplo a condição lógica AND.
  1. *Conjunção usando chaves:* neste caso, a condição de busca e conjunção deve permitir o uso dos métodos 1 e 2 das operações de seleção simples e a verificação ou checagem dos registros que satisfaçam a condição de conjunção.
  2. *Conjunção utilizando chaves compostas:* quando dois ou mais atributos são envolvidos na condição de busca e conjunção e uma composta (formada por dois ou mais atributos) existir, a operação será realizada utilizando-se a chave composta.
  3. *Conjunção pela interseção de ponteiros de registros:* se a operação de busca envolver o uso de chaves secundárias em um ou mais atributos especificados na condição de seleção e se estas chaves forem ponteiros de registros ou ponteiros de blocos de registros, as chaves podem ser utilizadas para a recuperação de conjuntos de ponteiros de registro que satisfaçam a condição. A interseção destes conjuntos de ponteiros é feita selecionando-se os registros que satisfaçam a operação de conjunção e usados para recuperar o registro. Se somente uma condição tem chave secundária, cada registro é testado para determinar se satisfaz as condições restantes.
- **Implementação de Operações de Junção (Join):** a operação de junção é uma das que consomem mais tempo no processamento de consultas. Quando a operação de junção envolver dois arquivos, é chamada de *two-way join.* As operações de junção que envolvem mais de dois arquivos é chamada de *multiway join.* Os métodos para a implementação

de junções são apresentados abaixo. Para isto, vamos supor uma operação de junção de R $\bowtie_{A=B}$ S onde A e B são atributos de domínio de R e S, respectivamente.

1. *Junção Nested-loop:* neste método, para cada registro de *R* são pesquisados todos os registros de *S* e testados se os dois registros correspondem a condição de junção.
2. *Junção Single-loop:* se uma chave existir em um dos dois atributos de junção (A e B de S), são recuperados todos os registros de *R* - um a um, e a chave é usada para acessar os registros de *S* que satisfaçam a condição de junção.
3. *Junção Sort-Merge:* caso os registros de *R* e *S* sejam armazenados de modo ordenado *(phisically sorted)* pelo valor dos atributos de A e B, a junção pode ser feita do modo mais eficiente possível. Os dois arquivos são pesquisados na ordem dos atributos de junção e recuperados os registros que satisfaçam ao critério. Caso os arquivos não sejam ordenados, a junção é feita pela classificação dos arquivos utilizando sort externo e, a partir daí, recuperados do mesmo modo.
4. *Junção Hash-Join:* os registros de *R* e *S* são, ambos, classificados para um mesmo *hash file* (arquivo classificado pelo valor do atributo), usando como *hash key (chave de classificação)* os atributos de junção *A* de *R* e *B* de *S*. Numa primeira operação, chamada de *partitioning phase*, uma leitura é realizada sobre o arquivo com menor quantidade de registros, supondo *R*, e classificando os registros no hash file. Uma segunda fase, chamada de *probing phase*, faz uma leitura no outro arquivo, supondo *S*, e combinando os registros com os registros de *R*, anteriormente gerados.

- **Implementando Operações de Projeção e Operações Set.** A operação de projeção consiste em selecionar atributos de uma tabela que satisfaçam a lista de atributos, gerando uma lista com a mesma

quantidade de tuplas da relação, contendo somente os atributos projetados. No caso de necessidade de eliminação de duplicatas - uso de uma operação SELECT DISTINCT, isto é feito por algoritmos de *sort-merge* ou *hash.*

O conjunto de operações *Set* são as operações de: união, intersecção, diferença e produto cartesiano. Na operação produto cartesiano, todas as tuplas de uma relação são combinadas com todas as tuplas de outra relação. As operações de união, interseção e diferença são aplicadas a relações com o mesmo número de atributos e domínio. O modo de implementar estas operações são variações da técnica de *sort-merge.* As duas relações são classificadas sobre os mesmos atributos e, após, pesquisadas de modo a produzir o resultado. A técnica de *hashing* também pode ser usada para implementar as operações set, de modo que uma tabela é particionada e a outra tabela é utilizada para recuperar a relação entre ambas.

- **Implementação de Operações de Agregação.** Os operadores de agregação - MIN, MAX, COUNT, AVERAGE E SUM, quando aplicados a toda a tabela, podem ser implementados pela pesquisa sobre toda a tabela ou pelo uso de um índice apropriado. No caso de ser utilizado o comando GROUP BY, o operador de agregação é aplicado separadamente a cada grupo de tuplas. A tabela é, num primeiro momento particionada em conjuntos de tuplas de mesmo valor do atributo de junção. A técnica mais usual é aplicar um algoritmo de sort ou hash nos atributos de junção de modo a separar os atributos em grupos. A partir disto, o algoritmo de agregação é aplicado a cada grupo.
- **Implementação de Junção Externa (Outer Join).** As operações de junção externa são feitas pela modificação dos algoritmos de junção ou pela execução combinada de operadores de álgebra relacional. Neste caso, o custo da junção externa é o resultado do custo das operações relacionais (seleção, projeção, união e junção).

- **Combinando Operações Utilizando Pipelining.** Uma consulta em SQL é traduzida em uma seqüência de operações da álgebra relacional. No caso de implementar uma operação a cada tempo, o resultado gera um overhead, pela criação de arquivos temporários e consumo de tempo. De modo a reduzir o overhead, o código de execução da consulta gera algoritmos que correspondam à combinação de operações da consulta. A técnica utilizada para combinar estas operações é chamada de *pipelining ou strem-based processing.*

3.3 Otimização de Consultas

O objetivo da otimização de consultas é a busca por um bom plano de execução para a consulta (RAMAKRISHNAN & GEHRKE, 2000). A otimização de consultas refere-se ao processo de produzir um plano de execução da consulta (QEP) que representa uma estratégia de execução para a consulta (ÖZSU, 1999). A otimização de consultas relacionais envolve duas etapas básicas (RAMAKRISHNAN & GEHRKE, 2000):

- Enumeração de planos alternativos, através da avaliação da expressão; tipicamente, um otimizador considera um subconjunto de todos os planos possíveis pelo fato de que as possibilidades podem ser muito grandes (RAMAKRISHNAN & GEHRKE, 2000).
- Uma estimativa do custo de cada plano enumerado e a escolha do plano de menor custo (RAMAKRISHNAN & GEHRKE, 2000).

Os otimizadores dos bancos de dados comerciais são complexas partes de software com muitos detalhes ocultos e protegidos e representam de 40 a 50 anos de esforços de desenvolvimento (RAMAKRISHNAN & GEHRKE, 2000).

As etapas básicas da otimização são (RAMAKRISHNAN & GEHRKE, 2000):

- **Plano de Avaliação de Consultas:** consiste em elaborar uma árvore de operações da álgebra relacional, com anotações adicionais a cada nó da árvore, indicando os métodos de acesso a serem utilizados para

cada relação e o método de implementação usado para cada operador relacional. Ex:

```
SELECT S.sname
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
AND R.bid = 100 AND S.rating > 5
```

A transformação desta consulta em álgebra relacional é:

$$\pi_{\text{sname}}(\sigma_{\text{bid} = 100 \wedge \text{Rating} > 5}(\text{Reserves} \bowtie_{\text{sid} = \text{sid}} \text{Sailors}))$$

Esta expressão forma a árvore abaixo:

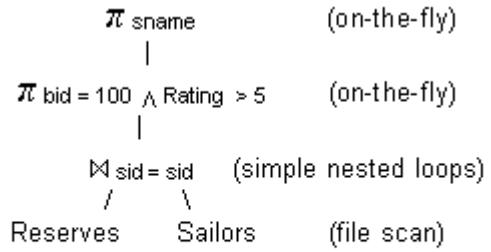


Figura 3.1. Consulta Relacional Expressa como uma Árvore
Fonte: (RAMAKRISHNAN & GEHRKE 2000)

- **Avaliação Pipelined:** quando uma consulta é composta por diversos operadores, o resultado de um operador é, algumas vezes, encaminhado para outro operador sem criar uma relação temporária que armazene o resultado intermediário. Isto economiza tempo de gravação do resultado intermediário e a sua leitura subseqüente. Caso a operação seja armazenada para uso do próximo operador, a operação é dita *materialized (materializada).* A avaliação pipelined tem um custo menor do que a materialização e deve ser utilizado sempre que a operação permitir. Quando a relação é uma operação unária (seleção, projeção, etc.), a mesma é pipelined em si mesma. Neste caso, o operador é chamado de *on-to-fly* (RAMAKRISHNAN & GEHRKE, 2000). Por exemplo, considere a junção: $(A \bowtie B) \bowtie C$, a figura 2.2 mostra a operação de junção.

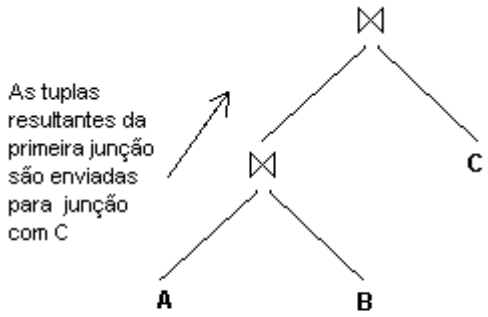


Figura 3.2. Árvore de Consulta Ilustrando Pipelining
Fonte: (RAMAKRISHNAN & GEHRKE 2000)

- **Interface Iterator para Operadores e Métodos de Acesso:** uma árvore de operadores relacionais é executada de acordo com uma ordem de operações. Cada operação tem uma ou mais entradas e saídas e, as tuplas resultantes da operação devem ser passadas entre os operadores, conforme a estrutura do plano. De modo a simplificar o código que é responsável pela coordenação do plano de execução o código que é responsável pela coordenação do plano de execução das operações, os operadores dos nós da árvore suportam uma interface uniforme, chamada de *uniform iterator interface.* O objetivo é esconder os detalhes internos de cada operação, sendo que, uma interface iterator inclui as funções *open, get_next e close.* A função open inicializa o estado do iterator, aloca buffers de entrada e saída e é usada para passar argumentos como condições de seleção que modificam o comportamento do operador. A função get_next é executada em cada nó de entrada e chama o código específico para cada operação processada. No término das operações, a função close é chamada de forma a encerrar a execução e desalocar os buffers e estados de execução.
- **Algoritmo System R:** O algoritmo system R realiza otimizações estáticas baseado na busca exaustiva do espaço de solução (ÖZSU, 1999). A entrada para o otimizador é uma árvore de álgebra relacional resultante da decomposição da consulta. A saída é um

plano de execução que implementa uma “ótima” árvore de álgebra relacional (ÖZSU, 1999). O design do otimizador do System R inclui:

- O uso de estatísticas sobre as instâncias do banco de dados, de modo a estimar o custo do plano de avaliação da consulta;
- A decisão de considerar somente planos com junções binárias nas relações internas que são a base da relação. Isto reduz o número de planos alternativos a serem considerados.
- A decisão de direcionar a otimização numa classe de consultas SQL, sem aninhar e manipular consultas aninhadas de um modo ad hoc (direto);
- A decisão de não executar a eliminação de duplicatas para projeções, exceto se a consulta possuir a cláusula DISTINCT;
- Um modelo de custo que considera os custos de CPU e I/O.

- **Algoritmo INGRES:** Ingres utiliza um algoritmo de otimização de consultas dinâmico que separa recursivamente a consulta em pequenos pedaços (ÖZSU, 1999). Combina duas fases de cálculo algébrico, a decomposição e a otimização. A consulta primeiro é decomposta em uma seqüência de consultas tendo uma única relação em comum. Após, cada consulta mono-relacionada é processada em um “one-variable query processor” (OVQP). O OVQP otimiza o acesso à relação selecionando, baseada em predicados, o melhor método de acesso à relação (index, sequential San, etc.) (ÖZSU, 1999).

3.3.1 System Catalog em Banco de Dados Relacionais

Conforme RAMAKRISHNAN & GEHRKE (2000), os catálogos *ou system catalog* armazenam as informações relativas à estrutura dos arquivos, índices, relações, descrição dos dados e visões mantidas pelo banco de dados. As informações são armazenadas em uma coleção de relações, mantidas pelo sistema. Estas relações são

chamadas de *system catalog, catalog ou de data dictionary* (RAMAKRISHNAN & GEHRKE, 2000). O catálogo do sistema é também chamado *de metadata*, pelo fato de conter informações sobre os dados e não dados propriamente ditos. As informações armazenadas no catálogo do sistema são utilizadas para a otimização das consultas. Um catálogo contém (RAMAKRISHNAN & GEHRKE, 2000):

- Para as relações:
  - O nome da relação, nome do arquivo, estrutura;
  - O nome e o tipo do atributo;
  - O nome do índice para cada índice da relação;
  - As restrições de integridade da relação.
- Para os índices:
  - O nome do índice e a sua estrutura;
  - Os atributos da chave de busca (search key);
- Para as visões:
  - O nome da visão e sua definição.

Além disto, estatísticas sobre as relações e índices são armazenadas e atualizadas periodicamente, sendo que são armazenadas, principalmente, as informações:

- Cardinalidade da relação;
- Tamanho da relação, em número de páginas;
- Cardinalidade dos índices;
- Tamanho dos índices, em número de páginas;
- O número de níveis dos índices;
- A escala do índice, em termos de valor mínimo e máximo das chaves existentes.

3.3.2 Otimização Heurística

A otimização heurística é uma técnica de otimização aplicada sobre a árvore de consulta ou o gráfico de estrutura da consulta, de modo a aumentar o desempenho da consulta. Uma das principais técnicas de otimização heurística é aplicar operações de seleção e projeção antes de aplicar operações de junção e outras operações binárias

(ELMASRI & NAVATHE, 2000). O objetivo é o de diminuir o número de registros envolvidos.

A otimização heurística é iniciada com a elaboração de uma *árvore de consulta inicial (initial query tree),* que corresponde a tradução da consulta SQL, sem qualquer otimização. Após, a otimização é feita pela inclusão de regras de equivalência relacional, e que irão transformar a consulta em uma *árvore de consulta final (final query tree),* otimizada.

Regras Gerais de Transformação para Operações da Álgebra Relacional. Estas regras são utilizadas para transformar as operações em álgebra relacional, mas não provêm otimização (ELMASRI & NAVATHE, 2000).

1. Cascata de σ: uma seleção conjuntiva pode ser dividida em uma seqüência de operações de seleção individuais:

   $$\sigma_{c1\ AND\ c2\ AND\ \ldots\ cn}(R) \equiv \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cn}(R))\ldots))$$

2. Comutatividade de σ: a operação de seleção é comutativa:

   $$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$$

3. Cascata de π: numa seqüência de operações de projeção, todas, menos a primeira, podem ser ignoradas:

   $$\pi_{List\ 1}(\pi_{List\ 2}(\ldots(\pi_{List\ n}(R))\ldots)) \equiv \pi_{List\ 1}(R)$$

4. Comutação de σ com π: se a condição de seleção *c* envolve somente os atributos A1, ... , An na lista de projeção, as operações podem ser comutadas:

   $$\pi_{A1, A2, \ldots An}(\sigma_{c}(R)) \equiv \sigma_{c}(\pi_{A1, A2, \ldots An}(R))$$

5. Comutação de $\bowtie$ e $\times$: a operação de junção é comutativa, bem como a operação de junção natural:

   $$R \bowtie_{c} S \equiv S \bowtie_{c} R$$

   $$R \times_{c} S \equiv S \times_{c} R$$

6. Comutação de σ com $\bowtie$ e $\times$: se todos os atributos da condição *c* envolvem somente atributos de uma das relações a serem juntadas, os dois operadores podem ser comutados:

$\sigma_c (R \bowtie S) \equiv (\sigma_c (R)) \bowtie S$

Do mesmo modo, se a condição *c* puder ser escrita como *c1* e *c2*, onde *c1* envolve somente atributos de R e a condição *c2* somente atributos de S, a operação é comutada do seguinte modo:

$\sigma_c (R \bowtie S) \equiv (\sigma_{c1} (R)) \bowtie (r_{c2} (S))$

Estas mesmas regras são aplicáveis à junção natural.

7. Comutação de π com $\bowtie$ e $\times$: supondo que a lista de projeção é $L = \{A_1, ... ,A_n, B_1, ...,B_m\}$ onde $A_1, ... ,A_n$ são atributos de R e $B_1, ... ,B_m$) são atributos de S. E a condição de junção *c* envolve somente atributos de L, os atributos podem ser comutados:

   $\pi_L (R \bowtie_c S) \equiv (\pi_{A1, ..., An} (R)) \bowtie_c (\pi_{B1, ... , Bm} (S))$

   Se a condição de junção *c* contiver atributos que não estão em L, estes devem ser adicionados à lista de projeção e uma operação de projeção final deve ser realizada. Sendo que os atributos $A_{n+1, ... , }A_{n+k}$ de R, e $B_{m+1, ... , }B_{m+p}$ de S envolvem uma operação de junção sob a condição *c*, mas não estão na lista de projeção L, a operação de comutação é:

   $\pi_L (R \bowtie_c S) \equiv ((\pi_{A1, ..., An, An+1, ..., An+k} (R)) \bowtie_c (\pi_{B1, ... , Bm, Bm+1, ..., Bm+p} (S)))$

   Para a junção natural $\times$, caso não exista a condição *c*, a primeira regra de transformação é aplicada, substituindo-se $\bowtie_c$ por $\times$.

8. Comutatividade das operação Set: as operações de união, interseção são comutativas.

9. Associatividade de $\bowtie$, $\times$, $\cup$ e $\cap$: sendo posto θ para qualquer um destes operadores:

   $(R \theta S) \theta T \equiv R \theta (S \theta T)$

10. Comutação de σ com operadores set: a operação de seleção é comutável com as operações de união, interseção e diferença se for dado θ para qualquer um dos operadores.

    $\sigma_c(R \theta S) \equiv (\sigma_c (R)) \theta (\sigma_c (S))$

11. A operação de π comuta com $\cup$:

$$\pi_L(R \cup S) \equiv (\pi_L(R)) \cup (\pi_L(S))$$

12. Convertendo uma seqüência σ e $\times$ em $\bowtie$: se a condição *c* da seleção executada após uma junção natural corresponde a uma condição de junção, convertendo a seqüência de seleção e junção natural por junção:

$$(\sigma_c(R \times S)) \equiv (R \bowtie_c S)$$

Esboço de um Algoritmo Heurístico de Otimização. Podemos esboçar os passos que um algoritmo utiliza utilizando algumas regras de otimização, de modo a transformar uma árvore inicial de consulta em uma árvore otimizada de execução mais eficiente (ELMASRI & NAVATHE, 2000). Este algoritmos utilizam as regras para transformar a árvore de consulta inicial em uma árvore otimizada.

1. Usando a regra 1, separar as operações de SELECT com operações conjuntivas, permitindo um grau maior de liberdade para mover as operações de seleção na árvore de consultas.
2. Usando as regras 2, 4, 6 e 10, prover comutatividade de seleção com outras operações, movendo cada operação Select para baixo da árvore de consulta.
3. Usando as regras 5 e 9, implementar comutatividade e associatividade de operações binárias, re-arranjando as folhas da árvore com base nos seguintes critérios: Primeiro, posicionando as folhas com operações de seleção mais restritivas (com menos tuplas, de menor tamanho ou de menor seletividade), de modo que sejam executadas primeiro. Segundo, assegurando que a ordem dos nós não cause uma operação Produto Cartesiano.
4. Usando a regra 12, combinar operações de Produto Cartesiano com a operação Select subseqüente na árvore em uma operação de Junção.
5. Usando as regras 3, 4, 7 e 11, realizar uma cascata de operações de projeção e comutar as projeções com outras operações, quebrando e movendo as listas de atributos de projeção para baixo da árvore e, se possível, criando novas operações de projeção.

6. Identificar sub-árvores que representam operações de grupo e que possam ser executadas por um algoritmo simples.

3.3.3 Otimização Utilizando Seletividade e Estimativa de Custo

Um otimizador não depende somente de regras heurísticas. Também devem ser estimados os custos e a comparação dos custos de execução das diferentes estratégias e escolher a que represente o menor custo estimado (ELMASRI & NAVATHE, 2000). Para isto, estimativas de custo precisas são requeridas, de modo que as diferentes estratégias sejam comparadas correta e realisticamente. Também deve ser limitado o número de estratégias a serem consideradas, de modo a não despender muito tempo na montagem das estratégias. Esta forma de otimização é mais adequada para *consultas compiladas (compiled queries),* onde a otimização é feita durante a compilação e a estratégia de execução é armazenada e executada diretamente pelo módulo de runtime. Nas *consultas interpretadas (interpreted queries),* uma otimização full-scale (completa) pode resultar em um tempo de resposta maior. Assim, uma otimização mais elaborada é indicada para consultas compiladas e uma otimização parcial é melhor para consultas interpretadas.

As otimizações baseadas no custo são chamadas de *cost-based query optimization (otimização baseada em estimativas de custo).* Utilizam as técnicas de otimização tradicionais e procuram no espaço de solução do problema, a que apresente o menor custo objetivo.

Componentes de Custo. Os componentes de custo para a execução de consultas são (ELMASRI & NAVATHE, 2000):

1. *Custo de acesso ao armazenamento secundário:* este custo é o gerado pela leitura e gravação de dados em dispositivos de armazenamento secundário. O custo de busca por registros em um arquivo depende da sua estrutura de acesso (ordem de armazenamento, hashing, chaves primária e secundária). Também, os fatores como o fato de os blocos serem alocados seqüencialmente ou de modo fragmentado no disco influenciam o custo.

2. *Custo de armazenamento:* este custo se deve ao armazenamento temporário dos dados, gerados pela estratégia de execução da consulta.
3. *Custo de processamento (computação):* é o custo da execução de operações na memória nos buffers de dados. Estas operações incluem busca, classificação e junção (merge) de registros e execução de processamento sobre valores de campos.
4. *Custo de utilização de memória:* é o custo relativo ao número de buffers de memória necessários para a execução da consulta.
5. *Custo de comunicação:* é o custo resultante do envio da consulta ou do seu resultado entre o site onde está a base de dados e o site ou terminal que originou a consulta.

Para grandes bancos de dados, a ênfase é dada na minimização dos custos de armazenamento secundário. Para pequenas bases de dados, a ênfase é dada em minimizar o custo de computação. Nos bancos de dados distribuídos, a ênfase é feita sobre o custo de comunicação.

Para cada plano de execução, deve ser estimado o seu custo (RAMAKRISHNAN & GEHRKE, 2000). Existem duas partes para estimar o custo de execução (RAMAKRISHNAN & GEHRKE, 2000):

1. Para cada nó da árvore, deve ser estimado o custo de execução da operação. Os custos são afetados significativamente quando é utilizada a avaliação pipelining ou quando são criadas relações temporárias que passam a saída de uma operação para outra.
2. Para cada nó da árvore, deve ser estimado o tamanho do resultado e como é ordenado (classificado). Este resultado é a entrada para a operação associada ao nó corrente, e o tamanho e a ordenação irão afetar a estimativa de tamanho, custo e ordem de classificação do nó associado (predecessor).

Histogramas: Um histograma é uma estrutura de dados mantida pelo DBMS que buscam aproximar a distribuição dos dados (RAMAKRISHNAN & GEHRKE, 2000). A figura 2.3 mostra duas distribuições - *D*. A primeira é uma distribuição não

uniforme de valores (de um atributo chamado *age*). A *freqüência* de um valor é o número de tuplas de cada valor de *age;* a distribuição representa a freqüência de cada valor de *age.* Neste exemplo, o menor valor de *age* é 0 e o maior é 14 e todos os registros de *age* são valores inteiros que variam de 0 a 14. A segunda distribuição busca aproximar *D*, assumindo que cada valor de *age* na faixa de 0 a 14 aparece de modo igual na coleção de tuplas. O número de freqüências é 45.

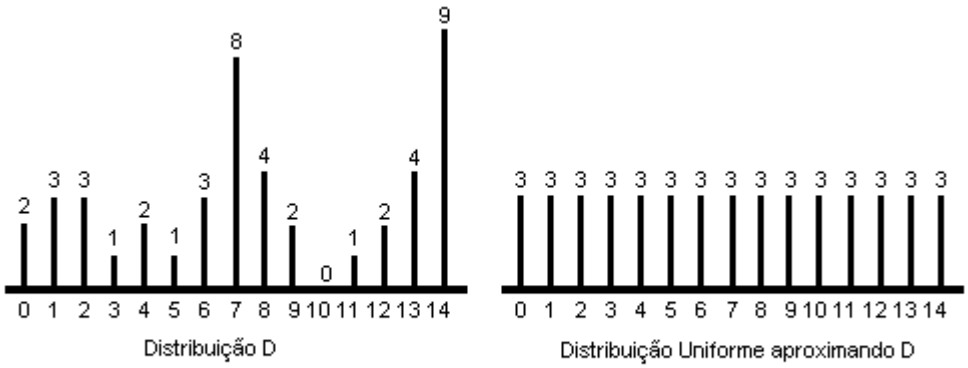


Figura 3.3. Distribuição Uniforme vs. Não Uniforme
Fonte: (RAMAKRISHNAN & GEHRKE 2000)

Considerando uma seleção *age > 13.* A partir da distribuição *D*, veremos que o resultado é 9. Utilizando a distribuição de aproximação uniforme, o resultado estimado é 1 + 15÷ 45 = 3 tuplas, estimativa a qual está errada (RAMAKRISHNAN & GEHRKE, 2000). Uma distribuição de dados mais acurada pode ser feita dividindo os valores de *age* em sub-valores chamados de *buckets*, e para cada buckets, contar o número de tuplas com valores de age. A figura 2.4 mostra dois diferentes histogramas chamados de *equiwidth* e *equidepth*.

A seleção *age > 13* utilizando o histograma equiwidth tem 5 como resultado estimado, pelo fato de que o alcance da seleção está dentro dos valores da faixa do bucket número 5. Uma vez que o bucket 5 representa um total de 15 tuplas, os valores da seleção correspondem a 1 ÷ 3 ∗ 15 = 5 tuplas. Utilizando o histograma equidepth, sobre a mesma seleção, o resultado obtido é 9. Este valor é resultante através do histograma, pelo bucket 5, que contém somente um valor de *age* que é 9.

Os sistemas de bancos de dados como DB2, Informix, Microsoft SQL Server, Oracle 8 e Sybase ASE usam histogramas de modo a estimar as características das

consultas como o tamanho do resultado e o custo (RAMAKRISHNAN & GEHRKE, 2000).

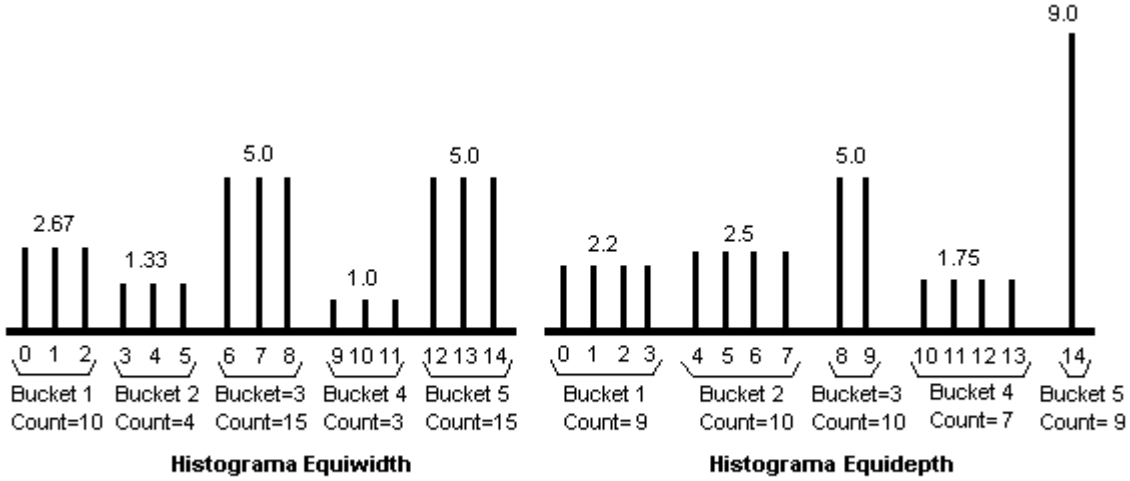


Figura 3.4. Histogramas Aproximando a Distribuição D
Fonte: (RAMAKRISHNAN & GEHRKE 2000)

3.3.4 Otimização Semântica

Uma outra forma de otimização, chamada de *semantic query optimization*, tem sido proposta. Esta técnica utiliza uma combinação das técnicas anteriores e utiliza as regras de consistência específicas no esquema do banco de dados, de modo a modificar a consulta tornando-a mais eficiente (RAMAKRISHNAN & GEHRKE, 2000). Considere a consulta:

```
SELECT E.LNAME, M.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS M
WHERE E.SUPERSSN=M.SSN AND E.SALARY>M.SALARY
```

Esta consulta retorna o nome dos empregados que ganham mais que os seus supervisores. Suponha a existência de uma regra de consistência que diga que nenhum empregado pode ganhar mais que o seu supervisor. O otimizador semântico checa a existência desta regra e não a executa, porque o resultado seria vazio. Um otimizador semântico pode economizar um tempo considerável, mas tem contra si o fato de que, a busca por regras de consistência que são aplicáveis a consulta pode, também, consumir muito tempo.

O propósito da otimização semântica de consultas é utilizar o conhecimento semântico (ex. restrições de integridade) para transformar a consulta em uma forma que pode ser respondida de modo mais eficiente que a consulta original (CHAKRAVARTHI et al., 1990). O uso de conhecimento semântico para otimizar consultas é chamado de *semantic query optimization* (CHAKRAVARTHI et al., 1990). Bancos de dados dedutivos - deductive databases, são providos da teoria da álgebra relacional, bem como de uma linguagem de consultas mais poderosa do que as linguagens de consulta relacionais. Um *deductive database system* é um sistema de banco de dados que inclui capacidades de definir (deductive) regras, que podem ser deduzidas ou inferir informações adicionais de fatos que estão armazenados no banco de dados (ELMASRI & NAVATHE, 2000). Provêm, também, uma representação uniforme para expressar os componentes do banco de dados, nomeação, fatos, regras de dedução e conhecimento semântico. Adicionalmente, provêm o formalismo necessário para o desenvolvimento da teoria semântica de otimização (CHAKRAVARTHI et al., 1990).

Nos bancos de dados dedutivos, a avaliação da consulta é dividida na fase de compilação e seguida por uma fase de modificação e análise. Os componentes básicos dos bancos de dados dedutivos são: o *intensional database (IDB)* que contém regras dedutivas e as relações definidas por regras dedutivas são chamadas de *intensional relations* (CHAKRAVARTHI et al., 1990); um conjunto de *integrity constraints (IC)* - que contém restrições de integridade que devem ser satisfeitas pelo banco de dados; e um *extensional database (EDB)*, que é formado por fatos expressos no bando de dados, por exemplo: é fato que uma tupla (a1,...,an) é membro de uma relação R, expressa pela literal atômica R(a1,...,an) ←. Estes fatos compreendem o EBD e a relação R é chamada de *extensional relation* (CHAKRAVARTHI et al., 1990).

A otimização é realizada aplicando restrições de integridade para otimizar a consulta sobre um banco de dados dedutivo, gerando diversas consultas semanticamente equivalentes. Estas equivalências são geradas nos estágios: *semantic compiler* e *semantic query transformer*. Durante a fase de *compilação semântica*, fragmentos das restrições de integridade - chamados resíduos, são computados e associados com regras dedutivas. O resultado é um conjunto de axiomas de restrição de integridade que serão filtrados e armazenados para uso futuro. Quando a consulta é passada para o sistema, a

transformação semântica utiliza estes resíduos para gerar um conjunto de consultas equivalentes que podem ser processadas de modo mais eficiente do que a consulta original. Esta proposta tem diversos benefícios (CHAKRAVARTHI et al., 1990):

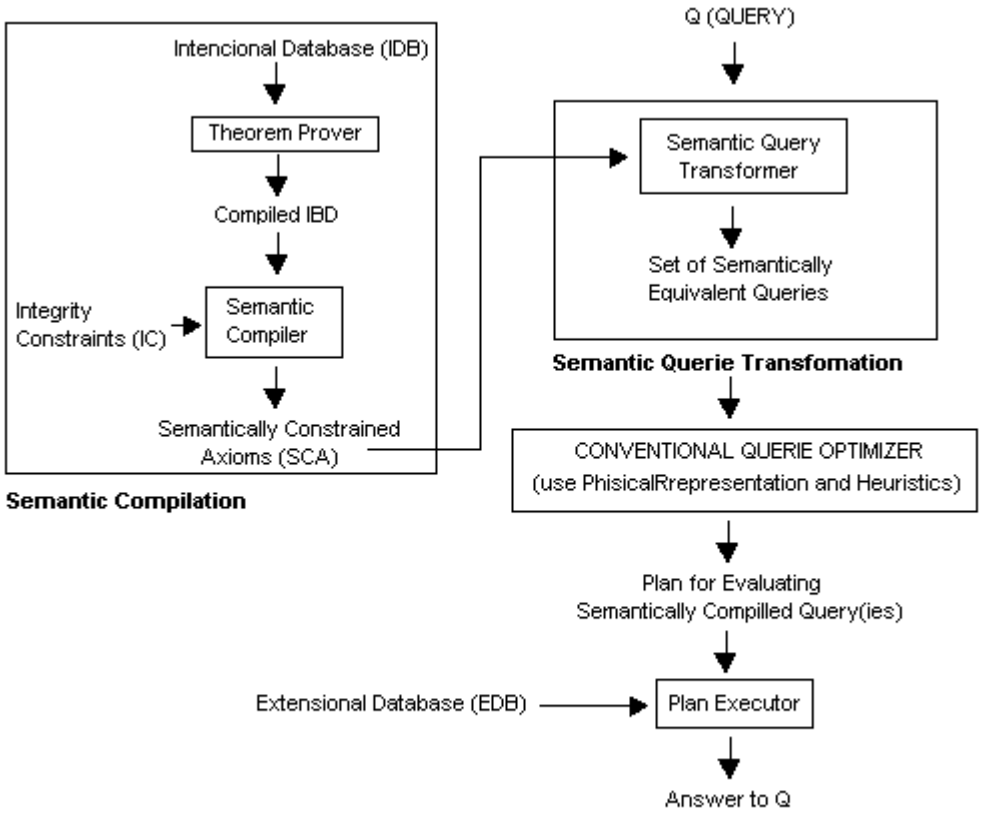


Figura 3.5. Processamento de Consultas utilizando Otimização Semântica
Fonte: (CHAKRAVARTHI et al. 1990)

A compilação semântica é feita utilizando componentes relativamente estáveis do banco de dados, é independente da consulta colocada no sistema e é computada a partir deste ponto, somente uma vez. Assumindo que os componentes não extensionais do banco de dados (IBC e IC) mantenham-se estáveis por um período relativo de tempo, o tempo dispendido na compilação semântica é amortizado sobre muitas consultas;

1. Quando a parte não extencional do banco de dados é alterada, novas regras dedutivas e restrições de integridade podem ser inseridas no sistema de forma incremental, prevenindo uma completa recompilação;

2. As transformações são realizadas em um alto nível de abstração, são heurísticas e mais apropriadas para este nível e são utilizadas no lugar das estimativas de custo;
3. Depois da fase de otimização semântica, as restrições de integridade são descartadas e não serão requeridas no processamento da consulta;
4. Este modelo pode ser estendido de modo a incorporar um mecanismo de feedback em um baixo nível de abstração, que pode ser utilizado para priorizar resíduos em termos dos seus efeitos efetivos.

3.4 Processamento de Consultas em Bancos de Dados Distribuídos

O processamento de consultas em bancos de dados distribuídos difere em alguns aspectos do processamento realizado em sistemas centralizados. As características do processamento de consultas distribuídas são, basicamente (ÖZSU, 1999):

- **Linguagem:** O maior trabalho no processamento da consulta é feito no contexto da linguagem de alto nível utilizada. Como visto anteriormente, existem diversos modos de realizar a consulta e otimiza-la. No contexto distribuído, a linguagem resultante da decomposição da consulta é uma forma que contem instruções da álgebra relacional e primitivas de comunicação. As operações da linguagem de saída são implementadas diretamente pelo sistema, assim, o processador de consultas deve realizar um mapeamento eficiente da linguagem de entrada para a linguagem de saída.
- **Tipos de Otimização.** A otimização consiste na busca do melhor ponto do espaço de soluções disponível para a consulta. No contexto dos bancos de dados distribuídos, a otimização deve preocupar-se em trocar operações de junção por semi-junções, de modo a minimizar a comunicação entre os sites.
- **Momento da Otimização.** A otimização pode ser realizada em diferentes tempos, relativos ao tempo de execução da consulta. Pode ser implementada de modo *estático (statically)*, antes da execução da consulta ou de modo *dinâmico (dinamically)*, durante a execução da consulta. A otimização

estática é feita durante a compilação da consulta. A otimização dinâmica é realizada em tempo de execução.

- **Estatísticas para Otimização de Consultas.** A otimização das consultas pode ser feita dinamicamente (em tempo de execução), utilizando estatísticas armazenadas no catálogo do banco de dados. Nos bancos de dados distribuídos, as estatísticas para otimização são relacionadas a fragmentos e devem incluir a cardinalidade do fragmento, tamanho e número de valores distintos dos atributos. Para minimizar a probabilidade de erro, estatísticas mais detalhadas devem ser feitas com o uso de histogramas.
- **Sites de Decisão.** Quando é utilizada a otimização estática (em tempo de compilação), um ou diversos sites podem participar da escolha da estratégia a ser usada para processar a consulta. O uso de um site centralizador para gerar a estratégia é o mais simples de ser implementado, mas requer o conhecimento de todo o banco de dados distribuído. A estratégia de distribuir o processo de decisão entre diversos sites é melhor e requer que cada site realize a consulta baseado nos dados locais. Uma estratégia híbrida pode envolver o uso de um site realizando as principais decisões e outros sites realizando as estratégias locais.
- **Aproveitamento da Topologia de Rede.** Nas redes WAN, o custo da consulta pode ser minimizado aplicando-se restrições no custo da comunicação. Isto é feito dividindo a consulta em dois problemas: a seleção da estratégia global, baseada no custo de comunicação; e a seleção da estratégia local, baseada nos algoritmos centralizados. Nas redes LAN, os custos de comunicação são comparáveis aos custos de i/o. Assim, a distribuição da consulta busca aumentar o paralelismo da execução, distribuindo a consulta entre os sites. Num ambiente cliente-servidor, o poder das estações clientes pode ser explorada de modo a otimizar a operação.
- **Aproveitamento de Fragmentos Replicados.** Uma consulta distribuída, expressa em uma relação global, pode ser mapeada em consultas a fragmentos de relações. Este processo é chamado de *location (localização)*

pelo fato de que a principal função é a localização dos dados envolvidos na consulta. Para obter disponibilidade, é comum o uso de replicação de fragmentos entre os sites.

- **Uso de Semi-junções:** A semi-junção é usada de modo a reduzir o tamanho da relação. Quando o principal custo a ser considerado é o de comunicação, as semi-junções são úteis em reduzir o tamanho dos dados trocados entre os sites.

Camadas do Processamento de Consultas Distribuídas

O processamento de consultas distribuídas envolve um site controlador e os sites locais. O esquema mostrado na figura 3.6 demonstra a distribuição do processamento da consulta entre os sites (ÖZSU, 1999).

Decomposição da Consulta. Esta etapa envolve a decomposição da consulta distribuída em consultas algébricas sobre relações globais. As informações necessárias são localizadas no esquema conceptual global. Os passos da decomposição são*: normalização*, que envolve a manipulação de quantificadores e qualificadores da consulta pela aplicação de prioridades de operadores lógicos; a segunda etapa é *a análise semântica* que detecta e rejeita consultas incorretas; a terceira etapa é a *simplificação*, que elimina predicados redundantes; a última etapa é a *reestruturação*, procurando transformar a consulta em sua melhor especificação algébrica.

Localização dos Dados. Esta fase envolve a localização dos dados distribuídos, determinando os fragmentos que serão envolvidos na consulta. Geralmente é feita em duas etapas: a primeira realiza o mapeamento da consulta, substituindo as relações distribuídas para uma consulta fragmentada. A segunda etapa envolve a simplificação e restruturação da consulta fragmentada, utilizando as mesmas regras utilizadas na decomposição.

Otimização Global. Esta etapa procura encontrar a melhor maneira de ordenar as operações da consulta fragmentada. Isto envolve minimizar o custo de computação, disco, comunicação, etc. O aspecto mais importante é a otimização de junções, transformando-as em semi-junções. A saída desta fase é uma consulta algébrica otimizada e com as operações de comunicação embutidas nos fragmentos.

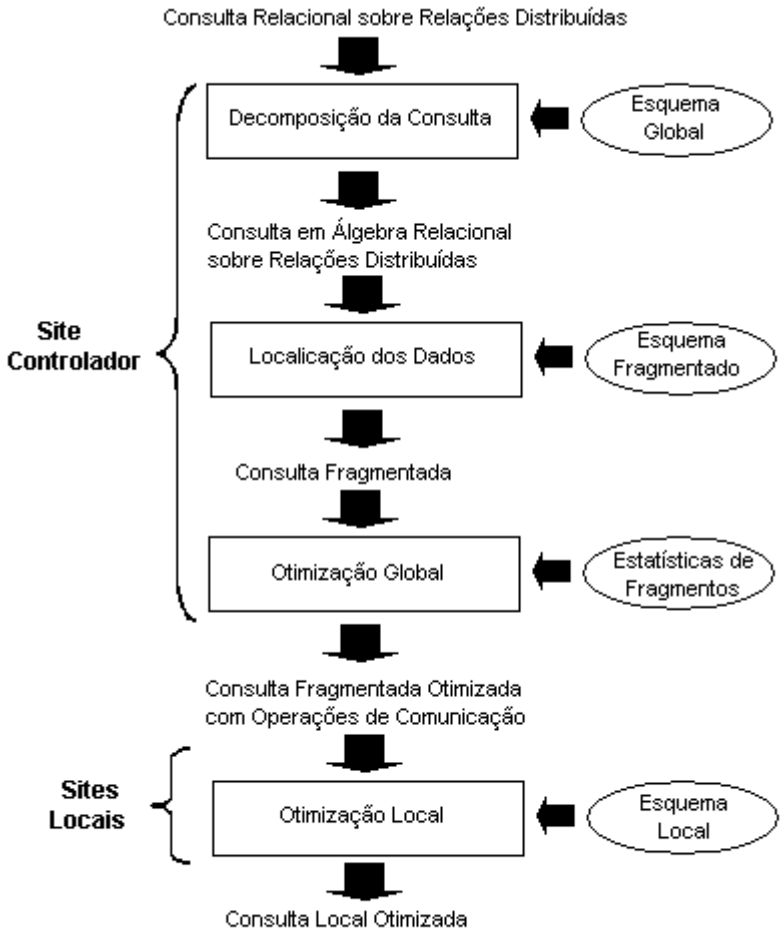


Figura 3.6. Camadas do Processamento de Consultas Distribuídas
Fonte: (ÖZSU 1999)

Otimização Local. Esta fase é feita em cada site, e a otimização da consulta é feita com base no esquema local. Nesta fase são usados os algoritmos dos sistemas centralizados.

3.4.1 Algoritmos de Otimização de Consultas Distribuídas

São apresentados três algoritmos básicos de otimização. Os mesmos são apresentados pelo fato de representarem diferentes classes de algoritmos e são considerados paradigmas (ÖZSU, 1999). As diferenças são:

- O momento da otimização é dinâmico no algoritmo distributed INGRES enquanto é estático nos outros;

- O objetivo funcional do SDD-1 e R* é minimizar o tempo total, enquanto no distributed INGRES é diminuir uma combinação de tempo de resposta e tempo total;
- Os fatores de otimização da função de custo são o tamanho da mensagem para SDD-1. O System R* emprega os custos de tempo de processamento local, tamanho da mensagem, número de mensagens, I/0 e CPU. Distributed INGRES considera ambos: tamanho da mensagem e tempo de processamento local (I/O + CPU);
- A topologia de rede é assumida como sendo uma rede wide ponto a ponto pelo SDD-1. Os algoritmos distributed INGRES e R* podem trabalhar com redes locais e wide;
- O uso de semi-junções na técnica de otimização é implementada pelo SDD-1. O distributed INGRES e R* realizam junções de modo similar aos algoritmos centralizados de suas contrapartes: INGRES e System R;
- Cada algoritmo assume informações estatísticas sobre os dados;
- INGRES lida com fragmentos.

Algoritmo Distributed INGRES: Este algoritmo é derivado do algoritmo centralizado INGRES (ÖZSU, 1999). Consiste em otimizar dinamicamente a estratégia de processamento da consulta. O objetivo é minimizar uma combinação de tempo de comunicação e tempo de resposta. A entrada do algoritmo de processamento da consulta é uma consulta expressa em cálculo relacional (na forma conjuntiva) e informações do esquema (o tipo da rede, localização e tamanho dos fragmentos). O algoritmo é executado por um site chamado de *master site*, onde a consulta foi iniciada. O algoritmo é caracterizado por limitar o espaço de solução do problema, onde uma decisão de otimização é feita sobre cada etapa, sem concernir sobre as conseqüências desta decisão sobre a otimização global. As consultas mono-relacionais (seleções e projeções) são inicialmente processadas localmente. As decisões relativas à transferência dos resultados para o site de resultado podem envolver a fragmentação destes fragmentos de modo a incrementar paralelismo. Este método é chamado de *fragment-and-replicate*. A seleção das relações

remanescentes e do número de sites de processamento é baseada numa função e na topologia da rede.

Algoritmo R*: O algoritmo de otimização distribuída R* é uma substancial extensão das técnicas desenvolvidas no otimizador System R* (ÖZSU, 1999). Opta por utilizar uma busca exaustiva de todas as alternativas de execução até escolher a de menor custo. A compilação da consulta é uma tarefa distribuída, coordenada por um *master site*, que é onde a consulta foi iniciada. O otimizador do site master toma todas as decisões intersites, como seleção do site de execução, fragmentação e método de transferência dos dados. Como no caso centralizado, o algoritmo deve selecionar uma ordem de junção e o caminho de acesso a cada fragmento. Estas decisões são baseadas em estatísticas e fórmulas que estimam o tamanho dos resultados intermediários e informações sobre o caminho de acesso. O algoritmo também deve selecionar os sites para junção dos resultados e o método de transferência de dados entre os sites. Para juntar as relações existem três alternativas: o site da primeira relação; o site da segunda relação ou um terceiro site. Dois métodos são utilizados para estas transferências:

- Ship-whole: A relação inteira é enviada para o site de junção e armazenada em uma relação temporária.
- Fetch-as-needed: A relação externa é escaneada seqüencialmente, e para cada tupla o valor de junção é enviado para o site da relação interna, que seleciona as tuplas internas, combinando o valor e enviando a tupla selecionada para o site da relação externa.

Algoritmo SDD-1: O algoritmo SDD-1 é derivado de um método chamado “hill-climbing” que tem a distinção de ser o primeiro algoritmo para processamento de consultas distribuídas (ÖZSU, 1999). Neste algoritmo, refinamentos sobre uma solução inicial praticável são recursivamente computados até que não possam mais serem realizados aperfeiçoamentos. O algoritmo não utiliza semi-junções e não assume replicação e fragmentação dos dados. O custo de transferência dos resultados para o site final é ignorado. A

entrada para o algoritmo é o grafo de consulta, localização das relações e estatísticas da relação. Após completar-se um processamento local inicial, uma solução inicial praticável é selecionada como uma tarefa de execução global que inclui todos as comunicações entre os sites. É obtida computando o custo de todas as estratégias de execução que transferem todas as relações requeridas para um único site, e então é escolhida a de menor custo.

CONCLUSÃO

O processamento de consultas em banco de dados é feito por um módulo do sistema chamado de *processador de consultas*. O processador de consultas busca realizar a consulta de maneira mais eficiente e a menor custo - de processamento e de tempo. As etapas desenvolvidas são a análise, otimização e execução. O processo de análise envolve examinar a consulta e gerar uma representação na forma de uma *árvore de consulta*. A partir desta árvore de consulta é gerado uma estratégia de execução que implica na escolha de um modo de realizar a consulta, entre os diversos possíveis. Após esta etapa, o processo de otimização busca encontrar uma estratégia razoável e eficiente de implementar a consulta. Os componentes do otimizador são: *search space* - que busca encontrar o melhor algoritmo para a execução e produzir uma árvore de operadores que definem a ordem em que a consulta é realizada. O segundo componente é o *search strategy* - que busca a obtenção de um plano completo, utilizando dois métodos: o método dinâmico e o método randômico. O terceiro componente é o *distributed cost model*, onde são implementadas funções que procuram predizer o custo das operações.

Os algoritmos básicos utilizados para implementar as consultas são: External sorting - que baseia-se na realização de sort-merge; e o algoritmo que implementa operações de seleção.

Os catálogos ou system catalogs dos bancos de dados armazenam as informações relativas a estrutura dos arquivos, índices, relações, descrições dos dados e visões mantidas pelo banco de dados. Estas informações são utilizadas no processo de otimização da consulta, como modo de estimar o custo da mesma. Os métodos de

otimização são: *otimização heurística* - aplicada sobre a árvore de consulta utilizando operações de seleção de projeção e, após operações de junção. A *otimização utilizando seletividade e estimativa de custo,* procura estimar os custos de realização das operações. A *otimização semântica,* combina as técnicas anteriores e utiliza as regras de consistência do banco de dados, modificando a consulta de modo otimiza-la.

O processamento de consultas nos bancos de dados distribuídos difere do realizado em sistemas centralizados. As suas características são: a *linguagem* - onde a consulta contém instruções da álgebra relacional e primitivas de comunicação; o *tipo de otimização,* de modo a considerar os custos de comunicação entre os sites; o *momento de otimização* - que refere-se a quando a otimização será realizada, sendo que os modos são o estático, realizado quando da construção, ou dinâmico, realizado em tempo de execução; As *estatísticas para otimização,* que usa informações armazenadas no catálogo do banco de dados, e que, nos bancos de dados distribuídos, devem estar relacionadas com os fragmentos das relações; o *site de decisão,* que é o site ou os sites responsáveis por gerar a estratégia de execução; o *aproveitamento da topologia de rede,* onde o custo de comunicação está vinculado ao meio de comunicação utilizado; ao *aproveitamento de fragmentos replicados,* que envolve a localização dos dados ou parte dos dados nos sites; e o *uso de semi-junções,* que busca reduzir o tamanho dos dados transferidos entre os sites.

O processamento das consultas envolve um site controlador e os sites locais. O site controlador é o responsável pela execução de toda a consulta, delegando aos sites locais a execução de partes da mesma. O processamento da consulta envolve as etapas de: decomposição da consulta, localização dos dados, otimização global e otimização local.

Os algoritmos básicos de otimização de consultas distribuídas são: o algoritmo Ingres, o algoritmo R e o algoritmo SDD-1. O algoritmo Ingres utiliza a consulta expressa em cálculo relacional e informações do esquema do banco de dados. O algoritmo R, busca otimizar a consulta pela busca exaustiva das alternativas de execução. O algoritmo SDD-1 aplica refinamentos sobre uma consulta inicial, utilizando informações relativas a localização e estatísticas da relação.

4 Apresentação dos Sistemas de Gerência de Banco de Dados Analisados

Introdução

Este capítulo investiga os SGBD DB2, Oracle, Ingres e Sybase, analisando a arquitetura de implementação de distribuição e replicação de dados cada sistema. Estuda o modo como realizam o processamento de consultas e como implementam um ambiente de distribuição e replicação de dados, pesquisando os componentes, tipos de distribuição e propriedades de cada SGBD. É mostrado ambiente sobre o qual a análise das propriedades de transparência de localização, nomeação e replicação são avaliadas e a configuração do sistema operacional Red Hat Linux 6.2 implementada.

4.1 SGBD IBM DB2

Conforme IBM (2001), o sistema de gerência de banco de dados DB2 é organizado em uma hierarquia de objetos compostos por: instâncias, bases de dados, nodegroups, table spaces e tabelas. O *database manager* - gerenciador do banco de dados, ou instância, é um sistema que gerência os dados, controlando o que pode ser feito sobre estes dados e gerênciando os recursos do sistema necessários. É um ambiente completo, contendo todas as partições do banco de dados. Possui sua própria base de dados que não pode ser acessada por outras instâncias, e todas as partições acessam o mesmo diretório do sistema. Um *nodegroup,* é um conjunto de uma ou mais partições onde as table spaces são armazenadas. Um banco de dados é organizado em *table spaces*. As definições das table spaces e os atributos são armazenados no catálogo do banco de dados. Para cada table space é alocado um *container,* que é o armazenamento físico da mesma. Uma *table* são dados lógicos organizados em linhas e colunas.

O SGBD DB2 pode formar um ambiente paralelo e multi-nó. Um *nó ou node* é uma partição do banco de dados. Uma *partição* é parte do banco de dados e contém seus próprios dados, índices, arquivos de configuração e arquivos de log (IBM, 2001). Um banco de dados *single-partition,* contém uma única partição e todos os dados são armazenados nesta partição. Um banco de dados *partitioned* é um banco de dados que tem duas ou mais partições. As tabelas podem ser localizadas em uma ou mais destas partições. A figura 4.1 demonstra os objetos do SGBD DB2.

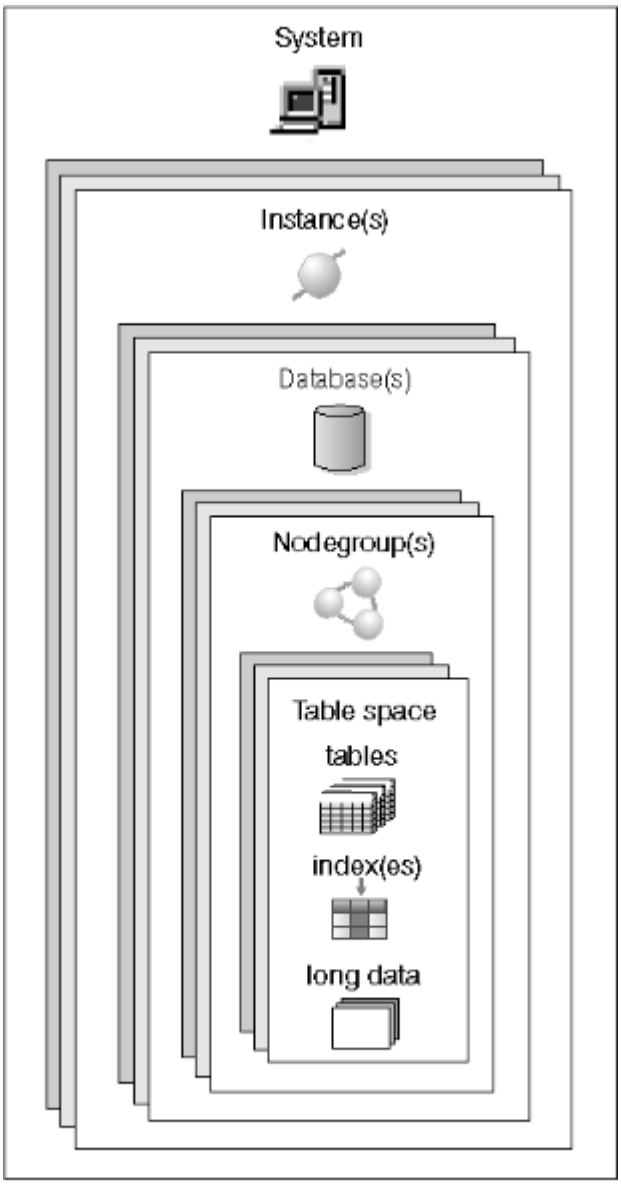


Figura 4.1. Objetos do Banco de Dados
Fonte: (IBM, 2001)

O sistema de processamento e otimização de consultas implementado no SGBD DB2 realiza pre-compilação de consultas SQL e otimização cost-based (MOHAN, 1993). O otimizador utiliza um sofisticado e detalhado modelo de custo para estimar o custo de execução da consulta. O sistema suporta os métodos nested loops, merge scan join e hybrid join. De modo a minimizar o erro na estimava de custo da consulta, o sistema realiza buscas em valores de indíces que ocorrem freqüentemente prevenindo e corrigindo erros na estimativa de tamanho de tabelas e fazendo com que isto resulte numa uniforme distribuição de valores dos campos.

O sistema utiliza técnicas de processamento paralelo de consultas para explorar a disponibilidade do sistema na execução das consultas, sendo:

- *Inter-query parallelism:* esta técnica permite a realização de diversas consultas concorrentes, de modo a aumentar a saída dos dados.
- *Intra-query parallelism:* refere-se a realização de partes de uma consulta ao mesmo tempo. Para isto, utilizado duas técnicas:
  - *Intra-partition parallelism:* referindo-se a habilidade de pegar uma consulta - como busca, junção, sort, merge, e gerar múltiplas tarefas idênticas em partes diferentes dos dados. Estas tarefas são executadas em diferentes CPUs em um sistema multi processado (SMP Server) e em uma partição do banco de dados.
  - *Inter-partition parallelism:* esta técnica separa um consulta em diversas partes e executadas em múltiplas partições do banco de dados. É feita em paralelo sobre as diversas partições.

Estas consultas são realizadas sem sincronização, o envolvimento do coordenador se refere a iniciar as tarefas e coletar os resultados.

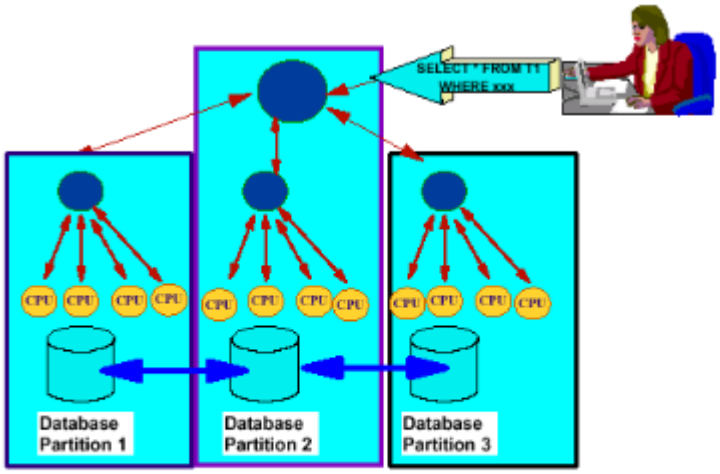


Figura 4.2. Processamento Paralelo de Consultas
Fonte: (IBM, 1999)

A topologia de particionamento das bases de dados emprega uma arquitetura flexível de armazenamento dos dados, sendo que cada partição é um subconjunto do banco de dados, contendo seus próprios dados, indíces, log de transações e arquivos de configuração (IBM, 1999). O contrôle das partições é feito via “node groups” configurados no sistema. O node group é um objeto que define o número de partições e o espaço de distribuição das tabelas *(tablespaces)* para todas as tabelas existentes. Um mapa de partições é criado e contém as

definições de cada nó do banco de dados, sendo possível a criação de 4.096 valores. Cada valor corresponde a *chave de particionamento (partitioning key)* da tabela, definindo em qual partição o dado será armazenado. Cada vez que uma tupla é adicionada ao banco de dados, a chave de particionamento é checada, resultado em um valor entre 0 e 4.095, que corresponde a partição. A figura abaixo mostra como um dado é adicionado a uma partição. A figura mostra um node group que tem três partições definidas. A definição de: em qual partição o dado será adicionado é feita pelo chave de particionamento e indexação do valor (hashing), resultando em um número que determina a partição. No exemplo, a indexação (hashing) resulta no número 8, que corresponde a partição 3. Isto é armazenado no catálogo do sistema e utilizado para otimizar as consultas.

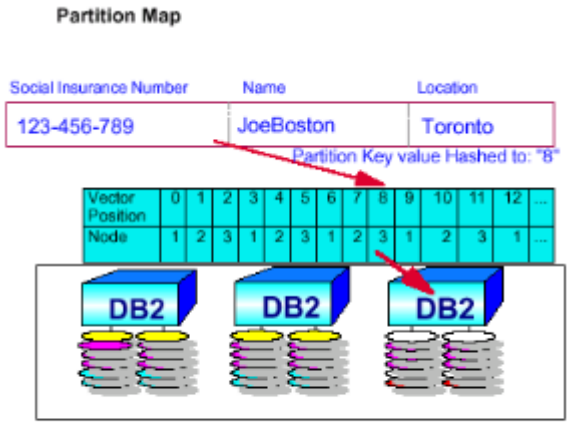


Figura 4.3. Particionamento de Dados
Fonte: (IBM, 1999)

4.1.1 Replicação e Distribuição de Dados

Replicação é o processo de manter um definido conjunto de dados em mais de uma localização. Envolve copiar as trocas de uma localização (a origem) para outra (o destino), e sincronizar os dados entre os locais. A origem e o destino pedem ser servidores lógicos (como um banco de dados DB2 ou um subsistema DB2 para OS/390 ou um grupo de compartilhamento de dados) que estão na mesma máquina ou em diferentes máquinas em uma rede distribuída (IBM, 2000). Os processo de replicação de dados no SGBD DB2 consiste de três componentes principais:

a. A interface de administração: utilizada para criar as tabelas de contrôle que armazenam os critérios de replicação;

b. O mecanismo de captura de alterações: que captura as alterações no banco de dados de origem e armazena as trocas em tabelas temporárias;
c. O programa de aplicação: que lê as tabelas e aplica as trocas no bancos de dados de destino.

As combinações possíveis envolvem a distribuição e a consolidação dos dados, sendo que as configurações ou aplicações típicas são (IBM, 2000):

- **Data distribution:** Neste ambiente, a cópia primária dos dados reside em um servidor de origem e as mudanças nos dados são replicadas para um ou mais servidores de destino. A figura 4.4 mostra este ambiente.
- **Data Consolidation:** Nesta configuração, um servidor central é utilizado como repositório de dados de diversas fontes de dados. A figura 4.5 demonstra esta configuração.
- **Update anywhere:** Nesta configuração, tanto as réplicas como a fonte de replicação são cópias de leitura/gravação.
- **Occasionally connected:** Nesta configuração, existe flexibilidade de conexão e transferência dos dados de e para uma origem primária dos dados. Esta configuração permite a conexão dos usuários a uma fonte de dados de modo a sincronizar suas bases de dados locais. A figura 4.7 demonstra esta configuração.

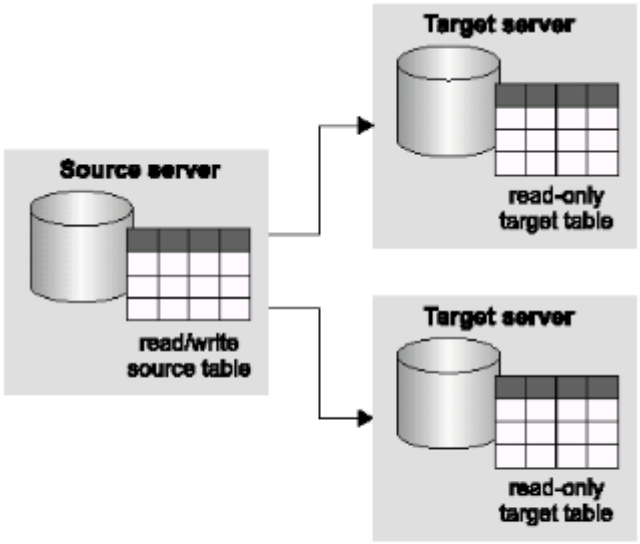


Figura 4.4. Data Distribution
Fonte: (IBM, 2000)

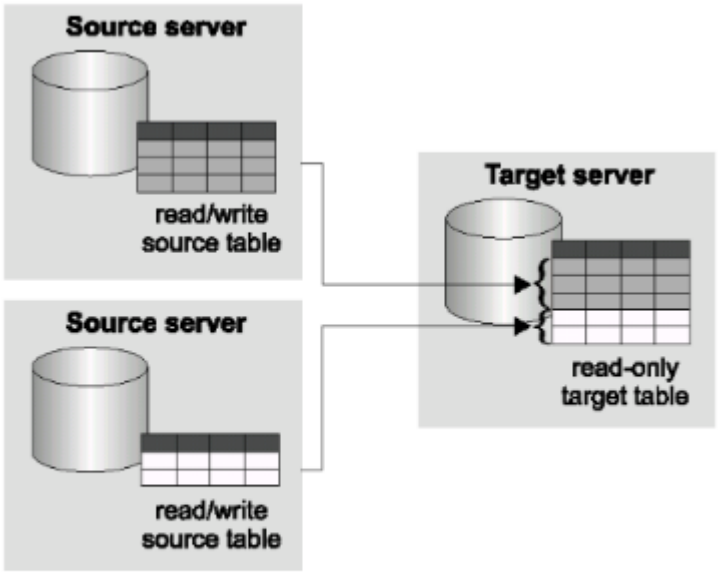


Figura 4.5. Data Consolitation
Fonte: (IBM, 2000)

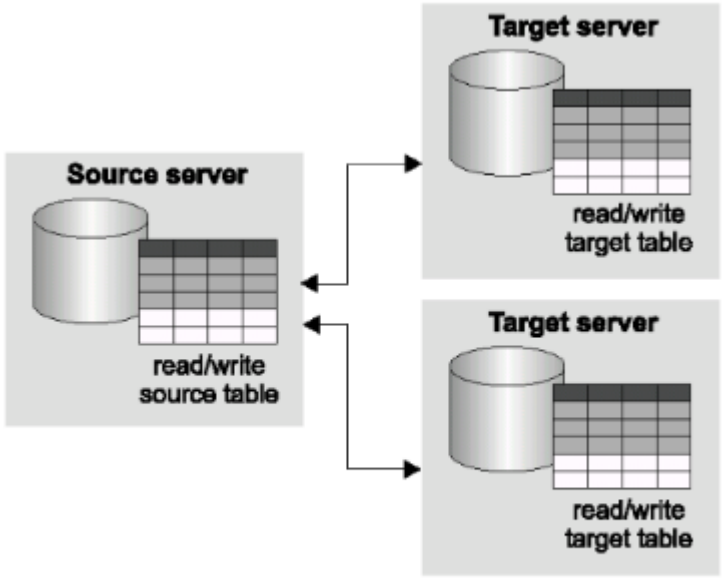


Figura 4.6. Update Anywhere
Fonte: (IBM, 2000)

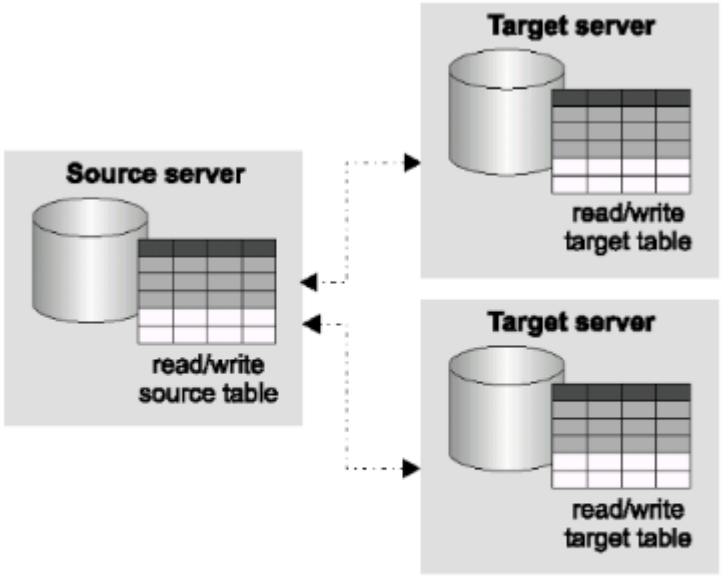


Figura 4.7. Occasionally Connected
Fonte: (IBM, 2000)

O processo de replicação utiliza control tables - tabelas de contrôle, que gerenciam as requisições de replicação, logical servers - servidores lógicos , que contém os componentes replicados, a interface de administração, os mecanismos de captura e de aplicação (IBM, 2000). Estes componentes são explicados abaixo:

- **Control Tables:** os componentes da replicação utilizam as control tables para comunicarem-se entre si e gerênciarem as tarefas de replicação. O mecanismo de captura e troca (change-capture mechanism) utiliza as seguintes tabelas de contrôle: register table, unit-of-work table, pruning control table, critical section table, warm start table, tuning parameter table e change data tables (IBM, 2000). O programa de aplicação (apply program) utiliza as seguintes tabelas de contrôle: apply trail table, register table, subscription set table, subscription statements table, subscription events table, subscription-targets-member table, subscription columns table, unit-of-work table, and change data tables(IBM, 2000).
- **Logical Servers:** Os componentes replicados residem em um servidor lógico. Um logical server refere-se a um banco de dados. Os tipos de logical servers são (IBM, 2000):
  - Source server: que contém o mecanismo de captura de alterações, as tabelas de origem que serão replicadas e as tabelas de contrôle para o programa de captura;

- Target server: que contém as tabelas de destino;
- Control server: que contém as tabelas de contrôle do programa de aplicação.

O programa de aplicação pode residir em somente um dos servidores da rede. Cada programa de aplicação é associado com um servidor de contrôle que é especificado quando o programa é inicializado (IBM, 2000).

- **Interfaces de administração:** as interfaces de administração são utilizadas para criar as tabelas de contrôle que armazenam o critério de replicação. As interfaces disponíveis são: DB2 Control Center e DataJoiner Replication Administration (DJRA) (IBM, 2000). O DB2 Control Center é uma ferramenta de administração utilizada para administrar o ambiente do SGBD DB2.
- **Mecanismo de captura e aplicação:** a solução para replicação de dados do DB2 utiliza este mecanismo para capturar dados, sendo: o *capture program* para a captura de tabelas, e *capture triggers* para tabelas em banco de dados não IBM, com exceção dos SGBD Teradata, Microsoft Access e Microsoft Jet (IBM, 2000).
  - Programa de Captura: quando uma tabela de origem é uma tabela do SGBD DB2, o programa captura as modificações feitas no banco de dados de origem. Utiliza o log do banco de dados para capturar as trocas e armazena-as em tabelas. O programa de captura é executado no servidor de origem (source server).
  - Programa de Aplicação: o programa de aplicação lê os dados diretamente das tabelas ou visões de origem para aplica-las nas tabelas de destino. Se as tabelas de origem são tabelas não IBM, o programa lê os dados a partir de um apelido. O programa de aplicação é executado, normalmente, no servidor de destino, mas pode ser executado em qualquer servidor da rede. Cada programa de aplicação é associado a um servidor de contrôle, que contém as tabelas de contrôle que tem as definições para os conjuntos de subscrições (subscription sets).
- **Método de Comunicação entre os componentes de replicação:** os componentes de replicação são independentes. Os programas de captura, aplicação e as triggers de captura atualizam as tabelas de contrôle para indicar o progresso da replicação e coordenar o processo de atualizações (IBM, 2000). Os componentes comunicam-se de modo diferente, dependendo se o servidor de origem é um servidor DB2 ou não. Para a replicação entre servidores DB2, o programa de

captura verifica as modificações através da leitura dos arquivos de *log* ou *journal*. A partir disto, armazena as trocas em tabelas chamadas change data (CD) tables. Em um determinado tempo, o programa de aplicação copia os dados para o banco de dados de destino.

Os conceitos utilizados no sistema de replicação implementado no DB2 são (IBM, 2000):

- **Replication Sources:** uma fonte de replicação é uma tabela do usuário ou uma visão, a partir da qual os dados serão replicados. Antes de iniciar a replicação dos dados, deve ser definido uma replication source para descrever a informação que será utilizada pelo mecanismo de troca-captura. As definições envolvem especificar as colunas a serem replicadas e decidir se as atualizações serão tratadas como operações update ou operações insert e delete. Também é necessário especificar:
    - **Colunas after-image e before-image:** Uma coluna after-image contém o valor dos dados da coluna na tabela origem depois dos dados serem atualizados. Uma coluna before-image contém os valores da coluna de origem antes dos dados serem atualizados. É necessário escolher se o mecanismo de captura utilizará somente as colunas after-image ou ambas. As colunas before-image são úteis para aplicações que requeiram realizar auditoria ou operações de rollback;
    - **Cópia full-refresh e differential-refresh:** O programa de aplicação copia os dados da origem para o destino utilizando cópia full-refresh ou differential-refresh. A cópia *full-refresh* implica nas operações de: deletar todas as linhas da tabela de destino, ler todas as linhas da tabela de origem e copiar as linhas para a tabela de destino. A cópia *differential-refresh* copia somente os dados modificados para a tabela de destino;
    - **O nível de detecção de conflitos se for utilizado:** A detecção de conflito refere-se ao uso de replicação onde as cópias replicadas podem ser modificadas, neste caso, uma mesma linha pode ser modificada na origem e no destino durante o mesmo ciclo de replicação. O nível de detecção pode ser implementado de duas formas: Standard - onde o mecanismo de aplicação pesquisa por conflitos em linhas já capturadas; e o Enhanced -

onde o programa de aplicação bloqueia todas as tabelas de destino, assegurando que todas as modificações foram checadas.

- **Substription sets**: Um substription-set contém os atributos da replicação. Deve ser criado definindo-se (IBM, 2000):
    - Nome;
    - Servidores de origem e destino;
    - Qualificador para o programa de aplicação:
    - Quando iniciar a replicação, a freqüência e o uso de replicação por tempo, por evento ou ambos;
    - O uso de bloco de dados (data blocking), se utilizar grandes volumes de trocas.

  Os subscription sets devem ter um *subscription set member*, para cada tabela ou visão de destino. Quando criado, devem ser definidos os atributos (IBM, 2000):
    - A tabela ou visão de origem e destino;
    - A estrutura da tabela ou visão de destino;
    - As colunas que serão replicadas;
    - As linhas que serão replicadas.

  Os subscription sets asseguram que todos os subscription-set members são tratados igualmente durante a replicação.
- **Apply qualifier:** o qualificador de aplicação associa um programa de aplicação com uma ou mais subscription sets. É utilizado para identificar os registros do servidor de contrôle que definem o trabalho de leitura de uma instância do programa de aplicação. Por exemplo: suponha-se que os dados em de uma tabela A são replicados utilizando cópia full-refresh para uma tabela destino A, e os dados da tabela de origem B são replicados utilizando copia differential-refresh. A definição de dois subscription sets - um para cada tabela, e o uso de apply qualifiers separados permite que duas instâncias do programa de aplicação realizem a cópia dos dados em tempos diferentes (IBM, 2000).
- **Data manipulation:** A replicação pode ser realizada sobre um subconjunto da tabela de origem. O uso de visões permite reestruturar os dados ou pode ser utilizado mecanismos de junção e uniões.
    - **Subconjuntos de tabelas:** O uso de particionamento da tabela permite replicar somente um subconjunto das colunas da tabela de origem. Para

replicar linhas deve ser utilizado uma cláusula where quando da definição do subscription-set member.

- **Visões:** O uso de visões pode ser utilizado de modo a reestruturar as cópias dos dados.
- **Junção e União:** Pode ser utilizado junções e uniões em tabelas destino, a partir de tabelas de origem. Os tipos de junções que podem ser utilizadas são (IBM, 2000): Junção simples sobre uma ou mais origens de replicação, com uma possível combinação com outras tabelas ou visões que não são por si só origens de replicação; Junção simples sobre tabelas CCD que são definidas como origem de replicação. O uso de junções e uniões permite manipular os dados de modo a: Junções de tabelas em um único destino; União de tabelas em um único destino e; união de tabelas para múltiplos destinos.

- **Target tables:** Quando um substription-set member é definido, deve ser especificado o tipo da tabela de destino que será utilizado. Os tipos disponíveis são (IBM, 2000):
  - User copy tables: onde as cópias são apenas para leitura, sem a adição de colunas de contrôle;
  - Point-in-time tables: são tabelas somente para leitura, com uma coluna timestamp adicionada. Quando as trocas são replicadas, o valor da coluna indica em que tempo a modificação foi realizada;
  - Aggregate tables: são tabelas somente para leitura que usam funções SQL sobre as colunas (como SUM e AVG), de modo a computar resumos sobre o conteúdo de todas as tabelas ou sobre as modificações realizadas. Os tipos permitidos são: *base aggregate tables* - que contém um resumo do conteúdo de uma tabela de origem, como, por exemplo, a média de consumidores que uma empresa tem a cada mês. Neste caso, a replicação é realizada com o uso deste tipo de replicação; e *change aggregate tables* - que lidam com as trocas feitas nas tabelas de contrôle e não com o conteúdo das tabelas de origem. É utilizada para conhecer as modificações

realizadas em um período de tempo, como, por exemplo, saber quantos consumidores foram adicionados (insert) ou excluídos (delete);

- Consistent-change-data (CCD) tables: O conteúdo destas tabelas são dados tornados permanentes (commited) e contém um indicador que demonstra se a tabela de destino foi modificado com o uso de operações insert, delete ou update e pode conter os dados antigos e os novos. Cada tipo de tabela CCD tem um uso diferente e pode ser utilizada para coletar e manipular os dados do seguinte modo (IBM, 2000):
    - Preparar modificações para locais remotos: se existem muitos destinos, o processo de replicação pode ser feito de modo que, ao invés de replicar para todos os destinos, replicar da origem para uma tabela CCD, e, a partir disto, replicar desta tabela para os destinos. Este processo pode diminuir o tempo de transferência sobre a rede de comunicação;
    - Replicar somente a modificação de uma linha para um destino. As tabelas CCD podem reduzir a tráfego sobre a rede e prevenir atualizações feitas repetidamente sobre as mesmas linhas, em um curto período de tempo;
    - Coletar informações para auditoria;
    - Atuarem como origem de troca de dados para mecanismos de replicação que não o programa de captura.
- Réplica ou row-replica tables: São tabelas onde as aplicações podem realizar atualizações. As modificações feitas sobre as réplicas são replicadas para a tabela de origem e direcionadas para as outras réplicas;
- User tables: Não é permitido especificar uma tabela de usuário como destino. Entretanto, na replicação update-anywhere, uma tabela de usuário é, automaticamente, o destino para as réplicas ou row-replicas associadas. A tabela do usuário é *parent of the replica* e as suas cópias são *dependent replicas.* A parent of the replica recebe atualizações de uma dependent réplica e replica as trocas para outras dependent réplicas.

- **Agendando atualizações:** A aplicação das modificações pode ser realizada de dois modos: *Synchronous replication* - onde as modificação são enviadas continuamente e as trocas são tornadas permanentes (committed) no banco de

dados de origem somente depois de replicadas para o banco de dados de destino. É chamada de replicação real-time. *Asynchronous replication* - onde as modificações são enviadas em estágios. Quando uma modificação é realizada sobre um dado, ele é armazenado temporariamente, por um intervalo de tempo pré-determinado, e enviado para o destino após este tempo. Os modos de definir as atualizações são (IBM, 2000):

- Intervalo de tempo: Onde é selecionado uma data e hora para iniciar o processo de replicação;
- Evento: Neste método, um evento é definido no subscription set e é definido um tempo para o evento ser processado;
- On-demand: Este método utiliza o comando ANSSAT. Este comando inicia os programas de aplicação e captura e tem a propriedade de terminar por eles mesmos, ao fim de um ciclo de replicação.

4.2 SGBD ORACLE

O banco de dados Oracle é uma coleção de dados que é tratada como uma unidade. Um banco de dados tem uma estrutura lógica e uma estrutura física. A estrutura lógica inclui tablespaces, schema objects, data blocks, extents e segments. Um banco de dados é dividido em *tablespaces* que são unidades lógicas de armazenamento do banco de dados. Cada tablespace tem um ou mais *datafiles* que armazenam fisicamente os dados (ORACLE, 1999c). A figura 4.8 apresenta esta divisão.

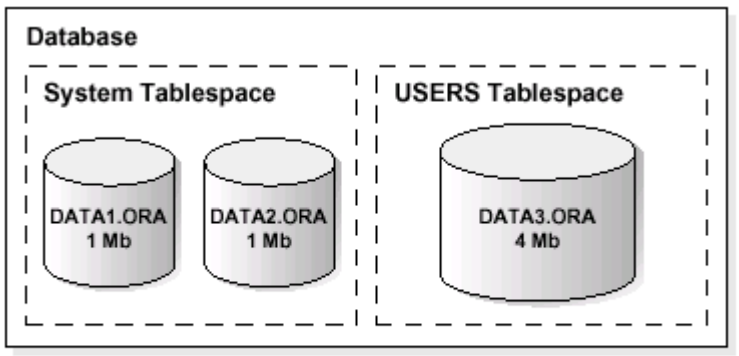


Figura 4.8. Database e Tablespace
Fonte: (ORACLE, 1999c)

Um *schema* é uma coleção de objetos do banco de dados. Os *schema objects* são as estruturas lógicas que referenciam diretamente os dados do banco de dados. Incluem

estruturas como tabelas, visões, seqüências, stored procedures, sinônimos, clusters e database links (ORACLE, 1999c). Não existe uma relação entre um tablespaces e um schema, objetos de um mesmo schema podem estar em diferentes tablespaces. Uma *tabela (table)* é a unidade básica de armazenamento. Uma *visão (view)* é uma representação dos dados de uma ou mais tabelas. Uma *visão materializada (materialized view)* provê um acesso indireto aos dados armazenado os resultados de uma consulta em um separado objeto do esquema. Uma *seqüência (sequence)* é uma listagem serial dos números únicos para as colunas numéricas de uma tabela, uma seqüência simplifica a programação de aplicações pelo fato de gerar automaticamente valores numéricos únicos para a linha de uma tabela ou múltiplas tabelas. Uma *unidade do programa (program unit)* são stored procedures, funções, pacotes, gatilhos e blocos anônimos do banco de dados. Um *sinônimo (synonym)* é um apelido para uma tabela, visão, seqüência ou program unit. Os sinônimos são utilizados para: esconder o nome real e o usuário do objeto, prover acesso público ao objeto, prover transparência de localização e para simplificar os comandos sql. Um *índice* é uma estrutura opcional associada as tabelas, que incrementam o desempenho na busca dos dados (ORACLE, 1999c).

4.2.1 Replicação e Distribuição de Dados

Um sistema de banco de dados distribuído permite as aplicações acessarem dados de bases de dados locais e remotas (ORACLE, 1999a). O sistema de banco de dados distribuído suportado pelo SGBD Oracle prevê dois tipos de distribuição: Um *sistema distribuído homogêneo,* onde todos os bancos de dados são Oracle; e um *sistema distribuído heterogêneo,* onde, ao menos um dos bancos de dados não é um banco de dados Oracle (ORACLE, 1999a). Os termos *sistema de banco de dados distribuídos (distributed database system)* e *banco de dados replicado (database replication)* são relacionados, mas distintos. Num banco de dados distribuído, o sistema gerência uma única cópia de todos dados e sustenta objetos do banco de dados. Tipicamente, aplicações de banco de dados distribuídos utilizam transações distribuídas para acessar dados locais e remotos e modificar o banco de dados global em tempo real (ORACLE, 1999a). O termo replicação refere-se a operação de manter e copiar objetos do banco de dados em múltiplos bancos de dados pertencentes a um sistema distribuído (ORACLE, 1999a).

O conceito central em sistemas de banco de dados distribuídos é a criação de *database links* (ORACLE, 1999a). O database links são conexões entre dois servidores de bancos de

dados que permitem aos clientes o acesso como se fossem um banco de dados lógico (ORACLE, 1999a). Um database link é um ponteiro que define uma comunicação de uma via (one-way) entre dois SGBD. É definido como uma entrada no dicionário de dados. A conexão entre dois bancos de dados, A e B, por exemplo, é feita do seguinte modo: um cliente conectado ao banco de dados A, para ter acesso ao banco de dados B, precisa utilizar um link armazenado do banco de dados A. Esta informação é válida somente para os usuário do banco de dados A. Caso os usuários do banco de dados B queiram ter acesso ao banco de dados A, devem ter um link armazenado no banco de dados B. A figura 4.9 demonstra o uso de database links.

Os database links são referenciados no sistema através de nomes globais (Global Database Names). Um global database name é um nome que identifica unicamente um banco de dados. É formado por um prefixo que identifica o domínio da rede, especificado quando da criação do banco de dados, e pelo o nome do banco de dados. No exemplo abaixo, o nome global MFTG.DIVISON3.ACME_TOOLS.COM é formado pelo nome do banco de dados MFTG e pelo domínio DIVISON3.ACME_TOOLS.COM. A figura 4.10 demonstra o uso de nomes globais.

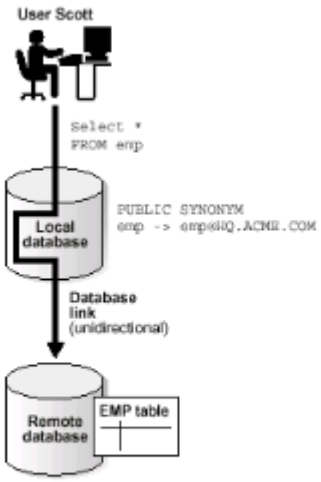


Figura 4.9. Database Link
Fonte: (Oracle, 1999a)

Replicação é o processo de copiar e manter objetos do banco de dados em múltiplas bases de dados, constituindo um sistema de banco de dados distribuído (ORACLE, 1999b). Os componentes de um sistema de replicação são (ORACLE, 1999b):

- **Replication Objects, Groups, and Sites**: Os objetos que podem ser replicados são chamados de *replication objects*. Os *replication objects* são objetos do banco

de dados que existem em múltiplos servidores em um sistema distribuído (ORACLE, 1999b) . Num ambiente replicado, as trocas feitas em um objeto replicado em um site são aplicadas nas cópias em todos os outros sites. Os objetos que podem ser replicados são:

- Tabelas;
- Índices;
- Visões;
- Pacotes (Package) e Corpos de Pacotes (Package Bodies) ;
- Procedimentos (Procedures) e Funções;
- Gatilhos;
- Seqüências (Sequences);
- Sinônimos.

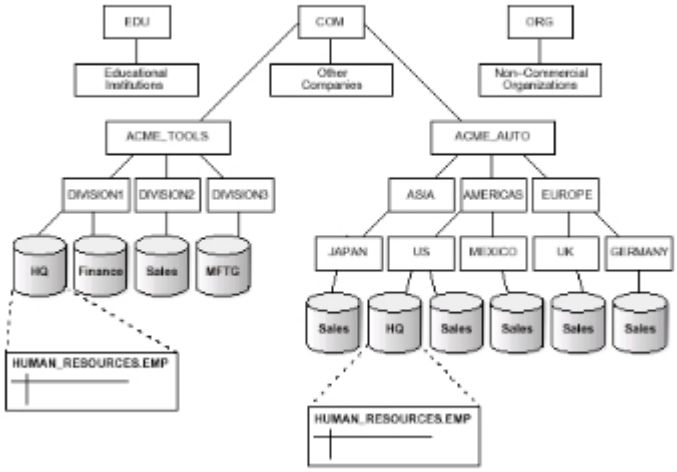


Figura 4.10. Nomes Globais

- **Replication Groups:** O SGBD Oracle manipula os objetos replicados utilizando os *replication groups.* Um *replication group* é um conjunto de objetos replicados, logicamente inter-relacionados (ORACLE, 1999b). Os objetos de um grupo replicado são gerênciados em conjunto (ORACLE, 1999b).
- **Replication Site:** Conforme ORACLE (1999b), um ambiente replicado suporta dois tipos básicos de locais: *master site* e *snapshot site*. Um site pode ser um *master site* e *um snapshot site* ao mesmo tempo. Um *replication group* pode existir em múltiplos *replication sites*. As diferenças entre um master site e um snapshot site são:

1. Um *replication group* em um *master site* é referenciado como *master group.* Um *replication group* em um *snapshot site* é referenciado como *snapshot group.* Cada *master group* tem um *master definition site* que é o site que centraliza o contrôle do grupo replicado e dos objetos replicados do grupo;
2. Um *master site* contém um cópia completa de todos os objetos em um *replication group* enquanto que um *snapshot site* pode conter todos ou um subconjunto das tabelas de um *master group;*
3. Todos os *master sites* em um ambiente de replicação multimaster comunicam-se diretamente um com o outro de modo a propagar continuamente as trocas nos dados e nos esquemas do *replication group.* Um *snapshot site* contém uma imagem (snapshot) dos dados da tabela em um dado momento. Tipicamente, um *snapshot site* é atualizado periódicamente de modo a sincroniza-lo com seu *master site*. Os *snapshots* podem ser organizados em *grupos de atualização*, de modo que a atualização seja feita sobre um ou mais *snapshot group* e que eles sejam atualizados ao mesmo tempo, assegurando que os dados de todos os *snapshots* do *grupo de atualização* correspondam a uma mesma consistência transacional em um dado ponto de tempo.

- **Tipos de Ambiente de Replicação:** O SGBD Oracle suporta os seguintes ambientes de replicação (ORACLE, 1999b):
    - **Replicação Multimaster:** A replicação multimaster, também chamada de replicação ponto-a-ponto ou replicação n-way, permite que múltiplos sites, atuando como pontos iguais, gerênciem grupos de objetos replicados e que cada site do ambiente de replicação seja um *master site*. As aplicações podem atualizar qualquer tabela replicada em qualquer site. A figura 4.10 ilustra este ambiente.
    - **Replicação Snapshot:** Um snapshot contém uma cópia completa ou parcial de uma tabela master de origem em um único ponto de tempo. Um snapshot pode ser somente de leitura ou atualizável. A replicação snapshot tem os seguintes benefícios:
        1. Permite acesso local, implementando tempo de resposta e disponibilidade;

2. Possibilita consultas *offload* no master site, pelo fato de que os usuários processam a consulta no snapshot local;
3. Implementa segurança dos dados permitindo a replicação de subconjuntos dos dados da tabela principal.

- Replicação Snapshot Read Only**:**: Este tipo de replicação provê acesso somente de leitura dos dados da tabela originada de um master site. As aplicações executam consultas nos snapshots read-only, não necessitando de acesso a rede de comunicação e contribuindo para a disponibilidade da rede. A figura 4.11 mostra este modo de replicação.

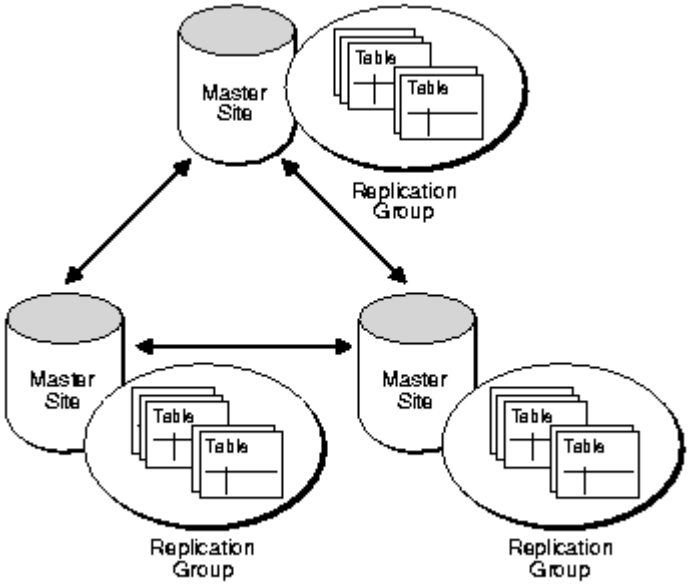


Figura 4.11. Replicação Multimaster

Entretanto, para realizarem atualizações nos dados, necessitam acessar a tabela do site master. Os benefícios da replicação read-only são:

1. Elimina a ocorrência de conflitos, uma vez que as tabelas não são atualizáveis;
2. Pode suportar replicações complexas, com as que contenham um conjunto de operações ou que utilizem cláusulas CONNECT BY.

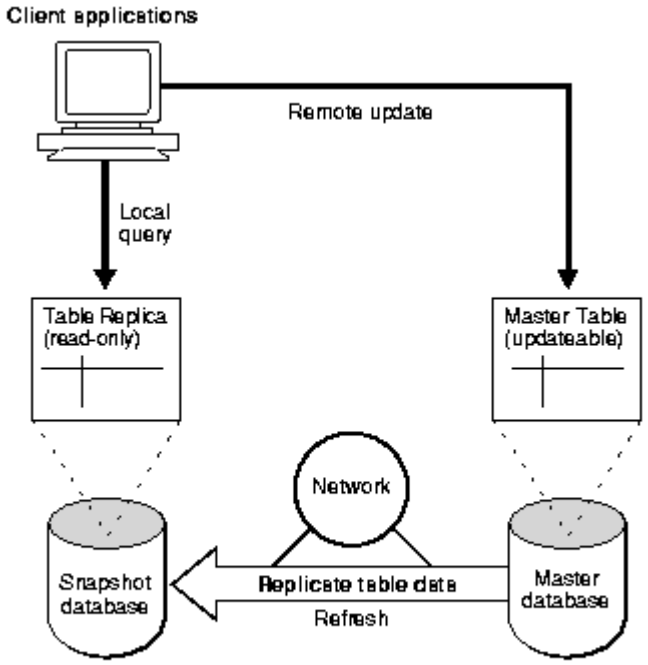


Figura 4.12. Replicação Snapshot Read Only
Fonte: (ORACLE, 1999b)

- Replicação Snapshot Updateable**:** Uma replicação snapshot updateable permite que os usuários realizem inserções, atualização e exclusões das linhas da tabela master de origem, realizando estas operações na tabela snapshot. Um snapshot updateable pode conter subconjuntos da tabela principal. A figura 4.13 demonstra esta replicação. Um snapshot updateable deve fazer parte de um *snapshot group* que á baseado em um *master group* de um *master site*. Tem as propriedades:
    1. São sempre baseados em uma única tabela;
    2. Podem implementar atualizações incrementais (rápidas);
    3. As trocas feitas em um *snapshot updateable* são propagadas para a tabela master remota. Se necessário, as atualizações da tabela master podem ser refletidas em todos os outros sites master;
    4. A atualização pode ser parte de um *grupo de atualização* da mesma forma que as atualizações dos snapshots read-only.

  A replicação snapshot updateable tem os benefícios:

1. Permite aos usuários consultar e atualizar os dados replicados localmente, mesmo quando desconectados do site master;
2. Requer menos recursos que a replicação multimaster. Por exemplo, um snapshot pode residir em um banco de dados Oracle 8i Lite, que requer menos recursos de memória e disco que os requeridos por um Oracle 8i server.

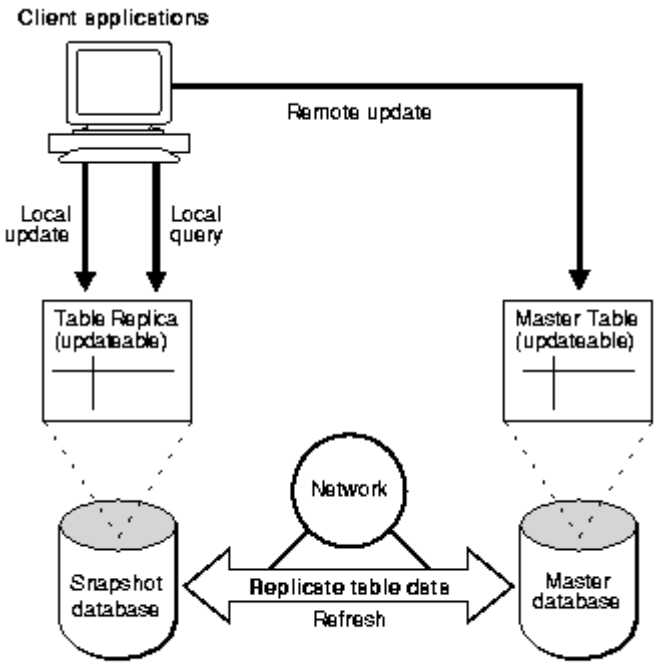


Figura 4.13. Replicação Snapshot Updateable
Fonte: (ORACLE, 1999b)

- **Configurações Híbridas:** As replicações multimaster e snapshot podem ser combinadas em um ambiente híbrido. Uma configuração híbrida pode ter um número qualquer de sites master e múltiplos sites snapshot para cada site master. A figura 4.14 demonstra este tipo de replicação.

A diferença entre as replicações são (ORACLE, 1999b):

- A replicação multimaster deve conter todos os dados da tabela que será replicada. A replicação snapshot pode conter subconjuntos da tabela principal;
- A replicação multimaster permite a réplica das modificações para cada transação no momento em que as mesmas ocorram. As atualizações snapshot são configuradas, a propagação das mudanças de múltiplas transações são mais eficientes, sendo uma operação batch, mas que ocorrem em intervalos de tempo menores;

- Os sites master detectam e resolvem conflitos que ocorrem a partir de modificações realizadas em múltiplas cópias dos mesmos dados.

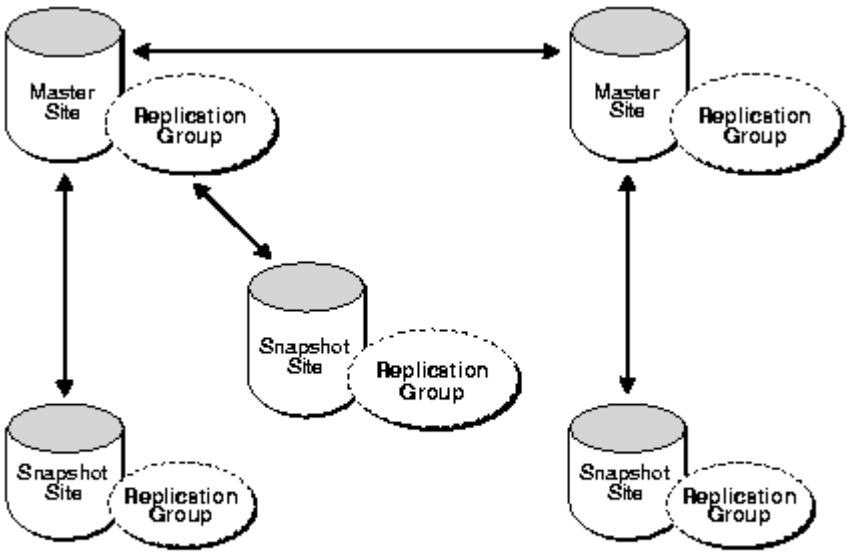


Figura 4.14. Replicação Híbrida
Fonte: (ORACLE, 1999b)

A administração do ambiente de replicação é feita com o uso de ferramentas e informações armazenadas no banco de dados, sendo (ORACLE, 1999b):

- Oracle Replication Manager: é uma ferramenta de administração do ambiente de replicação;
- Replication Management API: constituído por um conjunto de interfaces de programa de aplicação (Aplication Program Interface), que permitem encapsular procedimentos e funções para configurar um ambiente de replicação;
- Replication Catalog: cada master site e snapshot site tem um catálogo de replicação. É composto por um conjunto de tabelas de dicionário de dados e visões que mantém informações sobre os objetos replicados e grupos de replicação de cada site;
- Distributed Schema Management: num ambiente replicado, todos os comandos DDL devem ser executados utilizando o replication manager ou o pacote DBMS_REPCAT. Quando é utilizada uma destas interfaces, todos os comandos são replicados para os sites participantes do ambiente de replicação.

4.3 SGBD INGRES

A arquitetura do SGBD Ingres tem os seguintes componentes (CA, 1999a):

- O Ingres tools: constituído pelos aplicativos que os usuários utilizam para acessar e utilizar o banco de dados. Estas ferramentas incluem os aplicativos Query-By-Forms, Report-Writer e o Terminal Monitor. Quando o usuário acessa um banco de dados através destas ferramentas, um processo de comunicação é criado e iniciado com o servidor do banco de dados. As informações necessárias para estabelecer a conexão são providas pelo componente General Communication Facility;
- O Ingres Database Management System Server: A arquitetura do SGBD Ingres permite diversos usuários acessarem o banco de dados através de conexões a um ou mais processos do banco de dados. O servidor do banco de dados é um processo multi-threaded daemon que executa operações assíncronas de entrada/saída. O servidor do banco de dados consiste dos componentes (CA, 1998a):
    - Abstract Data Type Facility (ADF): Responsável por toda a manipulação que envolva tipos de dados. Este componente manipula números de ponto flutuante, strings de caracteres, inteiros e todas as operações de conversão e comparação entre os dados;
    - Data Manipulation Facility (DMF): Gerência a interface para armazenamento em disco, bem como as estruturas de armazenamento dos dados (hash, heap, isam, etc.). Utiliza o sistema de logging e locking para controlar o processamento das transações;
    - Optimizer Facility (OPF): Seleciona o melhor plano para otimização de consultas. É responsável pela conversão das árvores de consulta em um plano de execução de consultas;
    - Parser Facility (PSF): Converte as consultas do modo texto para o formato interno, utilizando informações do catálogo e informações sobre a estrutura das tabelas e chaves;

- Query Execution Facility (QEF): Executa os planos de execução de consultas e os utilitários do banco de dados. Manipula as consultas, transações e cursores;
- Querty Storage Facility (QSF): Provê os recursos para compartilhamento de memória utilizando locais temporários ou permanentes para armazenar árvores de consulta e planos de consultas;
- Relation Description Facility (RDF): É o ponto central de informações sobre as tabelas. É utilizado pelos componentes Star, PSF e OPF;
- System Control Facility (SCF): É o coordenador central que gerência as sessões baseadas nas requisições dos clientes. Coordena as ações sobre as operações envolvidas no processamento da consulta, incluindo monitoramento de therads e switching. É responsável pelo acesso aos recursos compartilhados como semáforos do sistema operacional e memória.

- O Logging and Locking System: Coordena o bloqueio, recuperação e geração de logs do banco de dados (journaling). Composto pelos componente:
  - Lock manager: Responsável por controlar o acesso concorrente ao banco dados;
  - Logging facility: Implementa um log de transações “write-ahead” para gerênciar as transações. Assegura que os registros de log são gravados de modo a tornar possível os processos de recuperação e de arquivamento. Este processo é compartilhado por todos os servidores do ambiente de banco de dados;
  - Recovery process: Responsável pela manipulação dos processos de recuperação em caso de ocorrência de falhas. Esta facilidade é compartilhada por todos os servidores do ambiente;
  - Archiver process: Cada instalação tem um único processo responsável por copiar a história de operações realizadas na base de dados (journaled databases) do arquivo de log de transações para os arquivos de jornal (journaled files). Os arquivos de jornal contém um subconjunto de informações do arquivo de log de transações associadas com um banco de dados específico;

- Transaction log file e optional dual log file: cada instalação contém um arquivo de transações e um arquivo opcional segundo arquivo de log. Contém informações sobre todas as transações em aberto e é utilizado para restaurar o banco de dados na ocorrência de falhas;
- Outros arquivos de log: utilizados por processos do banco de dados, de modo a isolar os erros relacionados a estes, como, por exemplo, erros de comunicação;

- A General Communications Facility: Gerência a comunicação entre todos os componentes do banco de dados. É composto por três elementos:
  - Name server: Existe um único nome de servidor para cada instalação. Provê informações que permitem aos usuários conectarem-se aos servidores do banco de dados locais e remotos;
  - Communications server: É um processo daemon que provê os elementos de comunicação sobre a rede. Uma instalação pode ter diversos processos de comunicação;
  - General Communications Architecture (GCA): É responsável por manter os processos de conexão de uma mesma instalação local. É utilizada pelas ferramentas do Ingres, pelo servidor do banco de dados, pelos servidores Star e pelas bibliotecas associadas ao uso de SQL embutido.

4.3.1 Replicação e Distribuição de Dados

O SGBD Ingres utiliza um gerênciador de dados distribuídos chamado de STAR que adiciona a capacidade de gerênciar um sistema de banco de dados distribuído (CA, 2000). O método de acesso aos dados é obtido a partir do recebimento de requisições, pelo servidor Star, que controla o envio da mesma para o banco de dados local. A figura 4.15 mostra a arquitetura do sistema.

Caso um dos banco de dados local não seja Ingres, o componente Enterprise Access é utilizado para realizar a comunicação com o Star. Para criar um banco de dados distribuído é necessário:

a. Registro**:** O registro descreve a conexão lógica (link) entre um objeto do banco de dados distribuído e um objeto do banco de dados local;

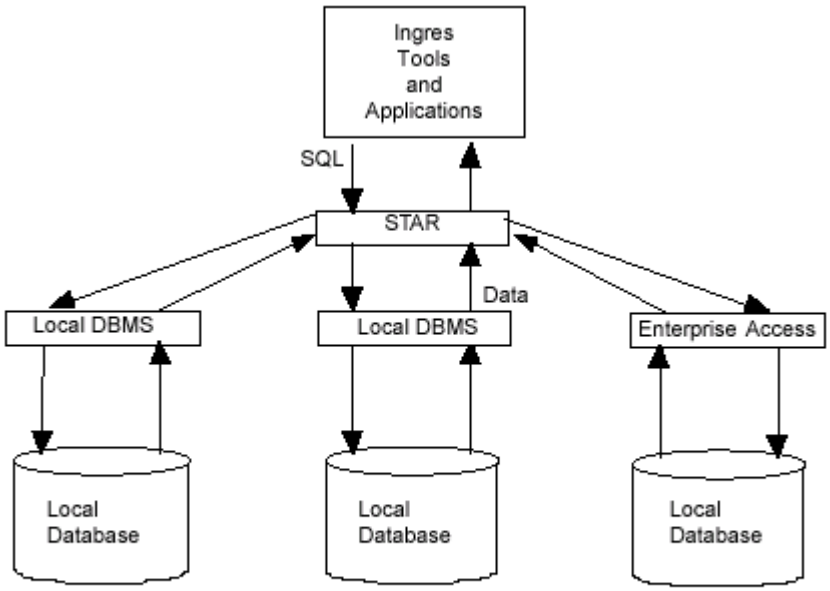


Figura 4.15. Arquitetura do Star
Fonte: (CA, 2000)

b. Propagação: A propagação tem o objetivo de registrar as tabelas que fazem parte do ambiente de distribuição. A propagação pode ser feita de três modos (CA, 2000):
    1) Registrando tabelas existentes no banco de dados local, utilizando o comando register as link;
    2) Criando tabelas com registro automático no banco de dados distribuído, utilizando o comando create table;
    3) Criando tabelas locais e registrando-as no banco de dados distribuído.
c. Catálogos: Quando um banco de dados distribuído é criado são gerados catálogos para coordenar o acesso aos dados. Os catálogos são um conjunto de informações relativas ao banco de dados distribuído;
d. Tipos de Objeto: Os objetos que podem ser distribuídos são: tabelas, visões, procedimentos e índices.

Um banco de dados distribuído tem os seguintes componentes (CA, 2000):

- Um banco de dados coordenador - *CDB (Coordinator Database)* que tem a função de manter os catálogos utilizados pelo servidor Star para manter os dados entre os objetos distribuídos. Quando um usuário solicita uma informação, o servidor acessa o banco de dados coordenador e o banco de

dados associado, através do servidor de banco de dados local para coletar a informação requerida.

- Um catálogo com as definições da instalação de cada banco de dados distribuído e do banco de dados coordenador, além de informações utilizadas para o processamento das consultas.
- Especificações opcionais do usuário sobre links a dados de outros bancos de dados.

O processo de replicação do SGBD Ingres difere da distribuição de dados realizada pelo aplicativo Star. O aplicativo Star tem o objetivo de manter dados entre dois ou mais bancos de dados distribuídos. No processo de atualização, o banco de dados local e os bancos de dados remotos são bloqueados até o término do procedimento. O aplicativo de replicação de dados do Ingres- Ingres/Replicator, atualiza o banco de dados local de modo assíncrono e, após, bloqueia os bancos de dados remotos até o termino do procedimento. Assim, o uso da replicação gera aumento do desempenho (CA, 1998c). Os produtos necessários para implementar um ambiente de replicação são (CA, 1998c):

- O servidor de banco de dados local: Necessário para interpretar e manipular as requisições;
- O componente Ingres/Net: Necessário para acessar os dados dos nós remotos, constituindo-se em uma interface com o protocolo de rede. A figura 4.16 mostra os componentes de um ambiente de replicação do Ingres.

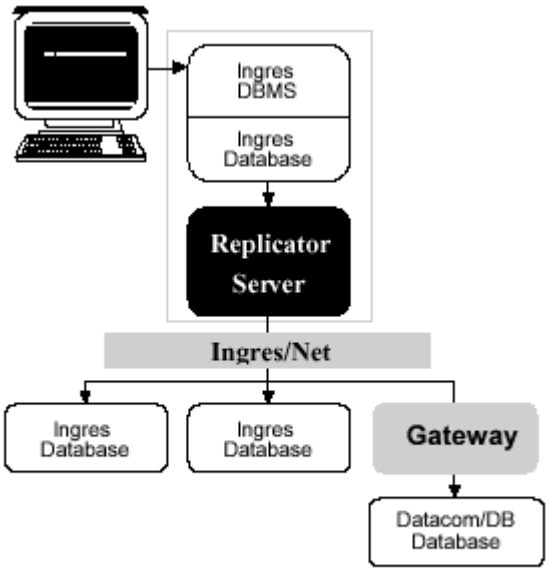


Figura 4.16. Arquitetura de Replicação
Fonte: (CA, 1998c)

O processo de replicação utiliza um Consistent Distributed Data Set (CDDS), responsável por provêr um método para definir a agrupar os dados que são partes do banco de dados global ou partes dos bancos de dados que são replicados entre os sites (CA, 1998c). Pode ser composto por:

a) Todo o banco de dados;
b) Subconjuntos de tabelas - particionamento vertical e horizontal.

A propagação dos dados é feita com base em um caminho (path) que pode ser de três tipos:

a) Origem: Sendo o banco de dados onde a tabela foi modificada;
b) Local: O banco de dados que propaga a mudança para o destino;
c) Destino: O banco de dados que recebe a troca.

Os tipos de replicação determinam como a replicação ocorre e como as colisões são detectadas. Os tipos de replicação suportados são:

a) Full peer: Neste modo, os sites de destino tem a capacidade de utilizar plenamente o processo de replicação e de manter as tabelas ocultas, tabelas de dados e processos de entrada e distribuição. Neste modo, cada tabela tem uma tabela oculta contendo informações necessárias para propagar as modificações e monitorar as trocas nos dados. Quando uma troca é aplicada a uma tabela, um registro é gerado na tabela oculta associada;
b) Read-only protegido: Neste modo, o destino tem informações para detectar colisões entre as cópias dos dados. Os usuários não podem modificar os dados;
c) Read-only sem proteção: Neste modo, não existe proteção quanto a modificações realizadas nas tabelas replicadas. As colisões não são detectadas e não são mantidos informações sobre a tabela destino.

Diversas combinações de ambientes de replicação podem ser configuradas. Os tipos principais são (CA, 1998c):

- **Replicação Central-to-Backup:** Este tipo tem o objetivo de criar um banco de dados de emergência. A figura 4.17 mostra este tipo de replicação;
- **Replicação Peer-to-Peer:** Neste modelo, cada servidor é autônomo no seu site, sendo que cada site tem as mesmas informações sobre as cópias do banco de dados geral. A figura 4.18 mostra este modelo;
- **Replicação em Cascata:** Neste modelo, um banco de dados de origem envia cópias para outro banco de dados. Este banco de dados de destino, por sua vez, copia os dados para outro banco de dados. A figura 4.19 ilustra este modelo;

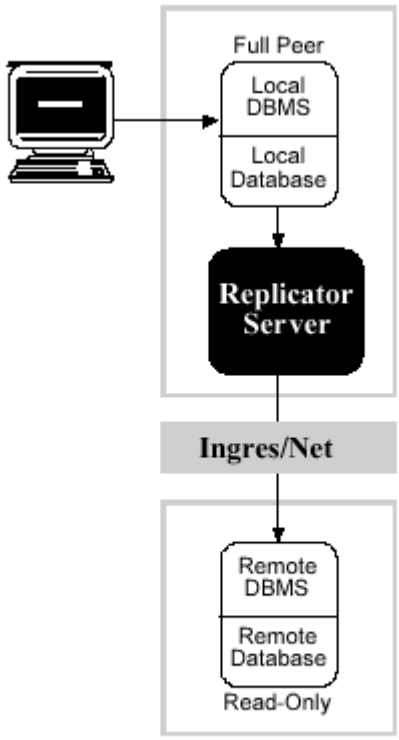


Figura 4.17. Replicação Central-to-Backup
Fonte: (CA, 1998c)

- **Replicação Central-to-Branch:** Neste modo, subconjuntos dos dados de um banco de dados central são enviados para as ramificações. Este modelo permite que as ramificações (branch) realizem modificações nos dados e enviem os dados de volta para o banco de dados central. A figura 4.20 demonstra este modelo;

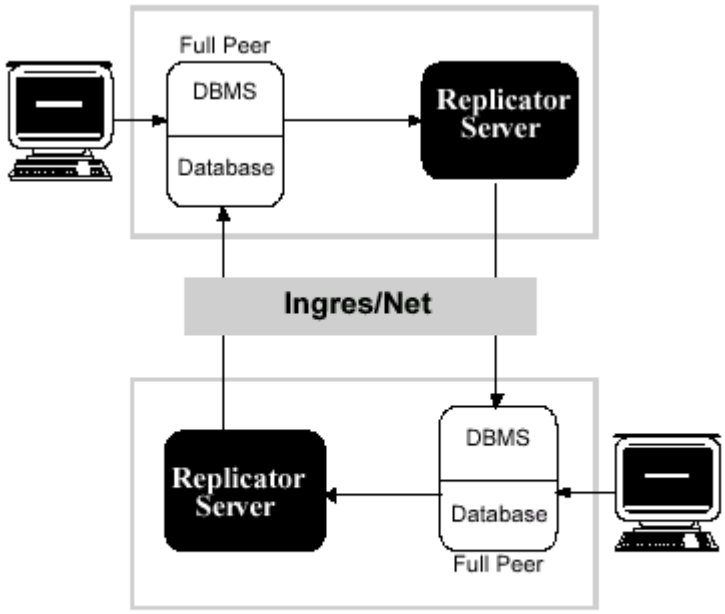


Figura 4.18: Replicação Peer-to-Peer
Fonte: (CA, 1998c)

- **Replicação Hub-and-Spoke:** Neste esquema, o banco de dados central (hub) tem uma relação peer-to-peer com cada um dos banco de dados do raio (spoke). A replicação é feita em cascata, onde cada spoke recebe dados replicados quando o banco de dados central ou qualquer banco de dados do raio é manipulado. A figura 4.21 mostra este modelo.

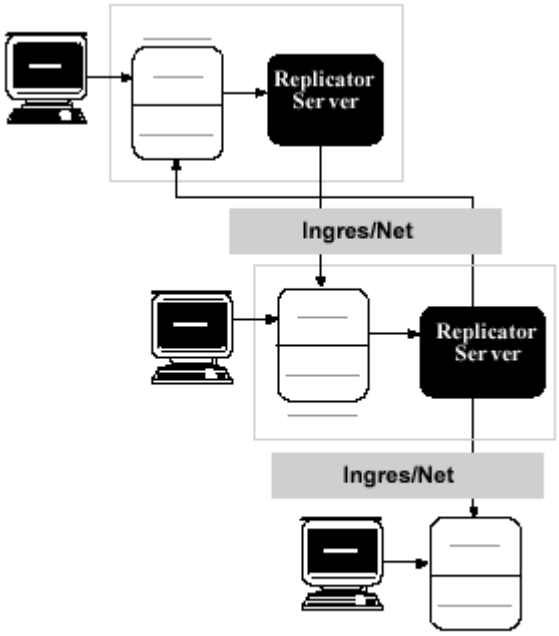


Figura 4.19. Replicação em Cascata
Fonte: (CA, 1998c)

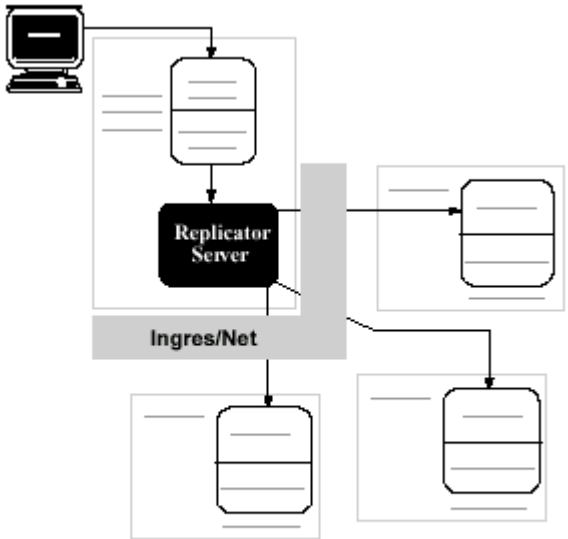


Figura 4.20. Replicação Central-to-Branch
Fonte: (CA, 1998c)

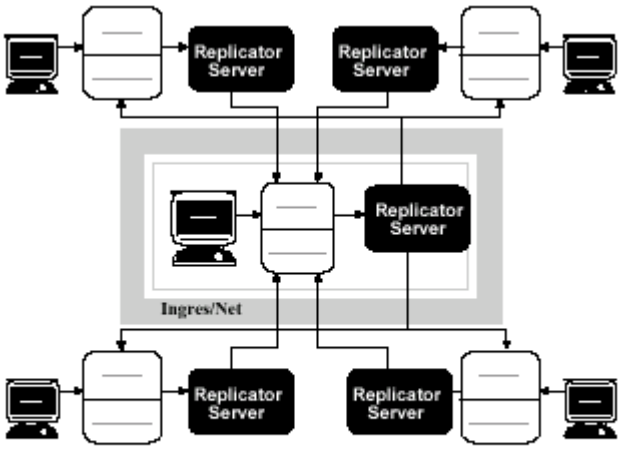


Figura 4.21. Replicação Hub-and-Spoke
Fonte: (CA, 1998c)

4.4 SGBD Sybase ASE

O sistema de gerencia de banco de dados Sybase é composto por (SYBASE, 1997):

- Um banco de dados master: Responsável por controlar a operação e armazenar informações sobre os usuários e suas bases de dados;
- Um banco de dados modelo: Este banco de dados provê modelos para novos bancos de dados dos usuários. Quando um banco de dados é criado, uma cópia do banco de dados modelo é gerada. O banco de dados modelo contém as tabelas do sistema necessárias para cada banco de dados do usuário. O novo banco de dados pode ser customizado pelo usuário adicionando tipos de dados, regras, usuários e privilégios de acesso;
- Um banco de dados de procedures: Os procedimentos do sistema são armazenados no banco de dados systemprocs. Quando um usuário executa um procedimento que inicia com “sp_”, o servidor busca o procedimento no banco de dados do usuário e, caso não encontre, no banco de dados de procedimentos.
- Um banco de dados temporário: O banco de dados temporário - tempdb, provê uma área de armazenamento temporária para tabelas e outros trabalhos temporários.

4.4.1 Replicação e Distribuição de Dados

A replicação e distribuição de dados no SGBD Sybase ASE utiliza um conjunto de componentes para implementar um ambiente de replicação de dados. Os componentes são (SYBASE, 1995):

1) Replication Server: Responsável por coordenar as atividades de replicação para os servidores locais. As principais tarefas são:
    a) Receber modificações dos dados do banco de dados local e distribuí-las para os outros sites;
    b) Receber modificações dos dados de outros servidores e aplicar as modificações no banco de dados local e envia-las para os outros servidores.

    O Replication Server System Database (RSSD) é um banco de dados que contêm as tabelas do sistema para o servidor de replicação (Replication Server). Cada servidor de replicação requer um RSSD e cada RSSD mantêm as tabelas do sistema para um servidor de replicação;
2) SQL Server ou Data Server: Um servidor SQL gerência os banco de dados que contém os dados originais ou replicados. Os clientes utilizam o SQL Server para armazenar e recuperar dados, e para processar transações;
3) Replication Agent ou Log Transfer Manager: Este componente notifica o servidor de replicação das ações no banco de dados que devem ser replicadas para outros bancos de dados;
4) Aplicação Cliente: É um aplicativo que acessa os dados de um servidor;
5) Replication Server Manager (RSM): Permite monitorar os servidores de dados, os servidores de replicação e configurar o ambiente de replicação.

A figura 4.22 mostra os componentes de um sistema de replicação.

O processo de configuração um ambiente de replicação requer um conjunto de configurações antes do processo de replicação poder ser iniciado. Os componentes requeridos para configurar o ambiente de replicação são (SYBASE, 1995):

- ID Server: É um servidor de replicação que registra todos os servidores e banco de dados do ambiente de replicação;
- Especificação de um RSSD e um LTM: RSSD é o banco de dados que mantém as tabelas do sistema. O LTM é o gerenciador dos logs;

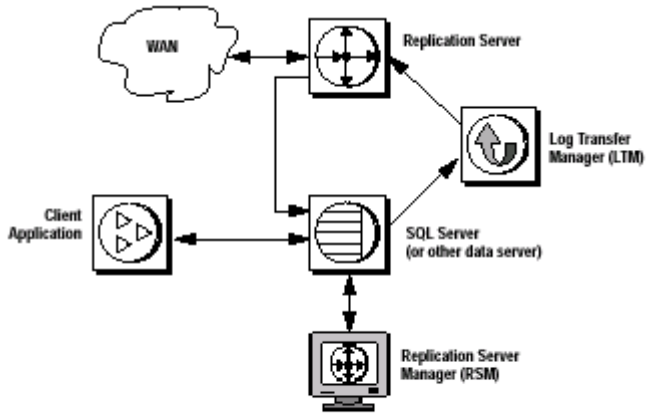


Figura 4.22. Componentes do Sistema de Replicação
Fonte: (SYBASE, 1995)

- Um arquivo de interfaces: Contendo as definições de rede para os servidores do sistema de replicação. O arquivo de interface contém:
    - ID Server;
    - Servidor de Replicação;
    - O RSSD SQL Server para este servidor de replicação;
    - O RSSD LTM para este servidor de replicação;
    - Os servidores de dados que gerência os bancos de dados;
    - Um Backup Server para copiar o banco de dados SQL Server;
    - Os LTMs para os bancos de dados gerênciados por este servidor;
    - Os Servidores de Replicação em outros sites que gerência os banco de dados que são replicados para este servidor;
    - Outros Servidores de Replicação com os quais este servidor tem rota.
- Partições: O servidor de replicação armazena mensagens destinadas para servidores de dados ou para outros sites em partições.

O modelo mais simples de replicação de dados é a distribuição de dados de um banco de dados primário para um ou mais banco de dados replicados (SYBASE, 1995). Neste modelo as cópias replicadas são somente para leitura (read-only). A figura 4.23 apresenta este modelo. Os clientes dos sites remotos podem realizar atualizações nos dados do site primário utilizando funções. Estas funções são criadas com o objetivo de enviar stored procedures para o banco de dados primário ou para os banco de dados replicados. Existem dois tipos de funções:

a) Funções Applied: executadas no banco de dados primário e que afetam os dados primários. O servidor de replicação propaga o procedimento, aplica a mudança nos dados de modo assíncrono nos sites replicados;
b) Funções Request: executadas nos banco de dados replicados e que modificam os dados no site primário. São usadas por aplicações remotas para modificar os dados no site primário.

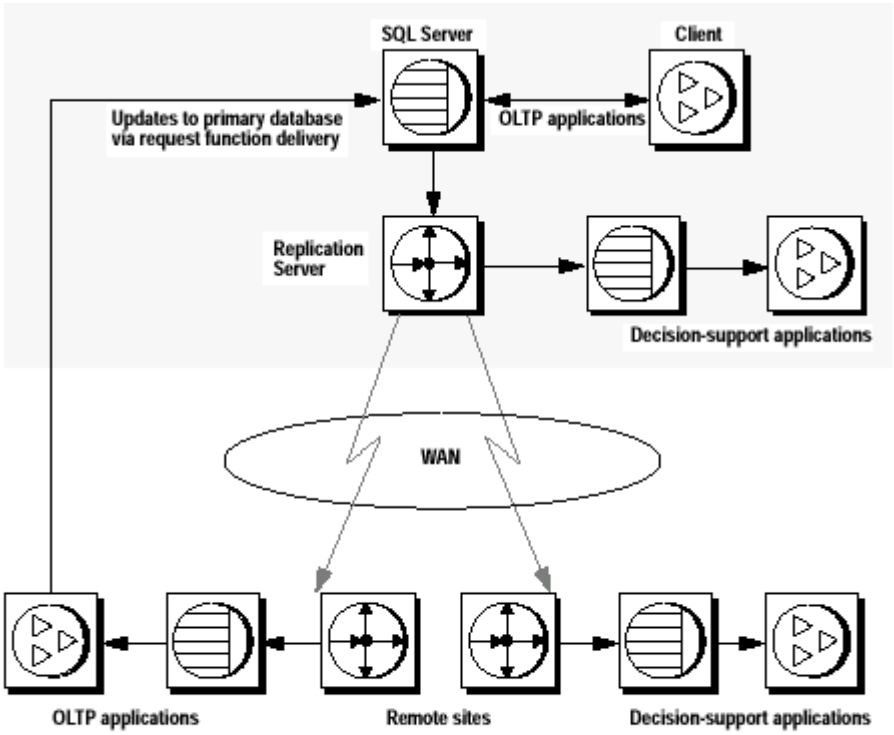


Figura 4.23. Replicação de Dados com Cópia Primária
Fonte: (SYBASE, 1995)

Outras estratégias de replicação podem ser implementadas, principalmente (SYBASE, 1997):

- Mapear Nomes de Tabelas e Colunas com String de Funções: As strings de funções podem ser utilizadas para traduzir o nome de tabelas e colunas para um nome de tabela replicada. Este modelo é útil se no site existem aplicações de clientes que utilizam diferentes tabelas e nomes de colunas que são definidas pela definição da replicação dos dados primários. Estas funções podem ser customizadas permitindo ao servidor de replicação manter os dados em tabelas e não requerendo a modificação das aplicações;

- Criar projeções com string de funções: Este modelo é utilizado no caso do site remoto não necessitar de todas as colunas da tabela primária. A projeção de um subconjunto de colunas de uma tabela pode otimizar o uso de recursos de rede e armazenamento, através da distribuição de partes dos dados;
- Replicar tabelas com múltiplos fragmentos primários: Um fragmento primário é um segmento horizontal de uma tabela que contém a versão original de um conjunto de linhas. Atualizações são aplicadas a esta primeira versão e distribuídas para os sites que tem cópias dos dados. A figura 4.24 apresenta este modelo.

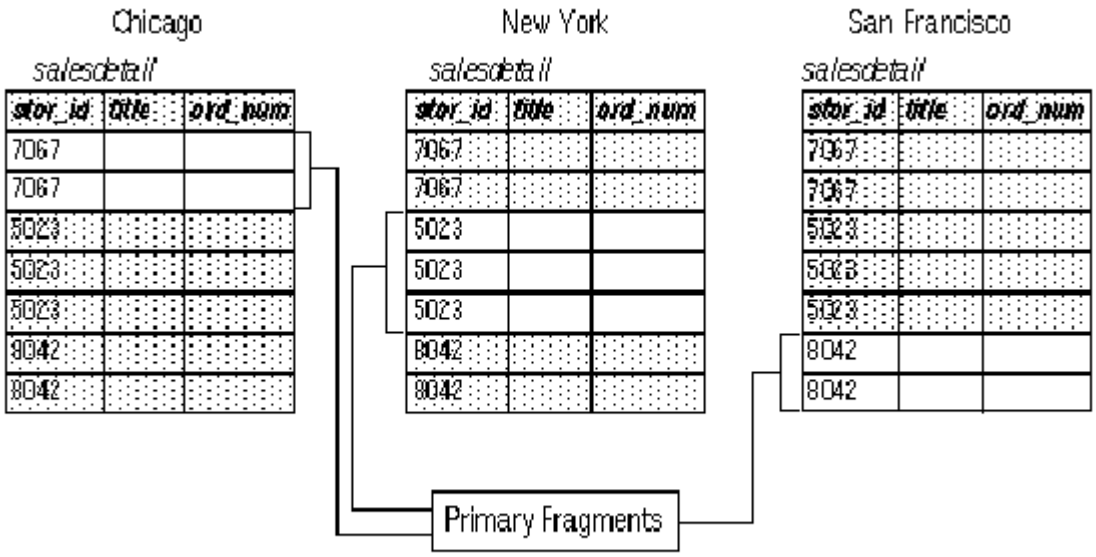


Figura 4.24. Tabela com Múltiplos Fragmentos Primários
Fonte: (SYBASE, 1997)

Os modelos de aplicações baseados em múltiplos fragmentos são (SYBASE, 1997):

a) Distribuição de fragmentos primários: Neste modelo as tabelas de cada site tem cópias primárias e cópias replicadas. Atualizações sobre as cópias primárias são distribuídas para os outros sites. Atualizações em cópias não-primárias são recebidas dos sites primários. A figura 4.25 apresenta este modelo;
b) Rollup corporativo: Este modelo contém múltiplos fragmentos mantidos em sites remotos e consolidados em um único site, agregando tabelas replicadas em um site central. A figura 4.26 apresenta este modelo;

c) Rollup corporativo redistribuídos: Este modelo é o mesmo do modelo anterior, com exceção que a tabela consolidada é redistribuída. A figura 4.27 mostra este modelo;

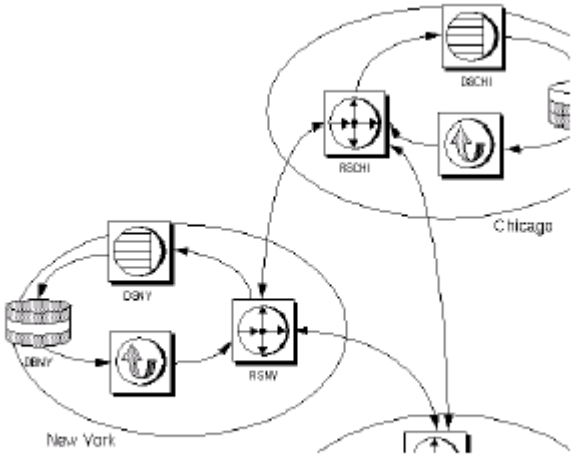


Figura 4.25. Distribuição de Fragmentos Primários
Fonte: (SYBASE, 1997)

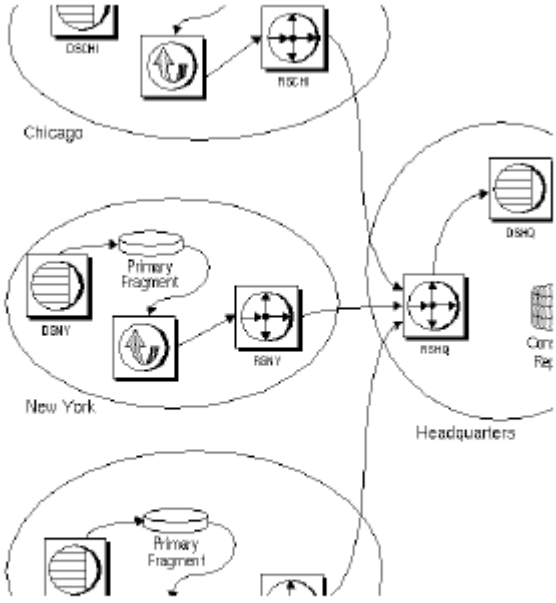


Figura 4.26. Distribuição de Fragmentos Primários com Rollup Corporativo
Fonte: (SYBASE, 1997)

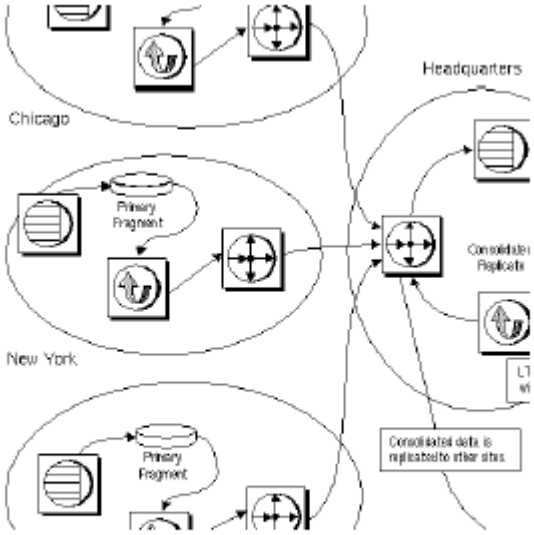


Figura 4.27. Distribuição de Fragmentos Primários com Rollup Corporativo Re-distribuído
Fonte: (SYBASE, 1997)

- Criar aplicações warm standby: Uma aplicação warm standby é uma aplicação do servidor de replicação que mantém um par de banco de dados SQL Server, um com funções para ser uma cópia standby do outro. Os clientes atualizam o banco de dados ativo e o servidor de replicação mantém no banco de dados standby cópias dos dados. No caso de falha do banco de dados ativo, o banco de dados standby pode ser ativado e utilizado pelas aplicações cliente, sem interrupção;
- Utilizar funções de replicação: Com este modelo podem ser executados procedimentos armazenados em outro banco de dados. Estas funções permitem replicar a execução de procedimentos (stored procedures) para outros sites, e implementar desempenho pela replicação de comandos e parâmetros do procedimento que realizou as mudanças;
- Replicar múltiplas cópias de tabelas num site: Este modelo é utilizado para manter diversas cópias de tabelas replicadas em um único site. Pode ser utilizado, por exemplo, por aplicações OLAP;
- Criar aplicações de atualização local: Este modelo permite que aplicações de sites remotos vejam as atualizações realizadas antes do sistema de replicação retorna-las do site primário. Por exemplo, uma aplicação atualiza uma tabela em um site remoto, clientes de outros sites podem visualizar os resultados da transação mesmo que o site primário não esteja acessível. Neste caso, as atualizações realizadas no site local são

enviadas para os sites e, após atualizada com sucesso no site primário, distribuídas para site remotos, incluindo o site onde a transação foi originada.

4.5 Ambiente de Implementação

O ambiente de realização das análises é composto por:

- Três servidores compostos por computadores Pentium III de 650 Mhz, 128 Mb de memória ram e disco de 20 Gb e placa de rede ethernet 100 Mbit/s. Executando sob sistema operacional Red Hat Linux 6.2;
- Uma estação cliente, composta por um computador Pentium III de 650 Mhz, 128 Mb de memória ram e disco de 20 Gb e placa de rede ethernet 100 Mbit/s. Executando sob sistema operacional Microsoft Windows 98;
- A interligação dos equipamentos por meio de um hub de 100 Mbits/s, com cabeamento de rede padrão 568b;
- Desenvolvimento do aplicativo utilizando o ambiente de desenvolvimento NetExpress 3.0, versão universitária, desenvolvido pela MicroFocus Corp., gerando código fonte na linguagem cobol;
- Utilização de drivers de ODBC para a interligação aplicativo-banco de dados, fornecidos pelos desenvolvedores dos SGBDs.

A instalação e configuração do ambiente de testes obedeceu ao seguinte critério:

a. Instalação do sistema operacional nos servidores;
b. Instalação dos SGBD nos computadores servidores;
c. Configuração do SGBD de cada servidor;
d. Instalação e configuração do SGBD cliente e driver de ODBC no computador cliente;
e. Criação do banco de dados e das tabelas em cada site;
f. Configuração do software cliente para acesso aos sites;
g. Inserção dos dados e validação do acesso aos bancos de dados;
h. Criação e configuração do SGBD distribuído nos servidores;
i. Configuração do software cliente para acesso ao banco de dados distribuído;
j. Realização dos testes.

4.5.1 Configuração do Sistema Operacional Linux Red Hat 6.2

O sistema operacional Red Hat 6.2 é configurado como um servidor customizado (custom server), opção que permite a escolha dos pacotes a serem instalados. Os pacotes que foram instalados no sistema estão relacionados no anexo E. Os servidores receberam nome, nome de domínio e endereço IP fixo. Foram cadastrados grupos e usuários para cada banco de dados, sendo a configuração:

Servidor 1:

Nome: server1

Domínio: edu.br

Endereço IP: 192.168.0.251

Arquivo de configuração de rede: /etc/hosts

```
127.0.0.1   localhost.localdomain   localhost
192.168.0.251  server1.edu.br server1
192.168.0.252  server2.edu.br server2
192.168.0.253  server3.edu.br server3
```

Arquivo de grupos (relação dos relativos aos banco de dados): /etc/group

```
kmem:x:9:ingres
oper:x:501:oracle
dba:x:502:oracle
ingres:x:503:ingres
db2fadm1:x:105:
db2iadm1:x:104:db2as
db2asgrp:x:106:db2inst1
Sybase:x:506:
```

Arquivo de usuários (relação dos relacionados aos BD): /etc/passwd

```
oracle:x:500:500::/home/oracle:/bin/bash
Ingres:x:501:9::/home/ingres:/bin/sh
sybase:x:505:506:Sybase ASE DBA account:
   /opt/sybase-11.9.2:/bin/bash
db2fenc1:x:506:105::/home/db2fenc1:/bin/bash
db2inst1:x:507:104::/home/db2inst1:/bin/bash
Db2as:x:508:106::/home/db2as:/bin/bash
```

Sistema de arquivos e diretórios de instalação dos bancos de dados: /home

```
drwxr-xr-x    3 db2as    db2asgrp     4096 Dec 11 14:05 db2as
drwxr-xr-x    2 db2fenc1 db2fadm1     4096 Dec 11 14:04 db2fenc1
drwxr-xr-x    6 db2inst1 db2iadm1     4096 Dec 11 14:50 db2inst1
drwx------   13 ingres   kmem         4096 Dec 17 15:06 ingres
drwx------   51 oracle   oinstall     4096 Dec 26 14:03 oracle
drwxrwxr-x    2 sybase   root         4096 Dec  5 20:27 sybase
```

Servidor 2:

Nome: server2

Domínio: edu.br

Endereço IP: 192.168.0.252

Arquivo de configuração de rede: /etc/hosts

```
127.0.0.1   localhost.localdomain   localhost
192.168.0.251  server1.edu.br server1
192.168.0.252  server2.edu.br server2
192.168.0.253  server3.edu.br server3
```

Arquivo de grupos (relação dos relativos aos banco de dados): /etc/group

```
kmem:x:9:ingres
dba:x:500:oracle
oinstall:x:501:oracle
ingres:x:502:
db2fadm1:x:105:
db2iadm1:x:104:db2as
db2asgrp:x:106:db2inst1,db2inst2
db2iadm2:x:101:db2as
sybase:x:506:
```

Arquivo de usuários (relação dos relacionados aos BD): /etc/passwd

```
oracle:x:500:500::/home/oracle:/bin/bash
ingres:x:501:9::/home/ingres:/bin/sh
sybase:x:505:506:Sybase ASE DBA     account:
   /opt/sybase-11.9.2:/bin/bash
db2fenc1:x:507:105::/home/db2fenc1:/bin/bash
Db2inst1:x:508:104::/home/db2inst1:/bin/bash
Db2as:x:509:106::/home/db2as:/bin/bash
```

Sistema de arquivos e diretórios de instalação dos bancos de dados: /home

```
drwxr-xr-x    3 db2as    db2asgrp     4096 Jul 19 16:08 db2as
drwxr-xr-x    2 db2fenc1 db2fadm1     4096 Jul 19 16:07 db2fenc1
drwxr-xr-x    6 db2inst1 db2iadm1     4096 Jul 28 15:21 db2inst1
drwx------   13 ingres   kmem         4096 Mar 28  2001 ingres
drwxr-xrwx   51 oracle   oinstall     4096 Jul 27 13:05 oracle
drwxrwxrwx    2 sybase   sybase       4096 Jul  6 19:19 sybase
```

Servidor 3:

Nome: server3

Domínio: edu.br

Endereço IP: 192.168.0.253

Arquivo de configuração de rede: /etc/hosts

```
127.0.0.1   localhost.localdomain   localhost
192.168.0.253  server3.edu.br server3
192.168.0.252  server2.edu.br server2
192.168.0.251  server1.edu.br server1
```

Arquivo de grupos (relação dos relativos aos banco de dados): /etc/group

```
kmem:x:9:ingres
dba:x:500:oracle
ingres:x:502:ingres
oinstall:x:501:oracle
sybase:x:503:sybase
db2fadm1:x:102:
db2iadm1:x:101:db2as
db2asgrp:x:103:db2inst1
```

Arquivo de usuários (relação dos relacionados aos BD): /etc/passwd

```
oracle:x:500:500::/home/oracle:/bin/bash
ingres:x:501:9::/home/ingres:/bin/sh
sybase:x:505:503:Sybase ASE DBA account:
   /opt/sybase-11.9.2:/bin/bash
db2fenc1:x:506:102::/home/db2fenc1:/bin/bash
db2inst1:x:507:101::/home/db2inst1:/bin/bash
Db2as:x:508:103::/home/db2as:/bin/bash
```

Sistema de arquivos e diretórios de instalação dos bancos de dados: /home

```
drwxr-xr-x    3 db2as    db2asgrp     4096 Dec 27 13:49 db2as
drwxr-xr-x    2 db2fenc1 db2fadm1     4096 Dec 27 13:49 db2fenc1
drwxr-xr-x    6 db2inst1 db2iadm1     4096 Dec 27 13:56 db2inst1
drwxrwxrwx   13 ingres   kmem         4096 Dec 17 16:52 ingres
drwx------   49 oracle   dba          4096 Dec 26 14:05 oracle
drwxr-xr-x    2 sybase   sybase       4096 Jan 14 18:13 sybase
```

CONCLUSÃO

Este capítulo investiga os sistemas de gerência de banco de dados objetos da análise e apresenta as suas características funcionais. Enfatiza o estudo do ambiente de distribuição e replicação implementado por cada SGBD e apresenta os modelos de distribuição e replicação de dados suportados.

Os SGBD analisados apresentam semelhanças no modo de implementar o ambiente de replicação e distribuição, basicamente consistindo de:

a) Um aplicativo responsável por configurar o ambiente de replicação e distribuição, conexões entre as bases de dados, usuários locais e remotos, tipos de replicação e distribuição, tabelas a serem distribuídas e/ou replicadas ;
b) Um módulo do SGBD responsável por gerênciar - capturar e aplicar, as mudanças nos dados e manter a integridade das cópias entre os sites;
c) Um módulo responsável por configurar as conexões ou ligações lógicas entre os sites participantes do ambiente de replicação e/ou distribuição, sob o protocolo TCP/IP;
d) A possibilitadade de implementar a distribuição de tabelas completas e a replicação de tabelas completas ou subconjuntos (fragmentos) das tabelas;
e) A replicação de tabelas com possibilidade de atualização ou para consulta (read-only);
f) A implementação da propagação das alterações nos dados para os sites participantes da replicação por intervalo de tempo ou associada a procedimentos armazenados no banco de dados.

5 Análise das Propriedades de Transparência de Localização, Nomeação e Replicação

Introdução

Este capítulo desenvolve duas análises. A primeira análise avalia os sistemas de gerência de bancos de dados com relação as propriedades de transparência de localização e de nomeação dos SGBD distribuídos. E investiga o modo como são implementados estas propriedades nos sistemas de banco de dados analisados. O ambiente para a realização desta análise é composto por três bases de dados interligadas por uma rede local. Nesta análise os sistemas de gerência de banco de dados devem oferecer recursos para esconder do usuário os detalhes quanto a localização física dos dados (transparência de localização) e os nomes dos objetos do banco de dados - em específico da tabela utilizada para os testes (transparência de nomeação). O fato de que as bases de dados estão distribuídas sobre uma rede de comunicação faz com que necessitem utilizar algoritmos para a otimização de consultas distribuídas, conduzindo a uma avaliação do desempenho de cada SGBD na realização de consultas distribuídas. Esta avaliação é desenvolvida através de um aplicativo que realiza as consultas e relata o tempo dispendido, sendo o resultado apresentado em tabelas comparativas.

A segunda análise tem o objetivo de avaliar a implementação de um ambiente de replicação dos dados. Investiga os recursos do sistema de gerência do banco de dados na implementação de cópias dos dados entre duas bases de dados - completos e autônomos, interligadas por uma rede local. A análise avalia como é implementado o ambiente de replicação, sendo necessário que os sistemas de gerência de banco de dados possuam recursos para a interligação entre as bases de dados e para a manutenção e implementação de consistência dos dados remotos, entre os sites participantes.

Esta investigação é importante pois estas propriedades permitem que sejam implementados ambientes de replicação e distribuição variados, resolvendo problemas como a necessidade de atualizações remotas de dados - somente para consulta, até um ambiente onde cada site realiza atualizações e consultas em todos os sites participantes.

5.1 Análise 1 - Transparência de Localização e Nomeação

Este primeiro estudo avalia um ambiente de banco de dados distribuído e a implementação das propriedades de transparência de localização e transparência de nomeação dos SGBDDs. O ambiente de testes implementado é constituído por:

- Três sites, contendo um SGBD autônomo e independente, executando o sistema operacional Linux Red Hat 6.2. Em cada site é criada uma base de dados (figura 5.1). O nome das bases de dados são: *server1* - para a base de dados do servidor server1.edu.br, *server2* - para a base de dados do servidor server2.edu.br e *server3* - para a base de dados do servidor server3.edu.br. No site número 2 é criado uma base de dados onde são realizadas as configurações para acesso as três bases de dados locais. Esta base de dados constituirá uma base de dados distribuída, uma vez que não possui nenhuma tabela local. É criado um banco de dados e, nesta base de dados, implementado o acesso a cada uma das bases de dados remotas. A avaliação do acesso e da transparência de localização e nomeação, bem como a desempenho na implementação das consultas é feita sobre esta base de dados distribuída;
- Em cada base de dados é criada uma tabela, com o nome de *tabela*. Na base de dados distribuída deve ser realizado o acesso a cada tabela criada. As tabelas são compostas pelos atributos:
  a. Código: é a chave do registro. Definida como um campo numérico de seis caracteres, com o nome de TABELA_COD;
  b. Descrição: descrição do registro. Definido como um campo caráter de 40 caracteres, com o nome de TABELA_DESCR;
  c. Local: descreve o site onde a tabela está localizada. Definido como um campo alfanumérico de 20 caracteres e com o nome de TABELA_LOCAL;
  d. Inserção: descreve o número da inserção dos dados. Incrementado a cada inserção, assim, caso sejam inseridos dados mais de uma vez, este campo controla quando o dado foi inserido. O critério base para

a inserção dos dados é faze-lo em três etapas, conforme o descrito abaixo. É um campo numérico de três caracteres, com o nome de TABELA_INSERCAO.

e. O volume de dados inseridos é: no site 1, 80.000 registros, divididos em: 25.000 como primeira inserção, 35.000 como segunda inserção e 20.000 como terceira inserção. No site 2, 100.000 registros, divididos em: 30.000 como primeira inserção, 45.000 como segunda inserção e 25.000 como terceira inserção. No site 3, 150.000 registros, divididos em: 40.000 como primeira inserção, 50.000 como segunda inserção e 60.000 como terceira inserção.

- Uma estação cliente, executando o sistema operacional Windows 98, configurado com SGBD clientes e executando um aplicativo desenvolvido com o objetivo de inserir os dados e implementar consultas sobre as bases de dados.
- A rede de comunicação é consituída por uma rede local ethernet, 100 Mb/s, inter-conectada por meio de um hub.

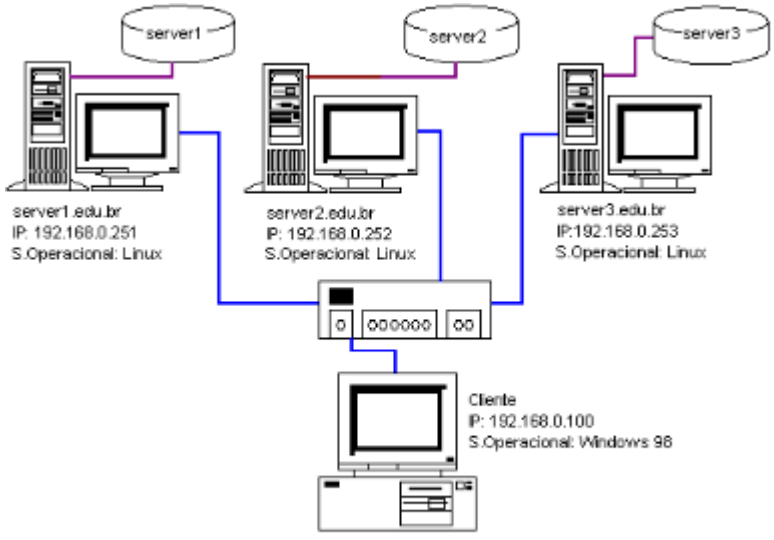


Figura 5.1. Ambiente da Análise nº 1

O teste consiste na realização das etapas descritas abaixo:

1. Configuração dos bancos de dados locais;
2. Inserção de dados nos bancos de dados locais;

3. Configuração do banco de dados distribuído;
4. Execução de consultas sobre a base de dados distribuída e registro do tempo despendido na realização da mesma;
5. Avaliação dos resultados obtidos.

5.1.1 SGBD DB2

A configuração realizada no sistema de gerência de banco de dados DB2 é a seguinte:

1) Instalação: O processo de instalação do banco de dados DB2, descrito em SCOTT (2001), é feito nas etapas:
   - Instalação dos pacotes descritos;
   - Extração dos arquivos de instalação com o comando tar xvf 018_EEE_LNX_NVL.tar, em um diretório temporário;
   - A partir do diretório de temporário, subdiretório 018_EEE_LNX_NVL, executar o programa ./db2setup, como usuário root;
   - No processo de instalação, criar as instâncias e o administrador para o sistema - neste processo são criados os usuários db2inst1, db2as e db2fenc1 e os diretórios para o sistema, sendo: /home/db2inst1 e /home/db2as e /home/db2fenc1;
   - Definir uma senha para o usuário db2inst1.
2) Iniciar o sgbd: executar os comandos db2 start e db2admin start;
3) Executar o aplicativo de sql interativo: digitar db2;
4) Para criar o banco de dados: create database server*N*, onde *N* é o número da base de dados;
5) Conectar com o banco de dados: connect to server*N;*
6) Criar a tabelas:

   create table TABELA (TABELA_COD dec(6,0) not null primary key,
   TABELA_DESCR char(40) not null, TABELA_LOCAL char(20) not null,
   TABELA_INSERCAO dec(3,0) not null)

A figura 5.2 mostra as bases de dados criadas.

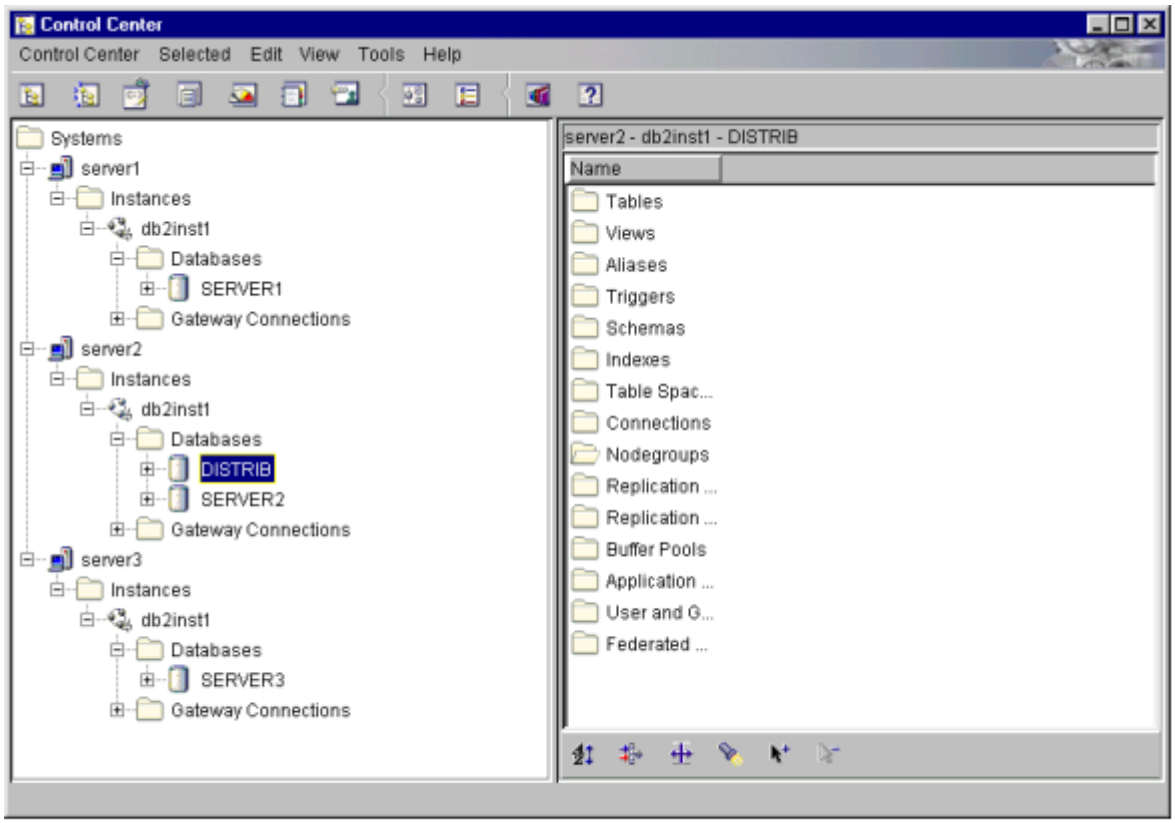


Figura 5.2. Bases de Dados do SGBD DB2

Este processo é realizado nas três bases de dados. No site 2, onde reside a base de dados distribuída, são realizadas as etapas:

1) Criar um banco de dados distribuído, com o nome de distrib;
2) Catalogar os nós dos bancos de dados, como os comandos:
    a) Para o servidor 1: catalog TCPIP node server1 remote 192.168.0.251 server db2cdb2inst1 remote_instance db2inst1 system server1 ostype linux;
    b) Para o servidor 2: catalog TCPIP node server2 remote 192.168.0.252 server db2cdb2inst1 remote_instance db2inst1 system server2 ostype linux;
    c) Para o servidor 3: catalog TCPIP node server3 remote 192.168.0.253 server db2cdb2inst1 remote_instance db2inst1 system server3 ostype linux;
    d) Para verificar, executar o comando list node directory, no sql interativo.

    A figura 5.3 mostra a distribuição dos servidores e nós.
3) Criar um método de acesso as bases de dados com o comando: create wrapper “DRDA” library “libdrda.a”;
4) Registrar os servidores com o comando:

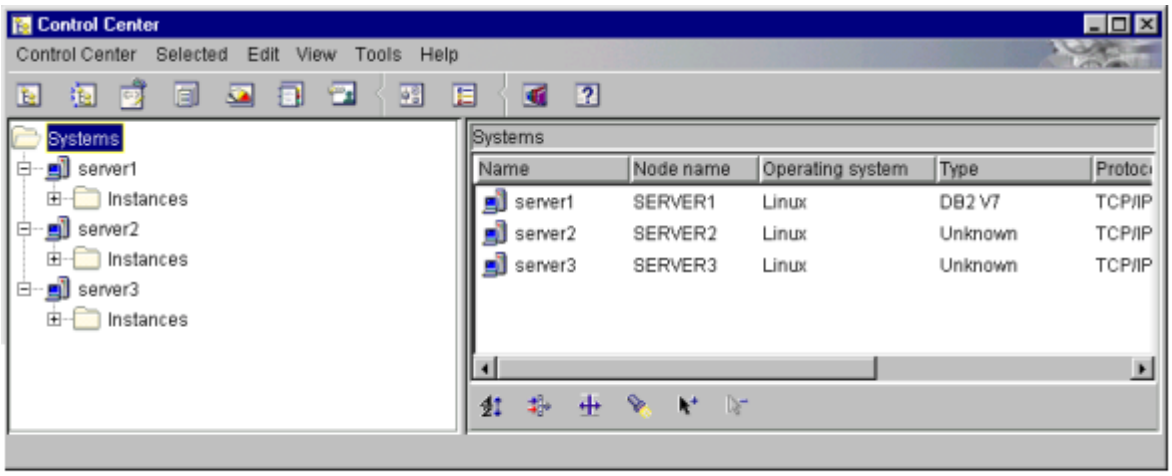


Figura 5.3. Sites do SGBD DB2

a) Para o servidor 1: create server db2_server1 type db2/linux version 7.2 wrapper DRDA authid db2inst1 password db2001 options (node ´s1_node´, dbname ´server1´);
b) Para o servidor 2: create server db2_server2 type db2/linux version 7.2 wrapper DRDA authid db2inst1 password db2001 options (node ´s2_node´, dbname ´server2´)
c) Para o servidor 3: create server db2_server3 type db2/linux version 7.2 wrapper DRDA authid db2inst1 password db2001 options (node ´s3_node´, dbname ´server3´)

5) Mapear os usuários locais para usuários remotos, como o comando:
   a) Para o servidor 1: create user mapping for db2inst1 server db2_server1 options (remote_authid ´db2inst1´, remote_password ´db2001´);
   b) Para o servidor 2: create user mapping for db2inst1 server db2_server2 options (remote_authid ´db2inst1´, remote_password ´db2001´);
   c) Para o servidor 3: create user mapping for db2inst1 server db2_server3 options (remote_authid ´db2inst1´, remote_password ´db2001´);

6) Registrar as tabelas locais para tabelas distribuídas:
   a) Para o servidor 1: create nickname db2inst1.TABELA1 for “db2_server1”.”db2inst1”.”TABELA”;
   b) Para o servidor 2: create nickname db2inst1.TABELA2 for “db2_server2”.”db2inst1”.”TABELA”;

c) Para o servidor 3: create nickname db2inst1.TABELA3 for “db2_server3”.”db2inst1”.”TABELA”;

A figura 5.4 mostra o banco de dados distribuído com os sites remotos.

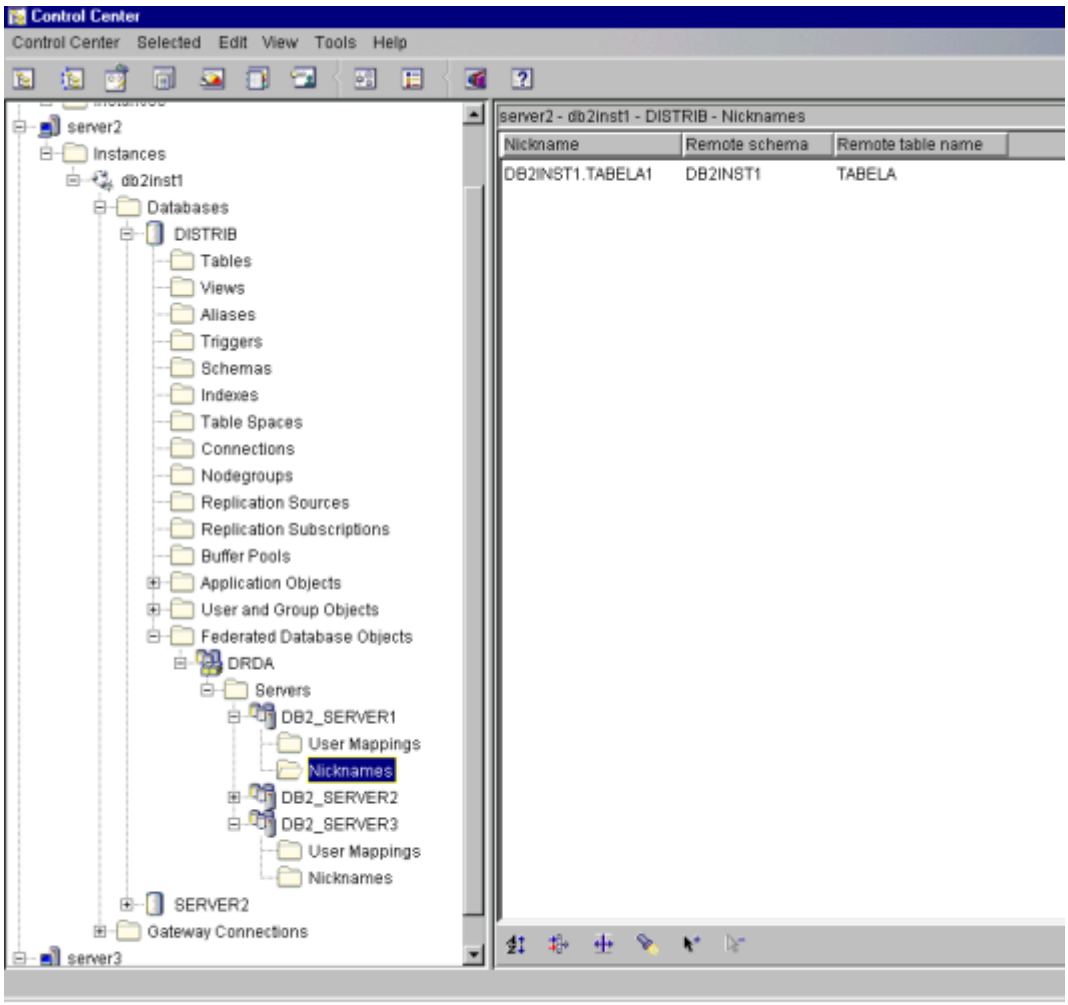


Figura 5.4. Bases de Dados Distribuídas no SGBD DB2

5.1.2 SGBD Ingres

A configuração do SGBD Ingres 2.0 segue as instruções do documento Ingres II Howto, DOMOCOS (1999) executado nas seguintes etapas:

1) Instalação: a instalação é realizada nas seguintes etapas:
   a) Criar um usuário com o nome de ingres e grupo kmem e definir uma senha;
   b) Criar um diretório temporário - neste caso, /home/mestrado/ingres;
   c) Copiar o arquivo de instalação para o diretório temporário;

d) Extrair o arquivo de instalação com o comando tar xvf /home/mestrado/ingres/OI120000.TAR e executar o comando install/ingbuild, a partir do prompt;

e) Configurar as variáveis de ambiente, no arquivo .profile do diretório /home/ingres, sendo: II_SYSTEM=/home, TERM_INGRES=vt100f, adicionar ao PATH as entradas $II_SYSTEM/ingres/utility, e criar uma linha com a entrada LD_LYBRARY_PATH=/lib:/usr/lib:$II_SYSTEM/ingres/lib;

f) Iniciar a instalação com o comando ./install/ingbuild, colocando como caminho o diretório onde estão os arquivos de instalação - /home/mestrado/ingres/OI120000.TAR.

2) Para iniciar o sistema: logar como ingres e digitar ingstart;

3) Criar a base de dados: createdbe server*N*, sendo *N* o número da base de dados, a partir do prompt.

4) Iniciar o monitor sql: digitar isql e criar a tabela com o comando: create table TABELA (TABELA_COD dec(6,0) not null primary key,

TABELA_DESCR char(40) not null,

TABELA_LOCAL char(20) not null,

TABELA_INSERCAO dec(3,0) not null);

Para criar a base de dados distribuída no site 2:

1) Criar um banco de dados distribuído com o comando: createdb distribuído/star;

2) Registrar os sites com o utilitário netutil informando o nome do nó, protocolo e endereço IP de cada servidor. A figura 5.5 apresenta o utilitário netutil;

3) Registrar as tabelas remotas utilizando o utilitário isql com o comando: isql distribuido/star. Os comandos para registrar as tabelas são mostrados abaixo. A :

a) No servidor 1: register TABELA1 as link from ingres.TABELA with node=server1, database='server1';

b) No servidor 2: register TABELA2 as link from ingres.TABELA with node=server2, database='server2';

c) No servidor 3: register TABELA3 as link from ingres.TABELA with node=server3, database='server3'.

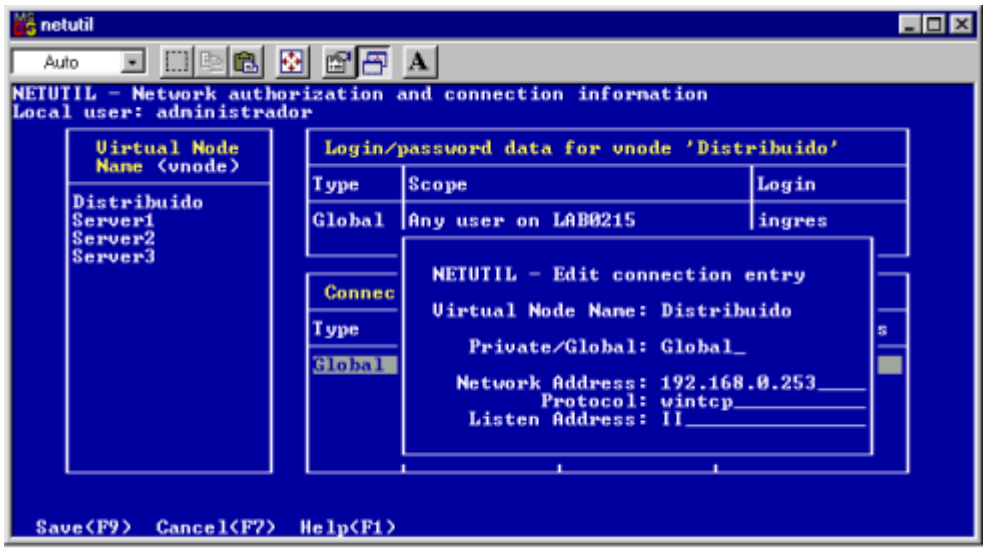


Figura 5.5. Utilitário Netutil do SGBD Ingres

A figura 5.6 mostra os sites no SGBD Ingres.

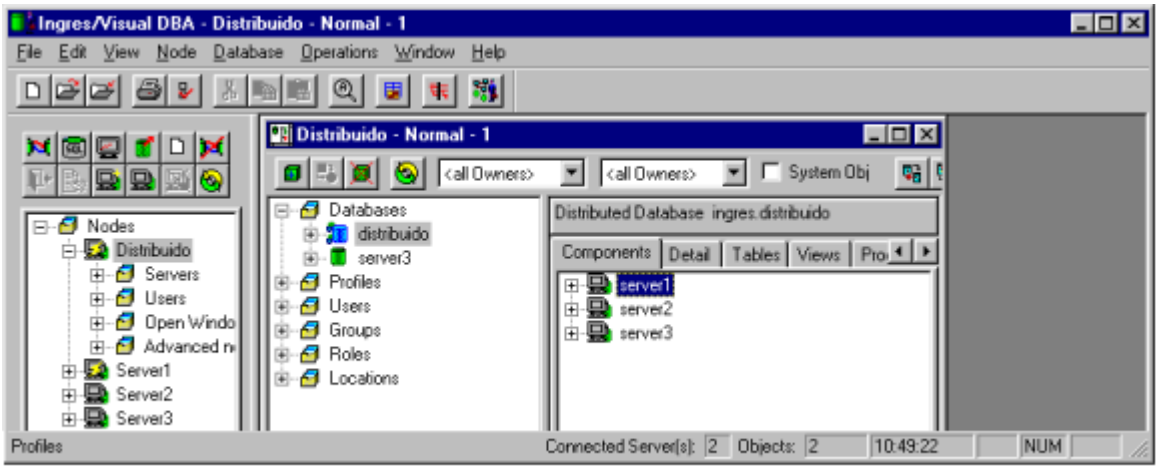


Figura 5.6. Sites do SGBD Ingres

A figura 5.7 mostra as tabelas do banco de dados distribuído.

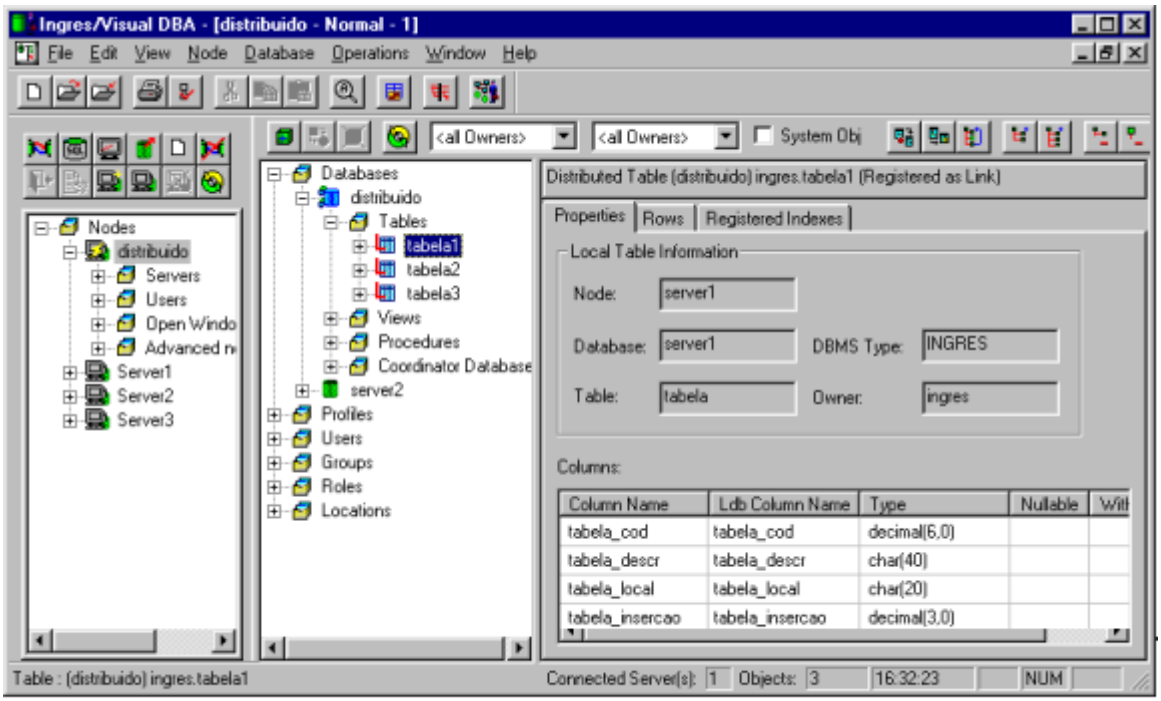


Figura 5.7. Tabelas do SGBD Distribuído

5.3 SGBD Oracle

A configuração do SGBD Oracle é feita conforme o documento ORACLE (2000) e segue as etapas:

1) Instalar o interpretador java jdk118_v3-glibc-2.1.3.tar.bz2, obtido do site: www.blackdown.org, e estabelecer um link entre os diretórios com o comando: ln -s /usr/local/jdk118_v3 /usr/local/java;
2) Instalar os pacotes: `compat-binutils-5.2-2.9.1.0.23.1.i386.rpm;`
   `compat-glibc-5.2-2.0.7.2.i386.rpm;`
   `compat-egcs-5.2-1.0.3a.1.i386.rpm;`
   `compat-egcs-c++-5.2-1.0.3a.1.i386.rpm;`
   `compat-libs-5.2-2.i386.rpm;`
3) Adicionar os usuário: oracle e os grupos: oinstall, dba e oper;
4) Setar as variáveis de ambiente:
   ORACLE_BASE=/home/oracle;
   ORACLE_HOME=/home/oracle;

ORACLE_SID=ORLI;

ORACLE_TERM=linux;

LD_LIBRARY_PATH=/home/oracle/lib;

DISPLAY=serverN.edu.br:0.0, onde N é o número do servidor;

e adicionar ao path o caminho /home/oracle/bin;

5) Extrair os arquivos de instalação, em um diretório temporário, com o comando: tar xvf linux81701.tar;
6) Ir para o diretório Disk1, iniciar uma sessão Xwindows, digitando startx, abrir uma janela de terminal e iniciar a instalação digitando ./runInstaller;
7) Alterar o arquivo /etc/oratab, substituindo o N por Y, de modo a definir a inicialização do banco de dados;
8) Para iniciar o banco digitar: startdb;
9) Iniciar o serviço de comunicação digitando: lsnctrl start;
10) Para criar a base de dados - se não criada no processo de instalação, iniciar a interface gráfica digitando startx e executar o programa dbassist;
11) Para criar a tabela, iniciar o monitor sql digitando: sqlplus;
12) Conectar com a base de dados digitando: connect sysadm manager as sysdba, sendo *sysadm* o usuário e *manager* a senha de acesso;
13) Criar a tabela digitando: create table TABELA

(TABELA_COD dec(6,0) not null primary key,
TABELA_DESCR char(40) not null,
TABELA_LOCAL char(20) not null,
TABELA_INSERCAO dec(3,0) not null)

A configuração do banco de dados distribuído é feita:

1) Criar uma base de dados com o nome de distribuído, utilizando o aplicativo dbassist;
2) Através do aplicativo Visual Studio, criar um usuário com o nome de mestrado;
3) Utilizando a aplicativo Net8, criar as conexões entre os bancos de dados (database links). A figura 5.8 mostra o registro dos database links;
4) Criar os vínculos com os bancos de dados remotos, através do aplicativo Visual Studio, registrando os bancos de dados remotos no banco de dados distribuído. A figura 5.9 mostra o registro dos sites remotos;

5) Definir os sinônimos para as tabelas utilizando o aplicativo Visual Studio. A figura 5.10 mostra a definição dos sinônimos para as tabelas remotas.

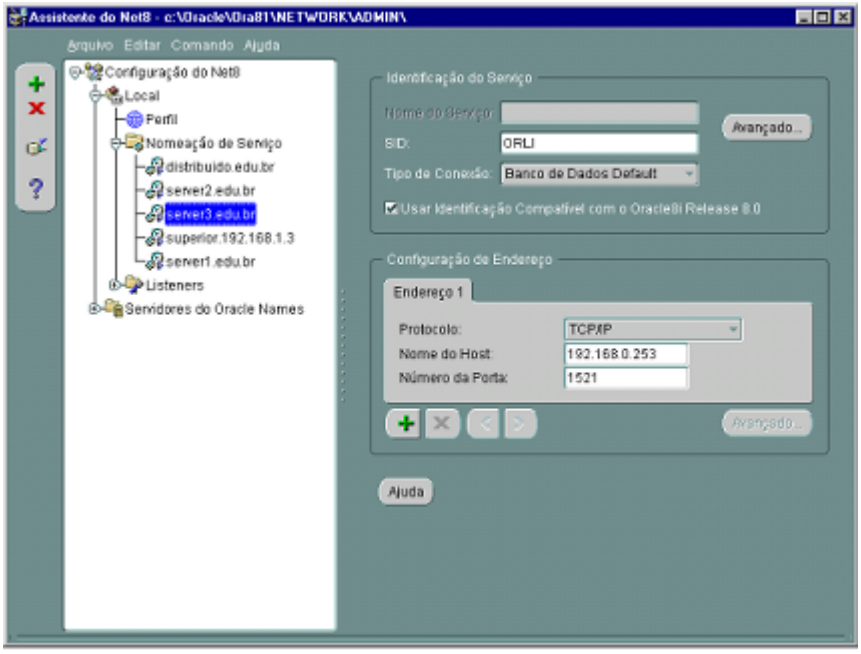


Figura 5.8. Database Links no Net8

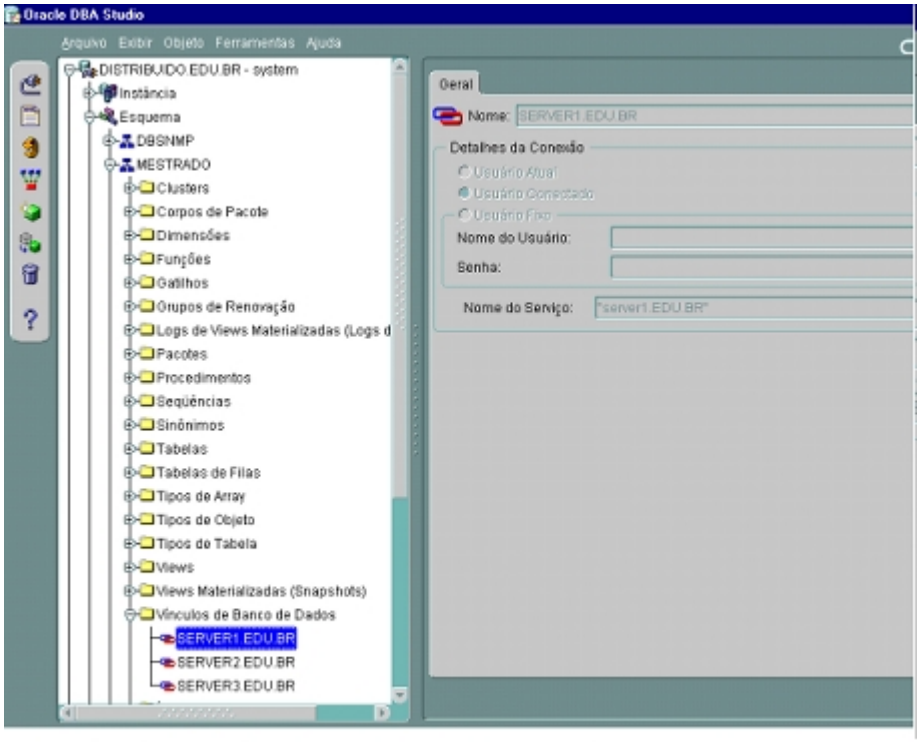


Figura 5.9. Registro dos Bancos de Dados Remotos

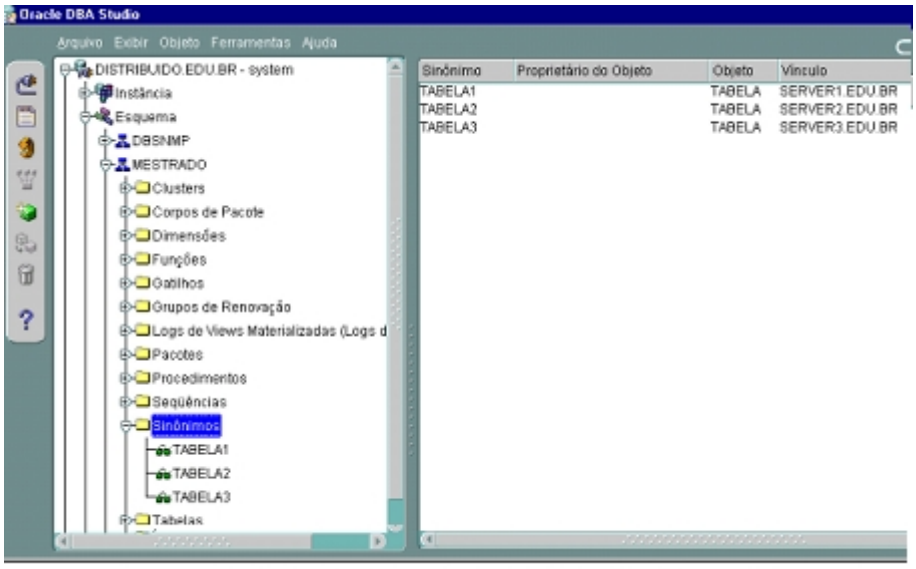


Figura 5.10. Sinônimos das Tabelas Remotas

5.1.4 SGBD Sybase ASE

A configuração do SGBD Sybase ASE segue as instruções do documento SYBASE (2001), nas seguintes etapas:

1) Instalação: como root, instalar em um diretório temporário os arquivos e adiciona-los ao sistema com o comando rpm, sendo:
   rpm -ivh sybase-common-11.9.2.rpm
   rpm -ivh sybase-ase-11.9.2.rpm;
2) Definir uma senha para o usuário sybase, com o comando: passwd sybase;
3) Logar como usuário sybase, iniciar uma sessão Xwindows e executar o programa de configuração com o comando ./svrbuild, a partir do diretório bin;
4) Definir como device o diretório /home/sybase com tamanho de 450 Mb, e os diretórios temp com 30 Mb e temproc com 60 Mb, e selecionar a opção build;
5) Renomear o arquivo RUN_server01 para RUN_SYBASE, onde server01 é o nome do servidor. Substituir pelo nome do site, nos outros sites substituir por server02 e server03;
6) Iniciar o banco digitando: cd install e startserver RUN_SYBASE;

7) Iniciar o monitor sql como o comando: isql -Usa -Sserver*N*, onde *sa* é o administrador e *N* o servidor;
8) Criar o banco de dados digitando: create database server*N* on master = 250 log on master = 30 with override go;
9) Acessar a base de dados: use server*N* go;
10) Criar a tabela digitando:create table TABELA

(TABELA_COD dec(6,0) not null primary key,
TABELA_DESCR char(40) not null,
TABELA_LOCAL char(20) not null,
TABELA_INSERCAO dec(3,0) not null)

A configuração do banco de dados distribuído é feito do seguinte modo:

1) Criar uma base de dados, com o comando create database distribuido, a partir do isql interativo;
2) Acessar a base de dados distribuída digitando: use distribuído go;
3) Registrar os servidores, sendo:
    a. Para o servidor 1: sp_addserver server1_d, sql_server, server1;
    b. Para o servidor 2: sp_addserver server2_d, sql_server, server2;
    c. Para o servidor 3: sp_addserver server3_d, sql_server, server3;

Sendo que os locais server1, server2 e server3 devem estar registrados na tabela de interfaces, localizada no diretório /opt/sybase, arquivo interfaces, do seguinte modo:

Server1_d
Master tcp ether server1 4100
Query tcp ether server1 4100

Onde: server1_d o nome do site remoto, server1 o nome local e 4100 a porta de comunicação. A tabela 5.11 mostra a configuração dos servidores. Deve existir uma entrada para cada servidor;

4) Habilitar o banco como distribuido digitando: sp_configure “enable CIS”,1;
5) Registrar as tabelas distribuídas, mostradas na figura 5.12. Os comandos são:
    a. Para o servidor 1: create existing table TABELA1 at “server1_d.server1.dbo.TABELA” go;
    b. Para o servidor 2: create existing table TABELA2 at “server2_d.server2.dbo.TABELA” go;

c. Para o servidor 3: create existing table TABELA3 at “server3_d.server3.dbo.TABELA” go;

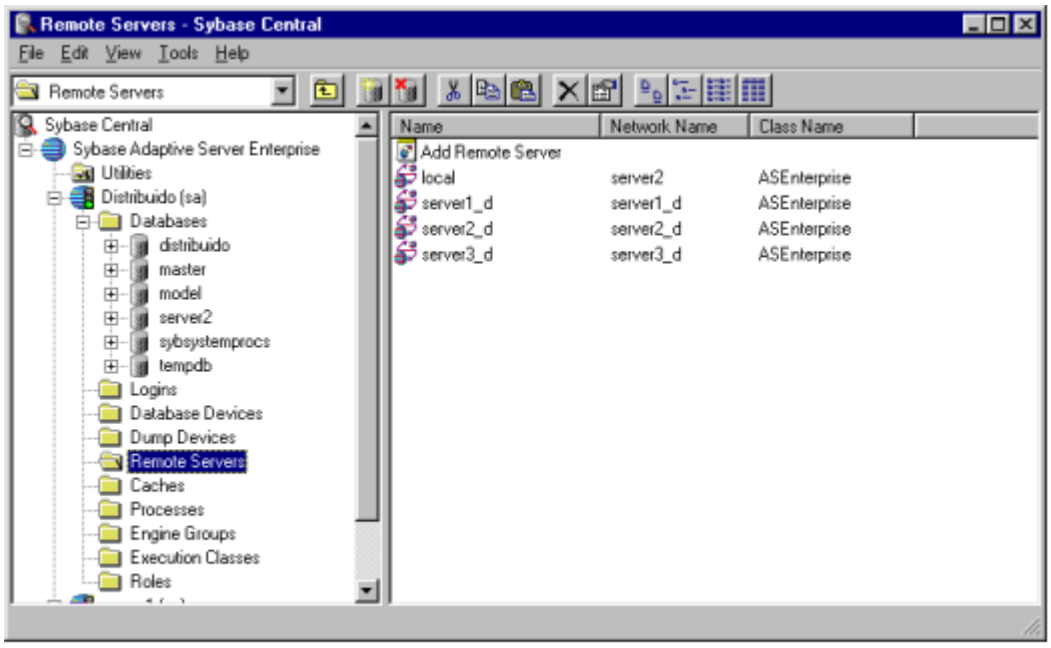


Figura 5.11. Sites Remotos no SGBD Sybase ASE

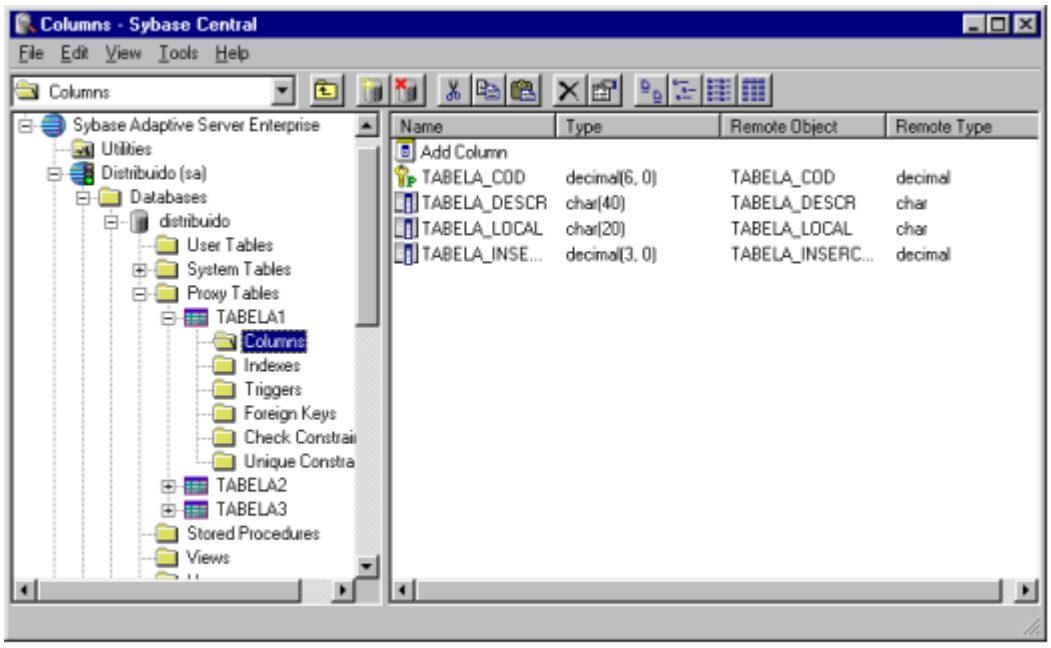


Figura 5.12. Tabelas Distribuídas no SGBD Sybase

5.1.5 Aplicativo para Análise dos SGBD

Para a realização da primeira análise foi desenvolvido um aplicativo utilizando a ferramenta de desenvolvimento NetExpress 3.0 University Edition, desenvolvida pela empresa MicroFocus Co. O programa desenvolvido tem a interface demonstrada na figura 5.13.

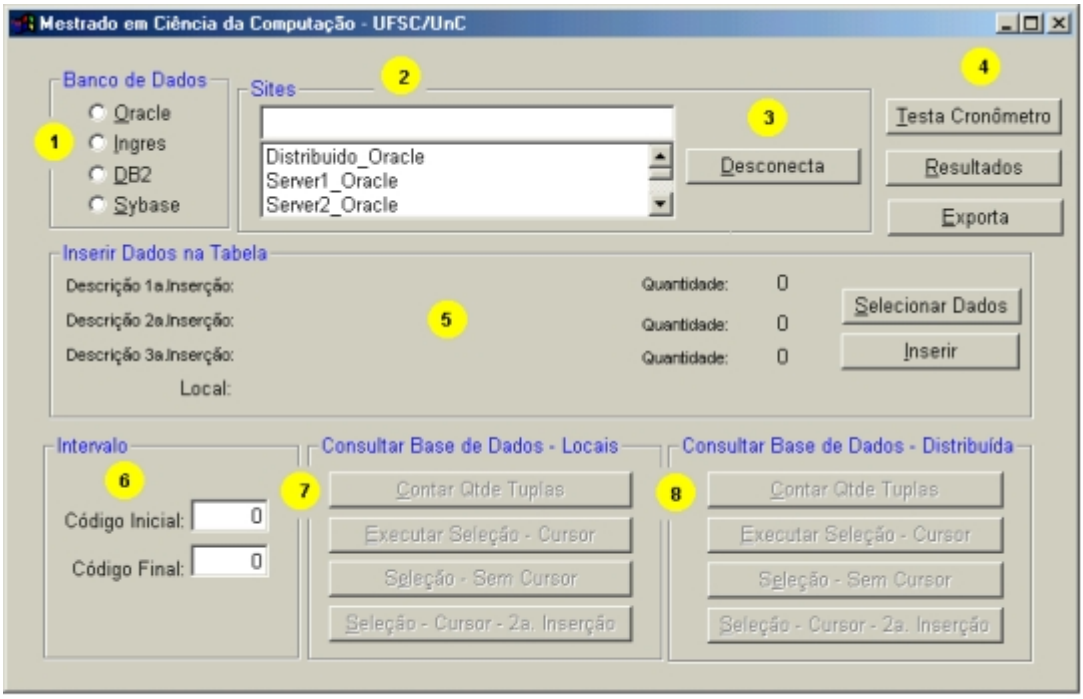


Figura 5.13. Aplicativo para Teste dos SGBD

Os campos numerados indicam as funções do programa, sendo:

1. Campo para seleção do SGBD e realizar a conexão com o mesmo;
2. Campo para selecionar o site, sendo que as opções são:
   a. Server1 - indicando o primeiro servidor;
   b. Server2 - indicando o segundo servidor;
   c. Server3 - indicando o terceiro servidor;
   d. Distribuído - indicando o servidor que executa o banco de dados distribuído.
3. Campo para desconectar do banco de dados;
4. Campo para testar o cronômetro do sistema, visualizar os resultados dos testes e exportar o resultado para um arquivo texto;
5. Campo para inserir os volumes de dados nos bancos. As quantidades estão pré-definidas no programa, obedecendo as quantidades definidas no item 5.1;

6. Campo para determinar o registro inicial e final para a consulta. É um critério para selecionar parte dos registros das bases de dados. Assim, informando-se zeros como valor inicial e 999999 como valor final a pesquisa é feita sobre toda a base de dados. Qualquer valor diferente disto limita a consulta a faixa de valores informada;
7. Campo para selecionar o tipo de consulta a ser realizada sobre os bancos de dados locais. Somente é habilitada se os sites selecionados forem bancos de dados locais (server1, server2 e server3);
8. Campo para selecionar o tipo de consulta a ser realizada sobre o banco de dados distribuído. Somente é habilitada se o site selecionados for o banco de dados distribuido.

A figura 5.14 mostra a tela de teste do cronômetro do sistema. Isto é utilizado para aferir a contagem do tempo realizada pelo aplicativo.

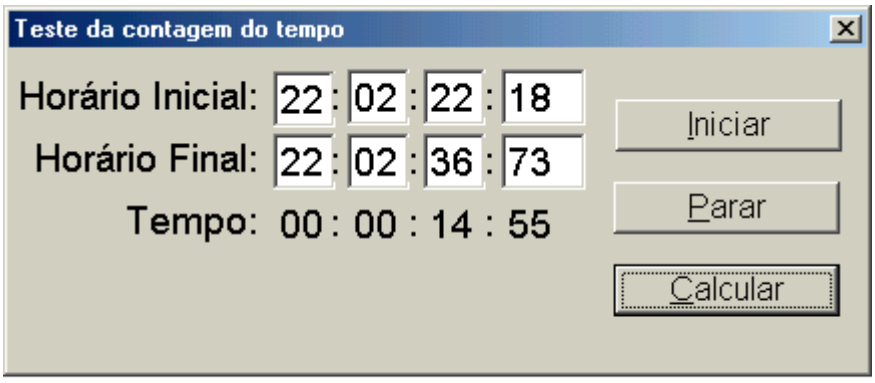


Figura 5.14. Aferição do Cronômetro do Aplicativo

A figura 5.15 mostra a seleção do SGBD Oracle e da primeira base de dados, localizada no primeiro servidor (server 1). Neste caso, o campo de consulta as bases locais deve estar habilitado e o campo de consultas as bases distribuídas desabilitada. A inserção dos dados é feita pela seleção dos botões do grupo: Inserir Dados na Tabela - fazendo com que sejam apresentados os volumes de dados a serem inseridos nas tabelas de cada site, e pela seleção do botão Inserir. As consultas são realizadas através da seleção dos botões dos grupos: Consultar Bases de Dados Locais - caso seja selecionado um site local, ou Consultar Base de Dados Distribuída - se selecionado o banco de dados distribuído. A figura 5.16 mostra a insersão dos dados no site 1, banco de dados Oracle. A figura 5.17 mostra a seleção do banco de dados distribuído. Neste caso, a consulta a base de dados locais e a inserção de dados fica desabilitada.

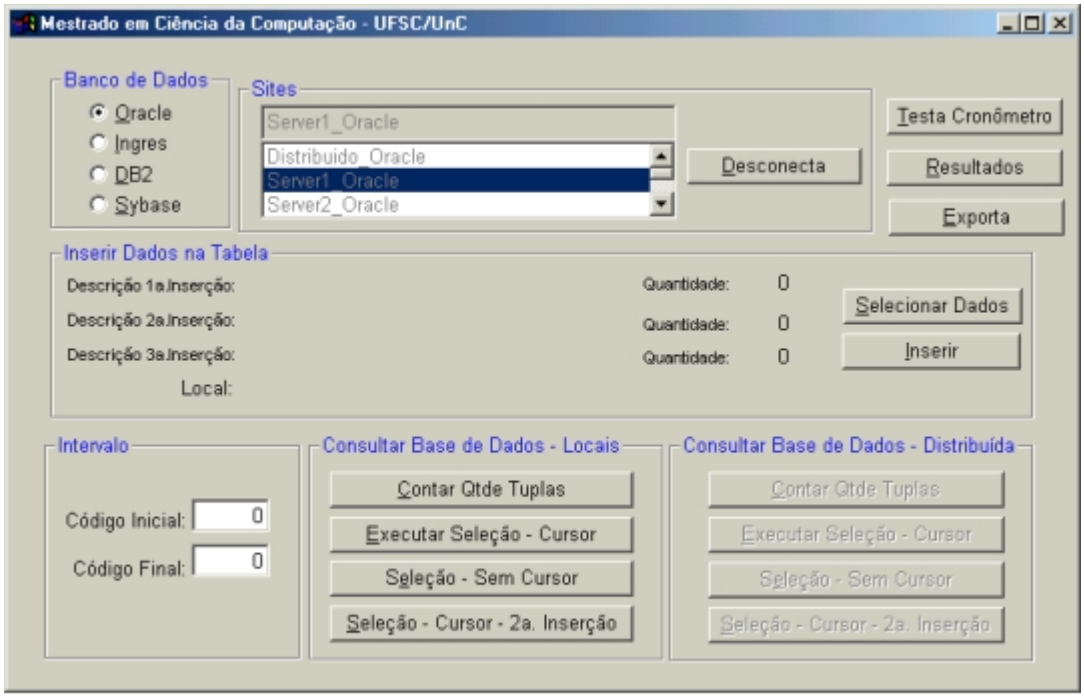


Figura 5.15. Seleção do SGBD Oracle no Site 1

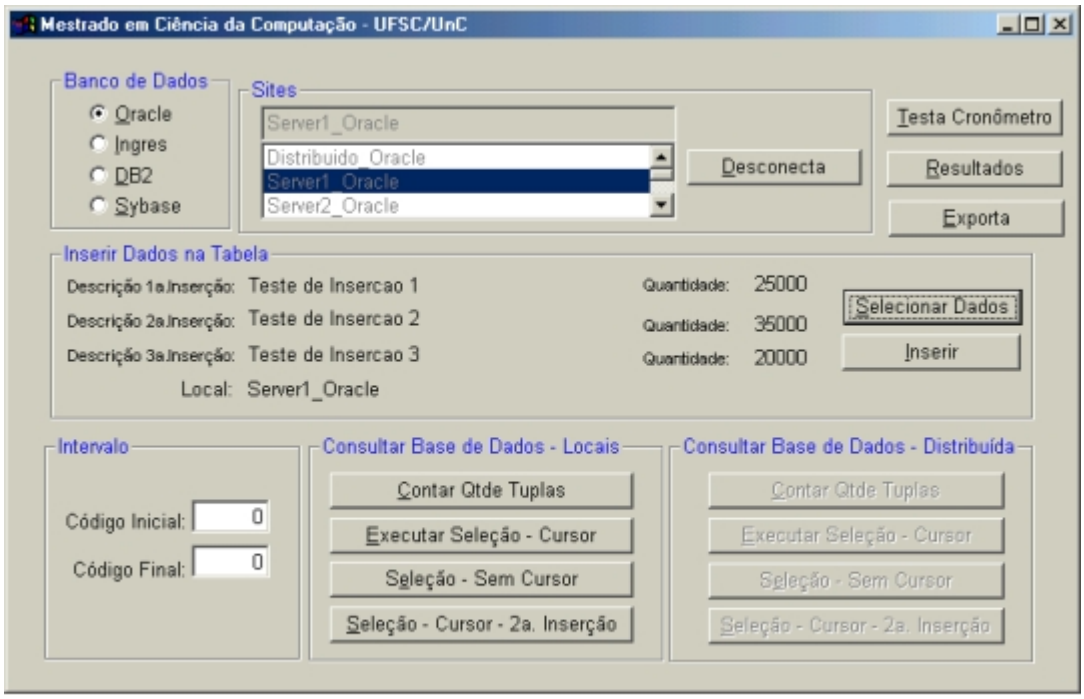


Figura 5.16. Inserção dos Dados no SGBD Oracle no Site 1

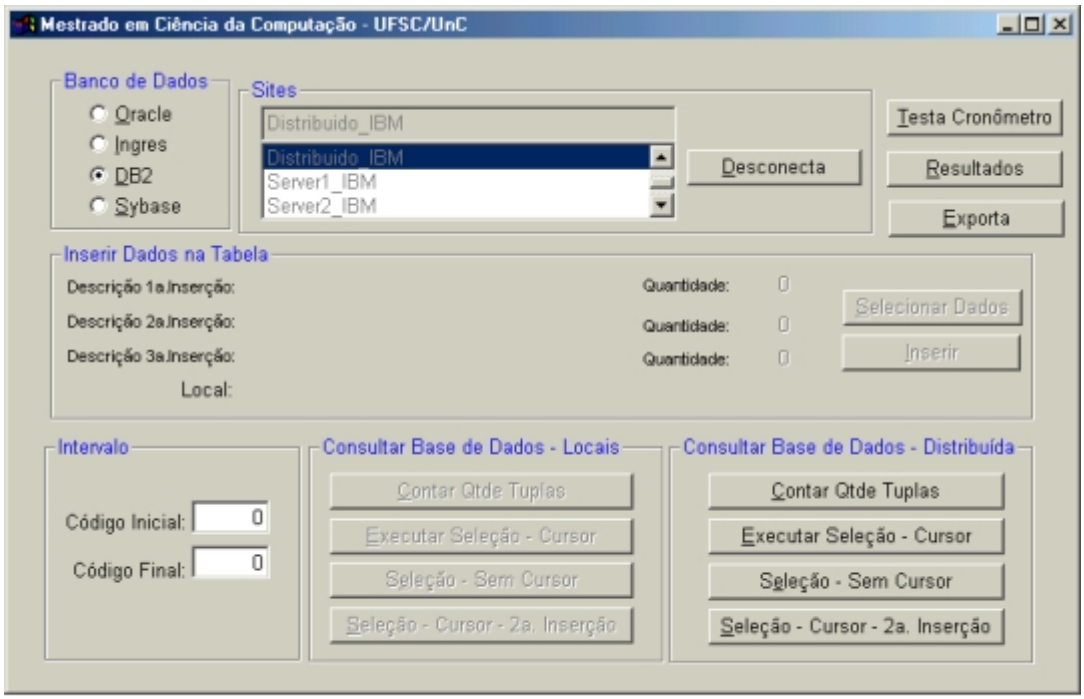


Figura 5.17. Seleção do Banco de Dados Distribuído - DB2

Os principais pontos do programa são apresentados abaixo.

1) Definição das tabelas do banco de dados. Duas definições foram incluídas neste programa:

   a. A primeira é a definição da tabela para acesso as bases de dados dos sites locais:

```
EXEC SQL INCLUDE TABELA END-EXEC.
    EXEC SQL DECLARE TABELA TABLE
    ( TABELA_COD            DECimal(6, 0)        NOT NULL
     ,TABELA_DESCR          CHARacter(40)        NOT NULL
     ,TABELA_LOCAL          CHARacter(20)        NOT NULL
     ,TABELA_INSERCAO       DECimal(3, 0)        NOT NULL
    ) END-EXEC.
******************************************************
 COBOL DECLARATION FOR TABLE TABELA
******************************************************
01  DCLTABELA.
    03 TABELA-COD                      PIC S9(06) COMP-3.
    03 TABELA-DESCR                    PIC X(40).
    03 TABELA-LOCAL                    PIC X(20).
    03 TABELA-INSERCAO                 PIC S9(03) COMP-3.
EXEC SQL INCLUDE TABELA1 END-EXEC.
    EXEC SQL DECLARE TABELA1 TABLE
    ( TABELA_COD            DECimal(6, 0)        NOT NULL
     ,TABELA_DESCR          CHARacter(40)        NOT NULL
     ,TABELA_LOCAL          CHARacter(20)        NOT NULL
     ,TABELA_INSERCAO       DECimal(3, 0)        NOT NULL
    ) END-EXEC.
```

   b. A segunda a é a definição da tabela para acesso as bases de dados dos sites remotos. A TABELA1 é a tabela definida para acesso ao banco de dados do site

1, a TABELA2 é a tabela para acesso a base de dados do site 2 e a TABELA3 é a tabela definida para acesso aos dados do site 3.

```
EXEC SQL INCLUDE TABELA1 END-EXEC.
    EXEC SQL DECLARE TABELA1 TABLE
    ( TABELA_COD            DECimal(6, 0)        NOT NULL
     ,TABELA_DESCR          CHARacter(40)        NOT NULL
     ,TABELA_LOCAL          CHARacter(20)        NOT NULL
     ,TABELA_INSERCAO       DECimal(3, 0)        NOT NULL
    ) END-EXEC.
************************************************************
 COBOL DECLARATION FOR TABLE TABELA1
************************************************************
01  DCLTABELA1.
    03 TABELA1-COD                      PIC S9(06) COMP-3.
    03 TABELA1-DESCR                    PIC X(40).
    03 TABELA1-LOCAL                    PIC X(20).
    03 TABELA1-INSERCAO                 PIC S9(03) COMP-3.
EXEC SQL INCLUDE TABELA2 END-EXEC.
    EXEC SQL DECLARE TABELA2 TABLE
    ( TABELA_COD            DECimal(6, 0)        NOT NULL
     ,TABELA_DESCR          CHARacter(40)        NOT NULL
     ,TABELA_LOCAL          CHARacter(20)        NOT NULL
     ,TABELA_INSERCAO       DECimal(3, 0)        NOT NULL
    ) END-EXEC.
************************************************************
 COBOL DECLARATION FOR TABLE TABELA2
************************************************************
01  DCLTABELA2.
    03 TABELA2-COD                      PIC S9(06) COMP-3.
    03 TABELA2-DESCR                    PIC X(40).
    03 TABELA2-LOCAL                    PIC X(20).
    03 TABELA2-INSERCAO                 PIC S9(03) COMP-3.
EXEC SQL INCLUDE TABELA3 END-EXEC.
    EXEC SQL DECLARE TABELA3 TABLE
    ( TABELA_COD            DECimal(6, 0)        NOT NULL
     ,TABELA_DESCR          CHARacter(40)        NOT NULL
     ,TABELA_LOCAL          CHARacter(20)        NOT NULL
     ,TABELA_INSERCAO       DECimal(3, 0)        NOT NULL
    ) END-EXEC.
************************************************************
 COBOL DECLARATION FOR TABLE TABELA3
************************************************************
01  DCLTABELA3.
    03 TABELA3-COD                      PIC S9(06) COMP-3.
    03 TABELA3-DESCR                    PIC X(40).
    03 TABELA3-LOCAL                    PIC X(20).
    03 TABELA3-INSERCAO                 PIC S9(03) COMP-3.
```

2) Conexão com o SGBD. A variável SGBD fornece o sistema de gerência de banco de dados a ser conectado - campo 1 da descrição do aplicativo. A variável DATABASE-TST fornece o site a ser acessado - fornecido pelo campo 2 da descrição do aplicativo. O usuário e senhas de acesso são definidos nos campos entre aspas.

a. Conexão ao banco de dados:

```
IF SGBD = 'O'
    EXEC SQL
     CONNECT TO :DATABASE-TST USER 'mestrado.mestrado001'
    END-EXEC.
 IF SGBD = 'I'
    EXEC SQL
```

```
    CONNECT TO :DATABASE-TST USER 'ingres.ingres001'
   END-EXEC.
IF SGBD = 'D'
   EXEC SQL
    CONNECT TO :DATABASE-TST USER 'db2inst1.db2001'
   END-EXEC.
IF SGBD = 'S'
   EXEC SQL
    CONNECT TO :DATABASE-TST USER 'sa.'
   END-EXEC.
```

b. Desconexão do banco de dados:

```
DESCONECTA-BD SECTION.
   EXEC SQL
     DISCONNECT CURRENT
END-EXEC.
```

3) Inserção dos dados na tabela:

```
EXEC SQL
   SELECT MAX(A.TABELA_COD) INTO :TABELA-COD
    FROM TABELA A
END-EXEC.
EXEC SQL
   SELECT MAX(A.TABELA_INSERCAO) INTO:TABELA-INSERCAO
    FROM TABELA A
END-EXEC.
EXEC SQL
   INSERT INTO TABELA (TABELA_COD, TABELA_DESCR, TABELA_LOCAL,
      ,TABELA_INSERCAO) VALUES (:TABELA-COD, :TABELA-DESCR,
      :TABELA-LOCAL, :TABELA-INSERCAO)
END-EXEC.
```

4) Operação Count no banco de dados local:

```
EXEC SQL
   SELECT count(A.TABELA_COD) INTO :TABELA-COD
    FROM TABELA A WHERE ( A.TABELA_COD >= :TABELA-COD-INI )
       AND  ( A.TABELA_COD <= :TABELA-COD-FIM )
END-EXEC.
```

5) Seleção com cursor no banco de dados local:

```
EXEC SQL
   DECLARE CSR2 CURSOR FOR SELECT A.TABELA_COD, A.TABELA_DESCR,
     A.TABELA_LOCAL, A.TABELA_INSERCAO
    FROM TABELA A WHERE ( A.TABELA_COD >= :TABELA-COD-INI )
         AND  ( A.TABELA_COD <= :TABELA-COD-FIM )
    ORDER BY A.TABELA_COD
END-EXEC.
EXEC SQL OPEN CSR2 END-EXEC.
PERFORM UNTIL SQLCODE < 0 OR SQLCODE = +100
    EXEC SQL
        FETCH CSR2 INTO :TABELA-COD, :TABELA-DESCR, :TABELA-LOCAL
          ,:TABELA-INSERCAO
    END-EXEC
    IF SQLCODE = ZEROS
       add 1 to ds-quantidade
    END-IF
END-PERFORM.
EXEC SQL CLOSE CSR2 END-EXEC.
```

6) Seleção sem cursor no banco de dados local:

```
EXEC SQL
   SELECT  A.TABELA_COD,A.TABELA_DESCR,A.TABELA_LOCAL,
      A.TABELA_INSERCAO INTO :TABELA-COD,:TABELA-DESCR,:TABELA-LOCAL
       ,:TABELA-INSERCAOFROM TABELA A
```

```
        WHERE ( A.TABELA_COD >= :TABELA-COD-INI )
           AND  ( A.TABELA_COD <= :TABELA-COD-FIM )
        ORDER BY A.TABELA_DESCR
    END-EXEC.
```

7) Seleção com cursor no banco de dados locais, selecionando somente os dados inseridos na segunda inserção:

```
    EXEC SQL
        DECLARE CSR3 CURSOR FOR SELECT A.TABELA_COD,A.TABELA_DESCR,
             ,A.TABELA_LOCAL,A.TABELA_INSERCAO FROM TABELA A
         WHERE ( A.TABELA_COD >= :TABELA-COD-INI )
          AND  ( A.TABELA_COD <= :TABELA-COD-FIM )
          AND  ( A.TABELA_INSERCAO = :INSERCAO-TST )
         ORDER BY
             A.TABELA_COD
       END-EXEC.
    EXEC SQL OPEN CSR3 END-EXEC.
       PERFORM UNTIL SQLCODE < 0 OR SQLCODE = +100
         EXEC SQL
           FETCH CSR3 INTO :TABELA-COD,:TABELA-DESCR,:TABELA-LOCAL,
            ,:TABELA-INSERCAO
         END-EXEC
         IF SQLCODE = ZEROS
            add 1 to ds-quantidade
         END-IF
       END-PERFORM.
    EXEC SQL CLOSE CSR3 END-EXEC.
```

8) Operação count no banco de dados distribuído:

```
EXEC SQL
     SELECT count(A.TABELA_COD) INTO :TABELA1-COD
        FROM TABELA1 A WHERE ( A.TABELA_COD >= :TABELA-COD-INI )
            AND  ( A.TABELA_COD <= :TABELA-COD-FIM )
END-EXEC.
IF SQLCODE NOT = ZEROS
    MOVE 'Erro consulta Base 1' TO MENSAGEM
    MOVE ZEROS TO DS-HORARIO-INICIO DS-HORARIO-FIM
       DS-QUANTIDADE
    GO TO COUNT-DISTRIBUIDO-F.
EXEC SQL
   SELECT count(B.TABELA_COD) INTO :TABELA2-COD FROM TABELA2 B
        WHERE ( B.TABELA_COD >= :TABELA-COD-INI )
        AND  ( B.TABELA_COD <= :TABELA-COD-FIM )
END-EXEC.
IF SQLCODE NOT = ZEROS
   MOVE 'Erro consulta Base 2' TO MENSAGEM
   MOVE ZEROS TO DS-HORARIO-INICIO DS-HORARIO-FIM DS-QUANTIDADE
   GO TO COUNT-DISTRIBUIDO-F.
EXEC SQL
   SELECT count(C.TABELA_COD) INTO :TABELA3-COD FROM TABELA3 C
       WHERE ( C.TABELA_COD >= :TABELA-COD-INI )
       AND  ( C.TABELA_COD <= :TABELA-COD-FIM )
       END-EXEC.
```

9) Seleção com cursor no banco de dados distribuído:

```
EXEC SQL
      DECLARE CSR1 CURSOR FOR
         SELECT A.TABELA_COD,A.TABELA_DESCR,A.TABELA_LOCAL,
                A.TABELA_INSERCAO FROM TABELA1 A
            WHERE ( A.TABELA_COD >= :TABELA-COD-INI )
            AND  ( A.TABELA_COD <= :TABELA-COD-FIM )
          UNION  SELECT B.TABELA_COD,B.TABELA_DESCR,B.TABELA_LOCAL,
                 ,B.TABELA_INSERCAOFROM TABELA2 B
```

```
                WHERE ( B.TABELA_COD >= :TABELA-COD-INI )
                AND  ( B.TABELA_COD <= :TABELA-COD-FIM )
              UNION SELECT C.TABELA_COD,C.TABELA_DESCR,C.TABELA_LOCAL
                    ,C.TABELA_INSERCAO FROM TABELA3 C
                   WHERE ( C.TABELA_COD >= :TABELA-COD-INI )
                   AND  ( C.TABELA_COD <= :TABELA-COD-FIM )
   END-EXEC.
   PERFORM PEGA-TEMPO.
   MOVE TEMPO-ED-R TO DS-HORARIO-FIM.
   IF SQLCODE NOT = ZEROS
        MOVE 'Erro seleção' TO MENSAGEM
        MOVE ZEROS TO DS-HORARIO-INICIO DS-HORARIO-FIM
        GO TO SELECT-DISTRIBUIDO-F.
   EXEC SQL OPEN CSR1 END-EXEC.
   IF SQLCODE NOT = ZEROS
        MOVE 'Erro seleção Cursor' TO MENSAGEM
        MOVE ZEROS TO DS-HORARIO-INICIO DS-HORARIO-FIM
        GO TO SELECT-DISTRIBUIDO-F.
   PERFORM UNTIL SQLCODE < 0 OR SQLCODE = +100
     EXEC SQL FETCH CSR1 INTO :TABELA-COD,:TABELA-DESCR,:TABELA-LOCAL
        ,:TABELA-INSERCAO
     END-EXEC
     IF SQLCODE = ZEROS
        add 1 to ds-quantidade
     END-IF
   END-PERFORM.
```

10) Seleção sem cursor no banco de dados distribuído:

```
EXEC SQL
    SELECT A.TABELA_COD,A.TABELA_DESCR,A.TABELA_LOCAL,A.TABELA_INSERCAO
           FROM TABELA1 A  WHERE (A.TABELA_COD >= :TABELA-COD-INI)
              AND  (A.TABELA_COD <= :TABELA-COD-FIM)
    UNION SELECT B.TABELA_COD,B.TABELA_DESCR,B.TABELA_LOCAL,
       B.TABELA_INSERCAOFROM TABELA2 B
           WHERE (B.TABELA_COD >= :TABELA-COD-INI)
           AND  (B.TABELA_COD <= :TABELA-COD-FIM)
    UNION SELECT C.TABELA_COD,C.TABELA_DESCR,C.TABELA_LOCAL
      ,C.TABELA_INSERCAO FROM TABELA3 C
           WHERE (C.TABELA_COD >= :TABELA-COD-INI)
          AND  (C.TABELA_COD <= :TABELA-COD-FIM)
END-EXEC.
```

11) Seleção com cursor no banco de dados distribuído, selecionando os dados inseridos na segunda inserção:

```
          EXEC SQL
            DECLARE CSR4 CURSOR FOR
            SELECT A.TABELA_COD,A.TABELA_DESCR,A.TABELA_LOCAL
                  ,A.TABELA_INSERCAO FROM TABELA1 A
             WHERE ( A.TABELA_COD >= :TABELA-COD-INI )
               AND  ( A.TABELA_COD <= :TABELA-COD-FIM )
               AND  ( A.TABELA_INSERCAO = :INSERCAO-TST )
            UNION SELECT B.TABELA_COD,B.TABELA_DESCR,B.TABELA_LOCAL
                        ,B.TABELA_INSERCAO FROM TABELA2 B
            WHERE ( B.TABELA_COD >= :TABELA-COD-INI )
              AND  ( B.TABELA_COD <= :TABELA-COD-FIM )
              AND  ( B.TABELA_INSERCAO = :INSERCAO-TST )
           UNION SELECT C.TABELA_COD,C.TABELA_DESCR,C.TABELA_LOCAL
                         ,C.TABELA_INSERCAO FROM TABELA3 C
             WHERE ( C.TABELA_COD >= :TABELA-COD-INI )
               AND  ( C.TABELA_COD <= :TABELA-COD-FIM )
               AND  ( C.TABELA_INSERCAO = :INSERCAO-TST )
           END-EXEC.
           PERFORM PEGA-TEMPO.
           MOVE TEMPO-ED-R TO DS-HORARIO-FIM.
           IF SQLCODE NOT = ZEROS
```

```
    MOVE 'Erro seleção' TO MENSAGEM
    MOVE ZEROS TO DS-HORARIO-INICIO DS-HORARIO-FIM
    GO TO SELECT-INSERCAO-DISTRIBUIDO-F.
EXEC SQL OPEN CSR4 END-EXEC.
PERFORM UNTIL SQLCODE < 0 OR SQLCODE = +100
  EXEC SQL
  FETCH CSR4 INTO :TABELA-COD,:TABELA-DESCR,:TABELA-LOCAL
                  ,:TABELA-INSERCAO
  END-EXEC
  IF SQLCODE = ZEROS
     add 1 to ds-quantidade
  END-IF
END-PERFORM.
EXEC SQL CLOSE CSR1 END-EXEC.
```

5.1.6 Análise de Desempenho dos SGBDD

A tabela 5.1 apresenta o tempo de execução das consultas, as tonalidades de fundo mais claras indicam os menores tempos, conforme legenda da tabela. As figuras 5.18, 5.19, 5.20 e 5.21 demonstram os resultados de cada teste.

Tabela 5.1. Resultados dos Testes

Melhores Resultado	1º	2º	3º	4º
SGBD	Tempo	Tempo	Tempo	Tempo
Count				
Sybase	00:00:08:24	00:00:11:04	00:00:11:15	00:00:11:26
IBM	00:00:23:07	00:00:23:13	00:00:23:35	00:00:25:43
Oracle	00:00:02:97	00:00:05:44	00:00:05:65	00:00:06:20
Ingres	00:00:09:34	00:00:09:39	00:00:10:49	00:00:11:04
Select com Cursor				
Sybase	00:03:02:57	00:03:35:25	00:03:36:41	00:03:51:95
IBM	00:01:17:39	00:01:17:45	00:01:19:75	00:01:20:25
Oracle	00:03:44:64	00:03:45:14	00:03:45:58	00:03:53:16
Ingres	00:06:08:50	00:06:11:08	00:06:25:85	00:06:27:94
Select sem Cursor				
Sybase	00:03:07:85	00:03:16:30	00:03:23:06	00:03:23:06
IBM	00:00:40:87	00:00:41:19	00:00:42:29	00:00:42:62
Oracle	00:00:17:79	00:00:18:07	00:00:18:29	00:00:18:29
Ingres	00:01:24:14	00:01:24:80	00:01:26:07	00:01:27:94
Select 2ª Inserção				
Sybase	00:00:52:17	00:01:09:43	00:01:17:61	00:01:19:70
IBM	00:00:24:66	00:00:25:54	00:00:25:59	00:00:25:87
Oracle	00:01:26:35	00:01:28:26	00:01:28:60	00:01:30:19
Ingres	00:02:19:08	00:02:21:60	00:02:25:17	00:02:27:25

1º 2º 3º 4º Melhor tempo

SGBD		Count	Select c/Cursor	Select sem Cursor	Select 2ª Inserção
Distribuido_IBM	Horário Início:	15.50.01.45	15.50.43.63	15.52.29.09	15.53.30.33
	Horário Término:	15.50.24.80	15.52.01.08	15.53.11.38	15.53.55.87
	Tempo Decorrido:	0.00.23.35	0.01.17.45	0.00.42.29	0.00.25.54
Distribuido_Ingres	Horário Início:	17.47.28.67	17.48.00.91	17.54.52.08	17.56.34.30
	Horário Término:	17.47.39.71	17.54.11.99	17.56.20.02	17.58.55.90
	Tempo Decorrido:	0.00.11.04	0.06.11.08	0.01.27.94	0.02.21.60
Distribuido_Oracle	Horário Início:	15.58.10.56	16.43.36.90	16.48.26.69	16.49.20.07
	Horário Término:	15.58.13.53	16.47.30.06	16.48.44.48	16.50.50.26
	Tempo Decorrido:	0.00.02.97	0.03.53.16	0.00.17.79	0.01.30.19
Distribuido_Sybase	Horário Início:	17.12.43.97	17.33.10.90	17.37.45.91	17.41.40.49
	Horário Término:	17.12.55.23	17.36.13.47	17.41.02.21	17.43.00.19
	Tempo Decorrido:	0.00.11.26	0.03.02.57	0.03.16.30	0.01.19.70

Figura 5.18. Resultados do 1º Teste

SGBD		Count	Select c/Cursor	Select sem Cursor	Select 2ª Inserção
Distribuido_IBM	Horário Início:	15.49.17.68	15.49.55.41	15.51.29.61	15.52.33.82
	Horário Término:	15.49.43.11	15.51.12.80	15.52.10.80	15.52.58.48
	Tempo Decorrido:	0.00.25.43	0.01.17.39	0.00.41.19	0.00.24.66
Distribuido_Ingres	Horário Início:	15.33.22.30	15.33.54.16	15.40.36.16	15.42.34.14
	Horário Término:	15.33.32.79	15.40.20.01	15.42.00.96	15.44.59.31
	Tempo Decorrido:	0.00.10.49	0.06.25.85	0.01.24.80	0.02.25.17
Distribuido_Oracle	Horário Início:	14.04.07.78	14.29.59.87	14.35.01.02	14.35.52.05
	Horário Término:	14.04.13.22	14.33.44.51	14.35.19.09	14.37.20.31
	Tempo Decorrido:	0.00.05.44	0.03.44.64	0.00.18.07	0.01.28.26
Distribuido_Sybase	Horário Início:	14.39.36.36	14.39.57.84	15.25.36.59	15.24.18.49
	Horário Término:	14.39.47.40	14.43.34.25	15.28.59.65	15.25.10.66
	Tempo Decorrido:	0.00.11.04	0.03.36.41	0.03.23.06	0.00.52.17

Figura 5.19. Resultados do 2º Teste

SGBD		Count	Select c/Cursor	Select sem Cursor	Select 2ª Inserção
Distribuido_IBM	Horário Início:	17.04.52.54	17.06.31.74	17.08.48.45	17.10.44.18
	Horário Término:	17.05.15.67	17.07.51.49	17.09.31.07	17.11.09.77
	Tempo Decorrido:	0.00.23.13	0.01.19.75	0.00.42.62	0.00.25.59
Distribuido_Ingres	Horário Início:	16.44.21.17	16.45.16.86	16.52.07.05	16.53.56.90
	Horário Término:	16.44.30.51	16.51.44.80	16.53.33.12	16.56.24.15
	Tempo Decorrido:	0.00.09.34	0.06.27.94	0.01.26.07	0.02.27.25
Distribuido_Oracle	Horário Início:	17.14.15.92	17.15.39.35	17.20.18.81	17.21.23.62
	Horário Término:	17.14.21.57	17.19.24.93	17.20.37.10	17.22.52.22
	Tempo Decorrido:	0.00.05.65	0.03.45.58	0.00.18.29	0.01.28.60
Distribuido_Sybase	Horário Início:	15.55.51.27	16.22.26.75	16.34.59.06	16.38.42.83
	Horário Término:	15.55.59.51	16.26.18.70	16.38.06.91	16.39.52.26
	Tempo Decorrido:	0.00.08.24	0.03.51.95	0.03.07.85	0.01.09.43

Figura 5.20. Resultados do 3º Teste

SGBD		Count	Select c/Cursor	Select sem Cursor	Select 2ª Inserção
Distribuido_IBM	Horário Início:	17.55.54.64	17.56.50.28	17.58.26.29	17.59.35.99
	Horário Término:	17.56.17.71	17.58.10.53	17.59.07.16	18.00.01.86
	Tempo Decorrido:	0.00.23.07	0.01.20.25	0.00.40.87	0.00.25.87
Distribuido_Ingres	Horário Início:	17.36.52.69	17.37.56.78	17.44.46.86	17.47.01.64
	Horário Término:	17.37.02.08	17.44.05.28	17.46.11.00	17.49.20.72
	Tempo Decorrido:	0.00.09.39	0.06.08.50	0.01.24.14	0.02.19.08
Distribuido_Oracle	Horário Início:	18.02.34.83	18.03.01.52	18.07.15.06	18.07.56.03
	Horário Término:	18.02.41.03	18.06.46.66	18.07.33.35	18.09.22.38
	Tempo Decorrido:	0.00.06.20	0.03.45.14	0.00.18.29	0.01.26.35
Distribuido_Sybase	Horário Início:	18.11.28.21	18.12.00.78	18.16.01.35	18.19.49.73
	Horário Término:	18.11.39.36	18.15.36.03	18.19.24.41	18.21.07.34
	Tempo Decorrido:	0.00.11.15	0.03.35.25	0.03.23.06	0.01.17.61

Figura 5.21. Resultados do 4º Teste

A tabela 5.2 classifica os tempos apresentando o SGBD que realizou a consulta em menor tempo para cada teste.

Tabela 5.2. SGBD com Melhor Desempenho

Classificação por Teste				
SGBD	1º Teste	2º Teste	3º Teste	4º Teste
Count	Oracle	Oracle	Oracle	Oracle
Select com Cursor	DB2	DB2	DB2	DB2
Select sem Cursor	Oracle	Oracle	Oracle	Oracle
Select na 2ª Inserção	DB2	DB2	DB2	DB2

5.1.7 Conclusões da Análise 1

A primeira análise investiga as propriedades de transparência de localização e transparência de nomeação dos bancos de dados distribuídos, observou-se:

1. Para obter a propriedade de transparência de localização os bancos de dados implementam:
   a. A comunicação entre os sites ou entre os SGBD locais é feita por meio do registro dos servidores. Neste estudo, foi necessário registrar os três bancos de dados locais no banco de dados distribuído, de modo a obter acesso as bases de dados remotas;
   b. Para registrar os SGBDs é necessário interligar os bancos de dados. A interligação é feita através de database links ou ligações de banco de dados;
   c. Os database links utilizam o protocolo TCP/IP e os sockets do TCP/IP. Assim, na configuração da ligação entre os bancos de dados é informado o nome do site - se cadastrado do arquivo /etc/hosts, ou o endereço IP do site remoto, o nome do serviço associado a uma porta do TCP/IP e um nome para a conexão. No banco de dados DB2 as portas utilizadas são a 50000 e 50001, no Sybase a porta 4100, no Oracle a porta 1521 e no Ingres a porta 1524;
2. Para obter a propriedade de transparência de nomeação, as tabelas devem ser registradas no banco de dados. O registro é através da criação de um sinônimo ou um apelido para a tabela. Tendo o SGBD a função de associar o nome atribuído com a localização e o nome da tabela no site remoto;

3. A análise dos resultados das consultas realizadas sobre a base de dados distribuída teve como resultado que o SGBD Oracle foi o que teve melhor desempenho na realização de consultas utilizando a função Count e a operação de seleção sem o uso de cursores do banco de dados. O SGBD DB2 teve maior desempenho na realização de consultas de seleção com o uso de cursores e na seleção dos dados que atendiam a condição de haverem sido inseridos na segunda inserção.

Os testes de realização de consultas distribuídas e da implementação da função count tem como resultados:

a) A implementação da função count, sobre 330.000 registros, apresentou o SGBD Oracle com o melhor desempenho;
b) A realização de uma seleção com o uso de cursores, sobre 330.000 registros, teve como o SGBD DB/2 como o de melhor desempenho;
c) A realização de uma seleção sem o uso de cursores, sobre 330.000 registros, teve o SGBD Oracle como o de melhor desempenho;
d) A realização de uma operação de seleção com o uso de cursor, sobre os dados inseridos na 2ª inserção - 130.000 registros, teve o SGBD DB/2 como o de melhor desempenho.

A tabela 5.3 apresenta um resumo dos comandos para configurar iniciar a execução e administrar os sistema de gerência de banco de dados.

Tabela 5.3. Comandos Básicos dos SGBD

Tarefa	DB2	Ingres	Oracle	Sybase
Iniciar SGBD	Db2 start Db2admin start	ingstart	Startdb Lsnctrl start	Startserver RUN_SYBASE
Sql interarivo	db2	isql	sqlplus	Isql -Usa -SserverN
Criar banco de dados	Create database	Createdb	dbassist	Create database
Conectar ao banco de dados	Connect to	Isql <base de dados>	Connect usuário senha	Use
Criar tabela	Create table	Create table	Create table	Create table

A tabela 5.4 apresenta um resumo dos comandos para configurar um ambiente de distribuição de dados.

Tabela 5.4. Comandos para Configurar um SGBD Distribuído

Tarefa	DB2	Ingres	Oracle	Sybase
Criar banco dados	Create database	Createdb /star	dbassist	Create database Habilitar como distribuído sp_configure "enable CIS", 1
Catalogar sites	Catalog TCPIP node	Utilitário netutil - virtual node name	Aplicativo Visual Studio - database link	sp_addserver
Método acesso	Create wrapper	Utilitário netutil - connection entry	Utilitário net8 - nomeação de serviço	Arquivo interfaces no diretório /opt/sybase
Registrar servidores	Create server	Realizado quando do catálogo do site	Aplicativo visual studio - criar vínculos	Realizado quando do catálogo do site
Mapear usuários	Create user mapping	Realizado quando do catálogo do site	Não	Não
Registrar tabelas	Create nickname	Register .. As link	Aplicativo visual studio - criar sinônimos	Create existing table

5.2 Análise 2 - Transparência de Replicação

A segunda análise avalia um ambiente de banco de dados distribuído replicado. O ambiente de testes implementado, mostrado na figura 5.22, é constituído por:

- Dois sites, contendo SGBDs autônomos e independentes interligados por uma rede ethernet de 100 Mb/s. Nos site é criado uma base de dados e uma tabela, sendo a tabela do site 1 deve ser replicada para o site 2. O nome das bases de dados são: *server1* - para a base de dados do servidor server1.edu.br e *server2* - para a base de dados do servidor server2.edu.br;
- A tabela criada é baseada na estrutura da tabela utilizada no capítulo 2 página 28 do livro Principles of Distributed Database Systems, segunda edição, descrita no referencial bibliográfico como ÖZSU & VALDURIEZ, 1999. Esta base de dados deve ser configurada de modo a realizar a replicação dos dados para o outro site. A tabela é compostas pelos atributos:
    a. ENO: Definido como um campo alfabético de dois caracteres;
    b. ENAME: Campo alfanumérico (char) de 30 caracteres;
    c. TITLE: Campo alfanumérico (char) de 20 caracteres;
    d. SAL: Campo númerico (decimal) de sete caracteres;
    e. PNO: Campo alfanumérico (char) de dois caracteres;
    f. RESP: Campo alfanumérico (char) de vinte caracteres;

g. DUR: Campo numérico (decimal) de dois caracteres;

- Uma estação cliente, com o SO Windows 98, onde é instalado o software de administração dos SGBD.

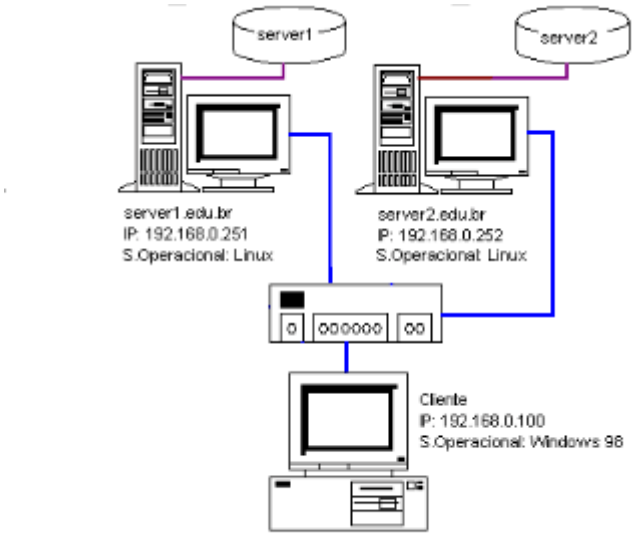


Figura 5.22. Ambiente da Análise nº 2

O teste consiste na realização das etapas descritas abaixo:

1. Configuração dos bancos de dados locais, com a criação da tabela no site 1;
2. Configuração do banco de dados do site 1 para replicar os dados para o site 2;
3. Inserção de dados no site 1 e verificação da replicação para o site2;
4. Avaliação dos resultados obtidos.

5.2.1 SGBD DB2

A configuração realizada no sistema de gerência de banco de dados DB2 de modo a implementar um ambiente de replicação partiu do princípio de que o SGBD e a base de dados de dados estão instalados e criados, processo realizado na análise nº 1. O ambiente de replicação a ser implementado é do tipo update anywhere, com os dados passiveis de serem atualizados em ambos os sites. As configuração específicas para a replicação dos dados são:

- Criar a tabela emp_server1 no site 1, para isto, acessar o monitor sql, digitando db2 e, a partir do prompt digitar db2 connect to server1;
- Criar a tabela com o comando: create table EMP_SERVER1 (ENO char(2) not null, ENAME char(30) not null, TITLE char(20) not null, SAL dec(7,0) not null,

PNO char(2) not null, RESP char(20) not null, DUR dec(2,0) not null, primary key (ENO, PNO));

- Catalogar um database link para conectar as bases de dados. Deve ser implementado em ambos os sites, com os comandos:
  1) No site 1: catalog TCPIP node server2 remote 192.168.0.252 server db2inst1 e catalog local node server1. A figura 5.23 mostra os nodos do site 1;

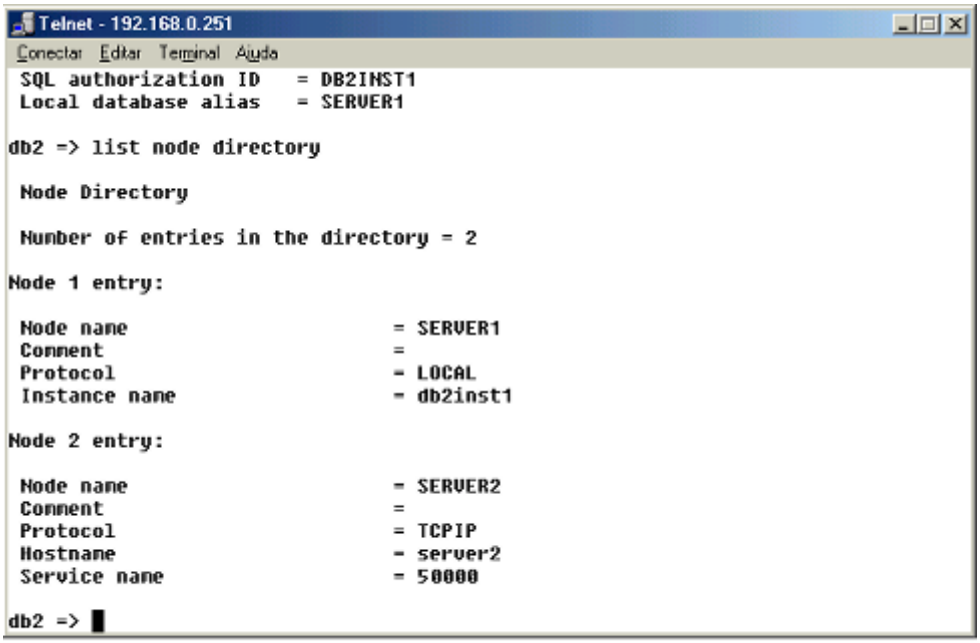


Figura 5.23. Nodos do SGBD DB2 no Site 1

  2) No site 2: catalog TCPIP node server1 remote 192.168.0.251 server db2inst1 e catalog local node server2. A figura 5.24 mostra os nodos do site 2;
- Catalogar o banco de dados remotos nos sites, com os comandos:
  1) No site 1: catalog database server2 at node server2;
  2) No site 2: catalog database server1 at node server1;
- Através do aplicativo Control Center da estação windows, configurar a replicação da tabela criando o replication source e o replication subscriber:
- Através do aplicativo Control Center da estação windows, configurar a replicação da tabela criando o replication source e o replication subscriber:
  1) No site 1 criar o replication source. A figura 5.25 mostra o replication source. A configuração é realizada definindo-se um nome - EMP_SERVER1, o schema - DB2INST1, a tabela a replicar - EMP_SERVER1 e o tipo de

detecção de conflito para o replication source - selecionado como standard. Também deve ser selecionado o uso do recurso capture before image - não selecionado, a figura 5.26 apresenta a configuração;

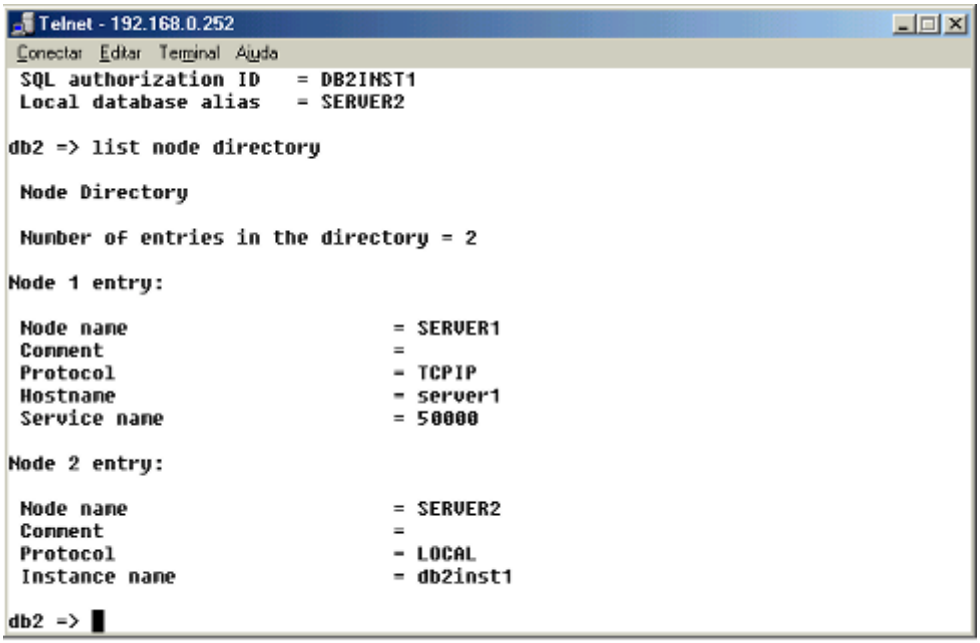


Figura 5.24. Nodos do SGBD DB2 no Site 2

2) Definir o replication subscription no site1 - mostrado na figura 6.27. As configurações realizadas implicam: na definição do programa de aplicação (apply qualifier) - EMP_QUAL; do nome da tabela destino, sendo mantido o nome da tabela de origem - EMP_SERVER1, demonstrada na figura 5.28; do tipo da tabela destino - selecionada como réplica, demonstrado na figura 5.29; e das colunas a serem replicadas para a tabela destino - foram selecionadas todas as colunas, conforme demonstrado na figura 5.30.
3) Definir se a replicação se dará por evento ou por intervalo de tempo. A replicação é configurada como sendo por intervalo de tempo e, como a replicação é do tipo update anywhere, devem ser definidos os períodos de tempo para cópia da origem para a réplica - demonstrado na figura 5.31, e da réplica para a tabela de origem - demonstrado na figura 5.32. Após realizadas estas configurações, o SGDB DB2 copia o replication source, replication subscription e a tabela EMP_SERVER1 para o site 2, conforme demonstrado nas figura 5.33, 5.34;

4) Iniciar o programa de captura dos dados e o programa de aplicação. Ambos os programas são iniciados no servidor 1, com os comandos:
   a. Programa de captura: db2 connect to server e asnccp server1 > /dev/null&. Sendo server1 o servidor que controla o processo de replicação;
   b. Programa de aplicação: asnapply EMP_QUAL SERVER1 > /dev/null&. Sendo que SERVER1 é o servidor de origem - que executa o programa de captura, e EMP_QUAL o nome do programa de aplicação definido na configuração da replicação.

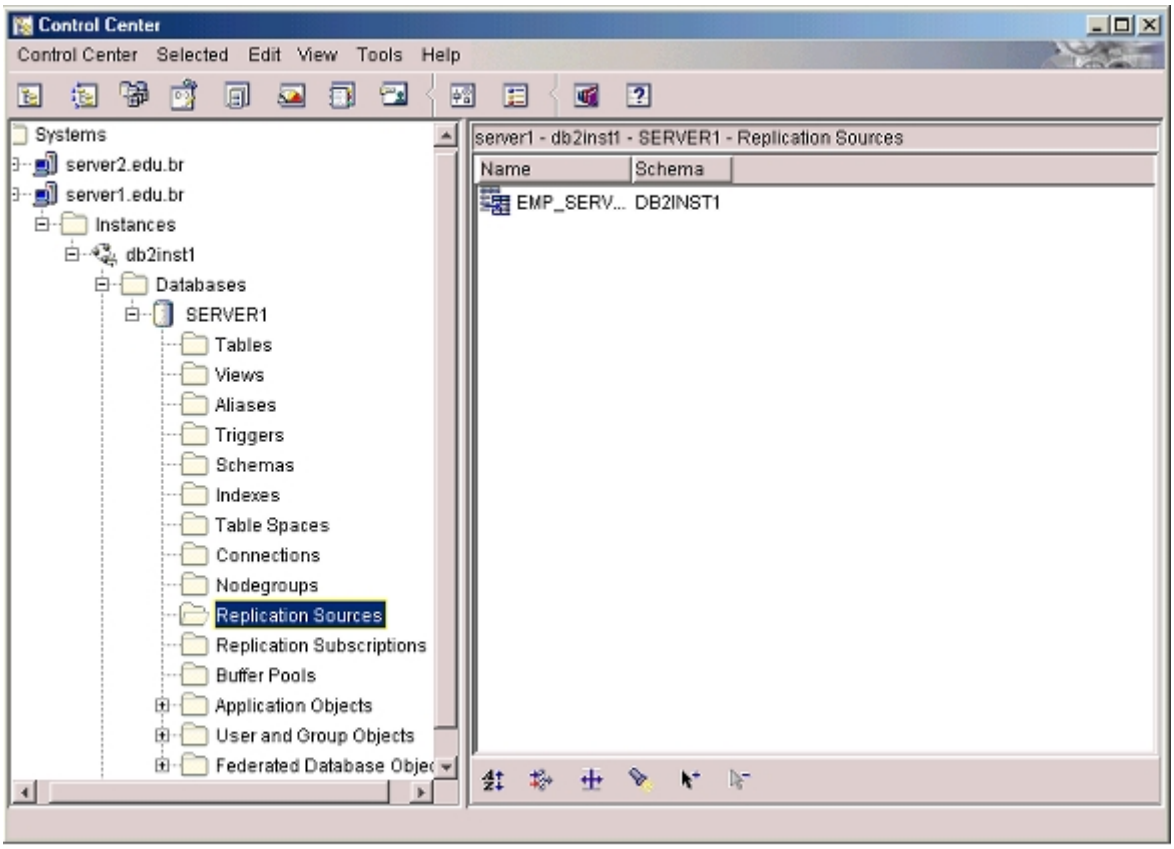


Figura 5.25. Replication Source no Site 1

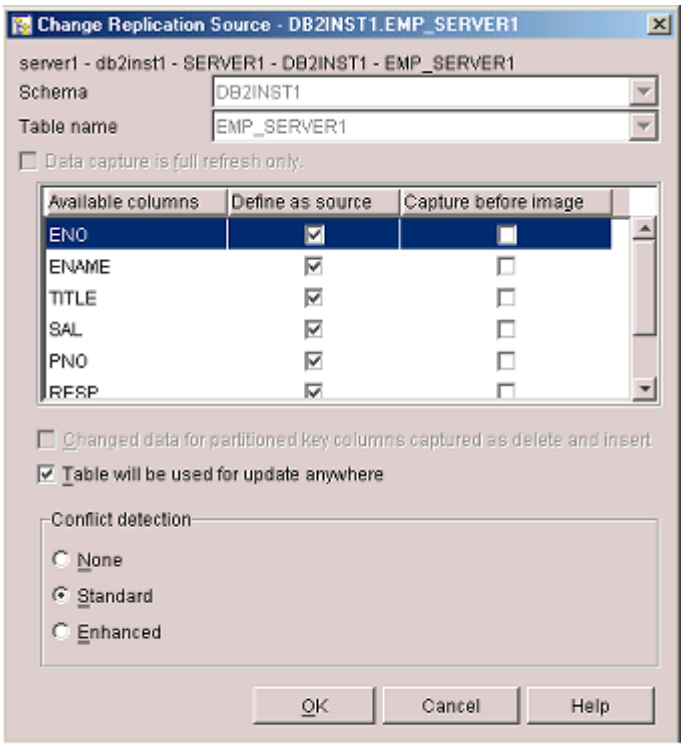


Figura 5.26. Definição da Tabela a Replicar no SGBD DB2 no Site 1

Para testar a replicação dos dados, foram inseridos um conjunto de dados no site 1 e verificado se os mesmos eram replicados para o site 2. A figura 5.35 mostra as operações de seleção e inserção no site 1 e a figura 5.36 as operações de seleção no site 2. A figura 5.35 mostra que foi realizada inicialmente uma operação de seleção sobre a tabela EMP_SERVER1 no site 1 e que traz como resultado a existência de duas linhas. Da mesma forma, a figura 5.36 mostra a execução de uma operação de seleção, apresentando como resultado as mesmas duas linhas. A figura 5.35 mostra, também, a realização da operação de inserção e de uma nova operação de seleção, apresentando como resultado a existência de três linhas. Uma nova operação de seleção é realizada no site 2 - demonstrada na figura 5.36, e que apresenta como resultado as mesmas três linhas existentes no site 1.

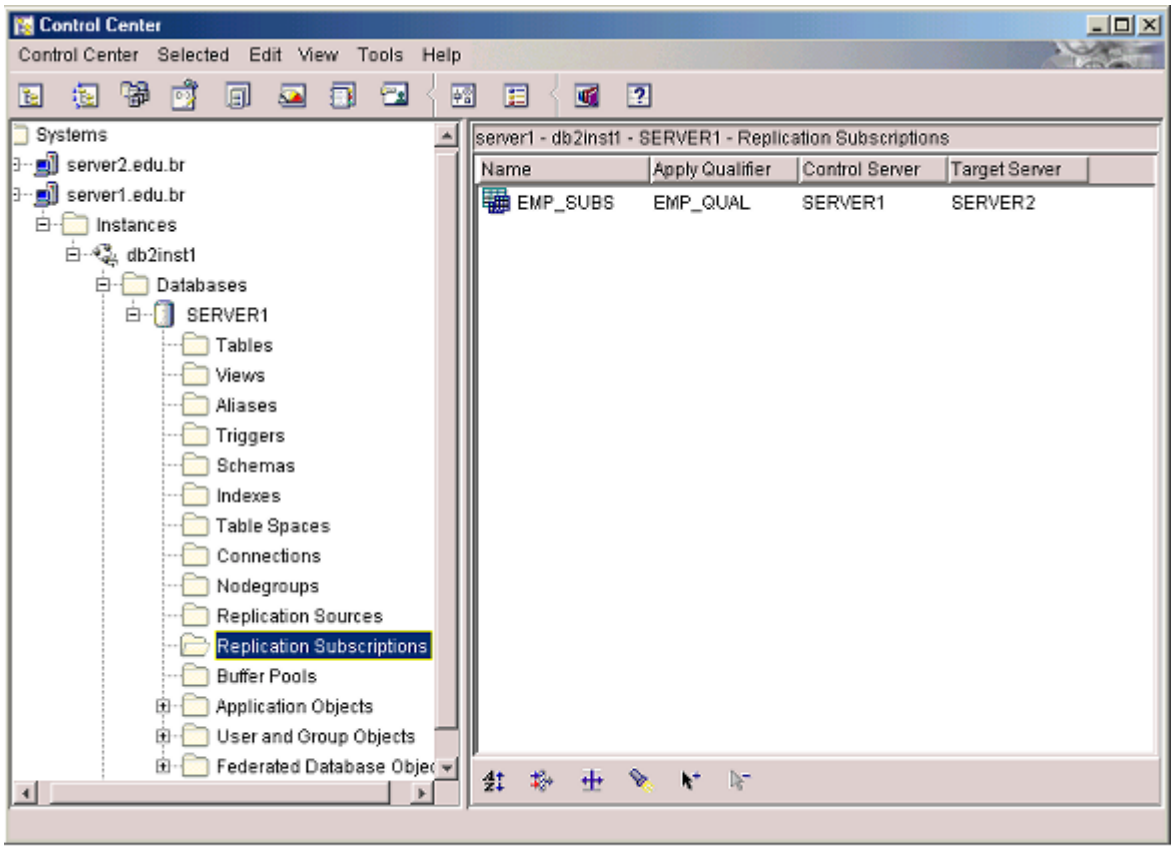


Figura 5.27. Replication Subscriptions no SGBD DB2 no Site 1

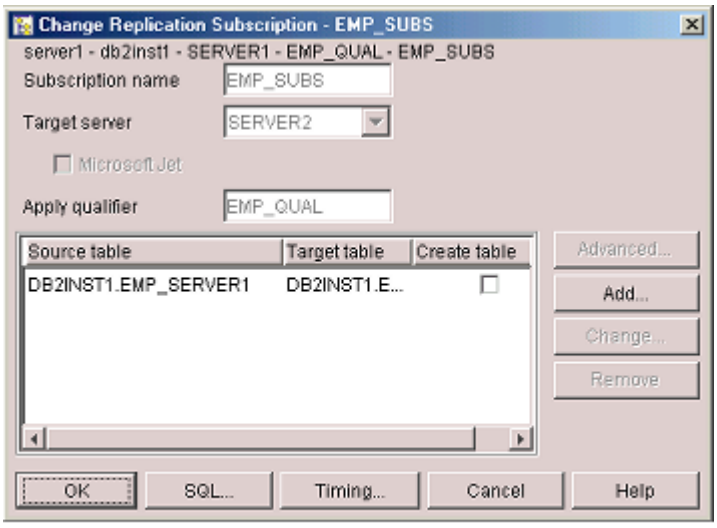


Figura 5.28. Definição do Apply Qualifier no DB2

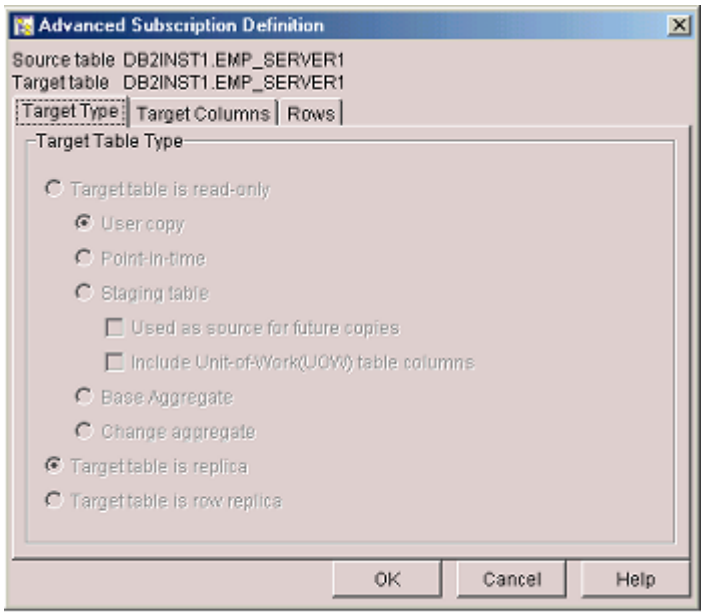


Figura 5.29. Definição do Tipo de Tabela Destino

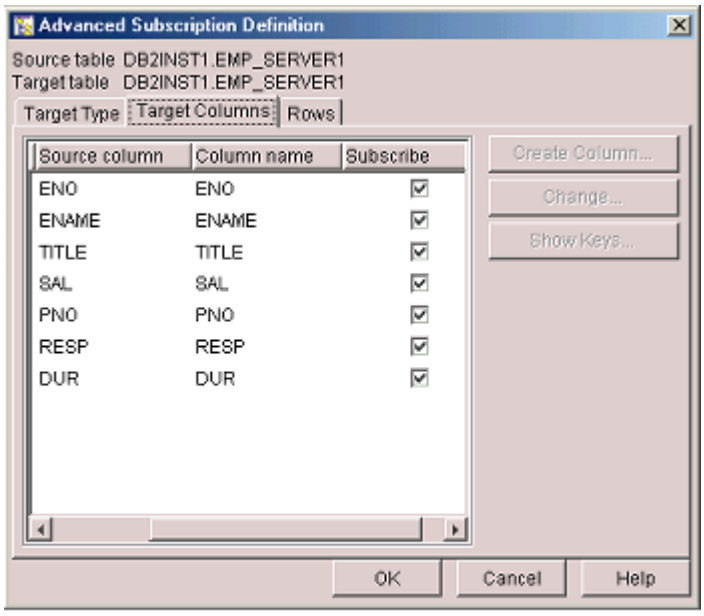


Figura 5.30. Colunas da Tabela Destino

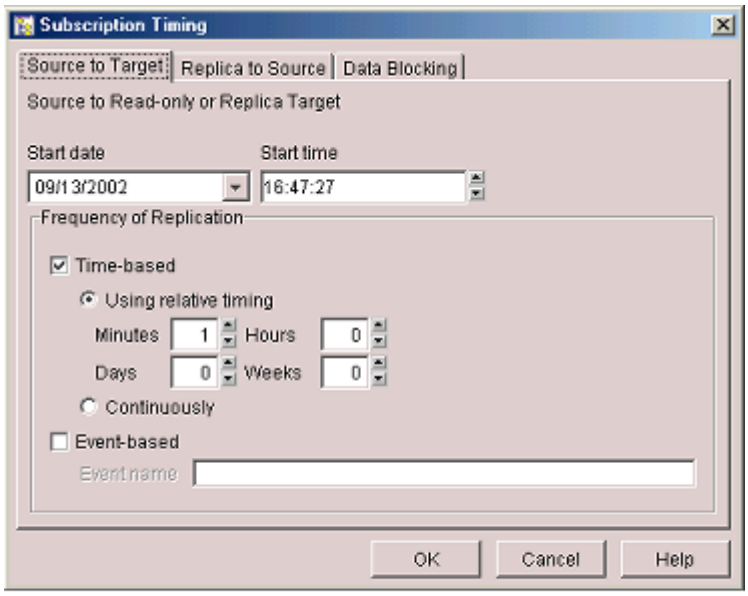


Figura 5.31. Intervalo de Propagação da Origem para o Destino

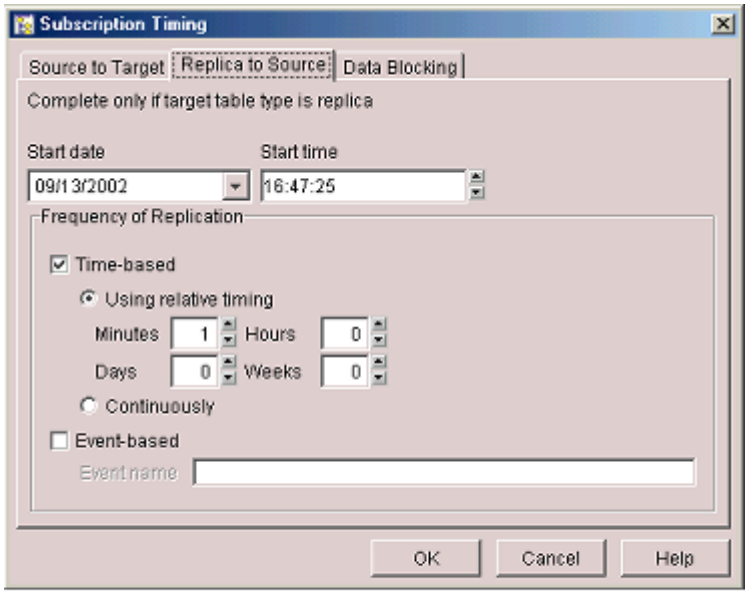


Figura 5.32. Intervalo de Propagação do Destino para a Origem

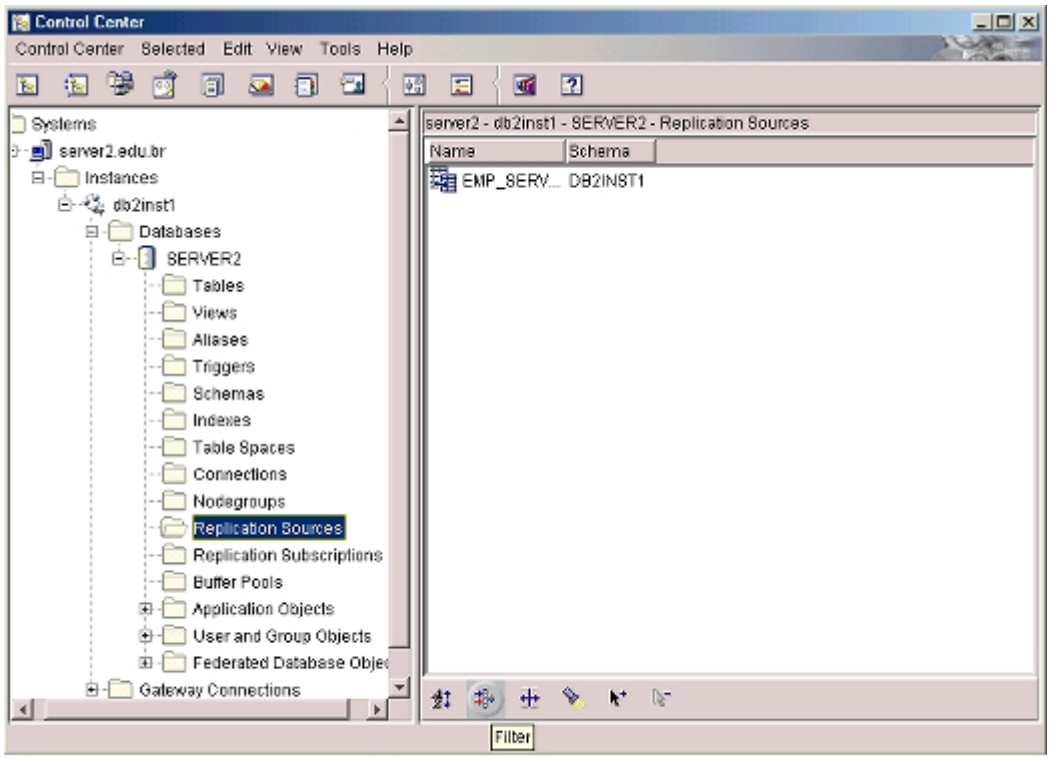


Figura 5.33. Replication Source no Site 2

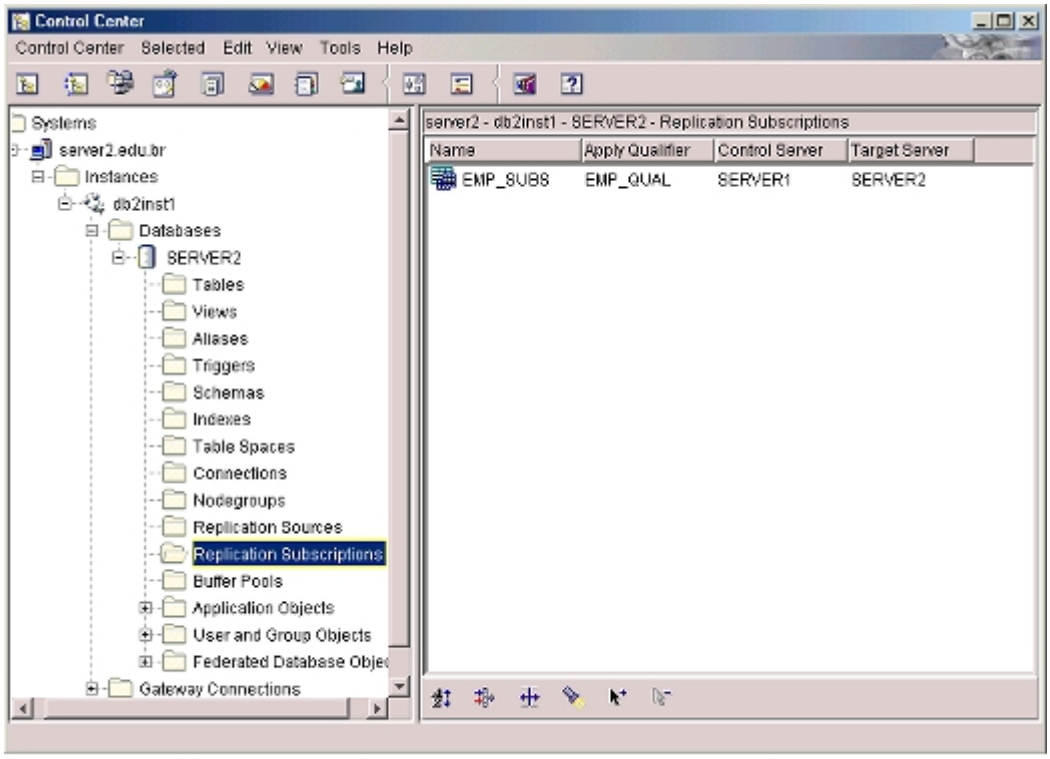


Figura 5.34. Replication Subscription no Site 2

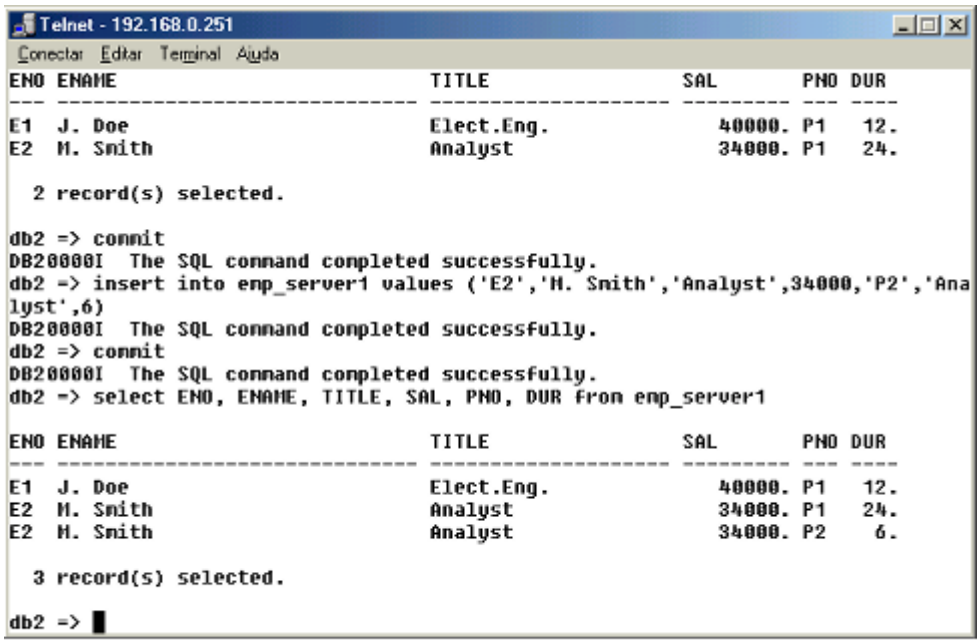


Figura 5.35. Dados da Tabela EMP_SERVER1 no Site 1

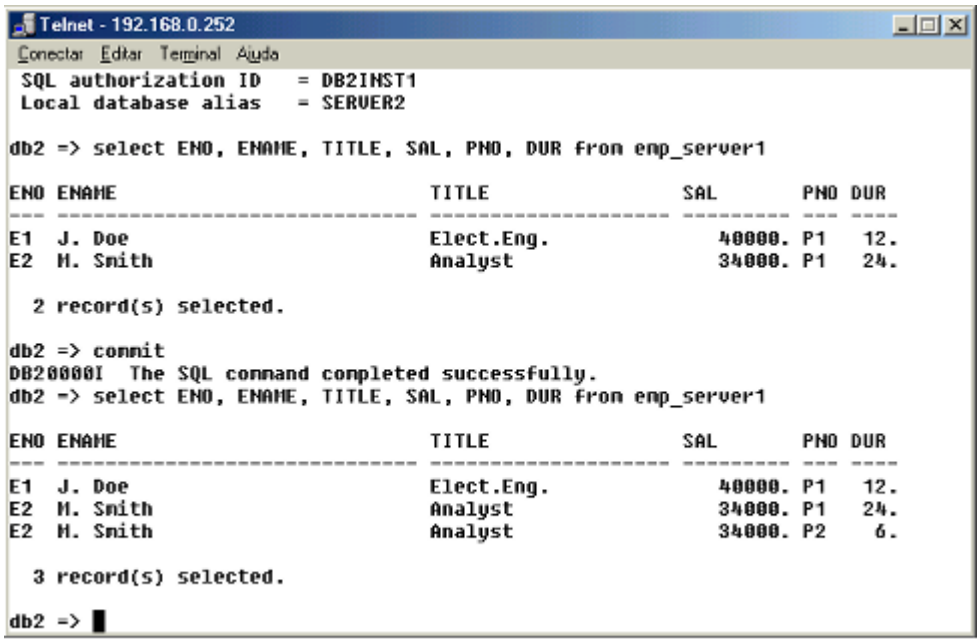


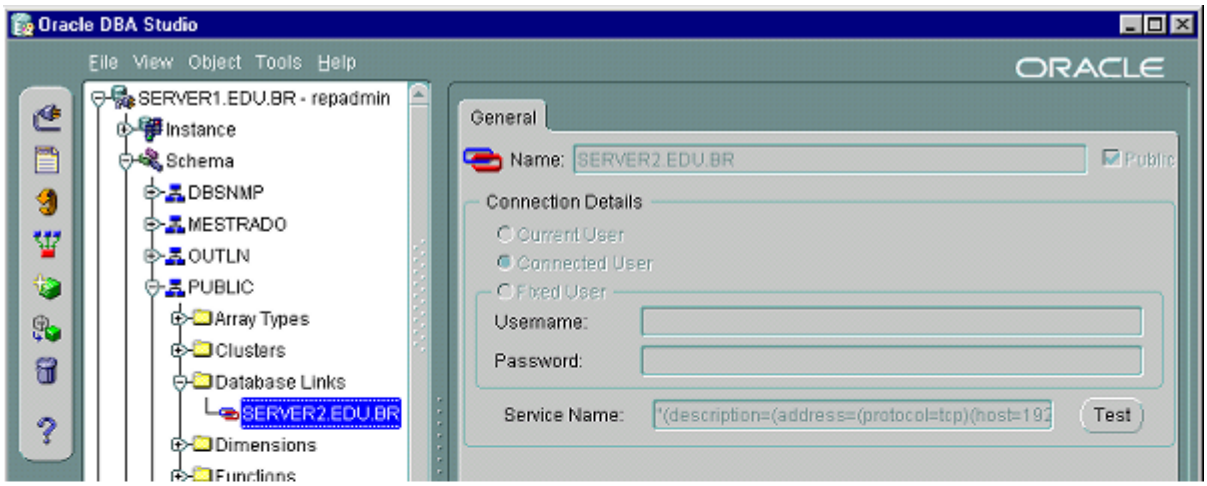
Figura 5.36. Dados da Tabela EMP_SERVER1 no Site 2

5.2.2 SGBD Oracle

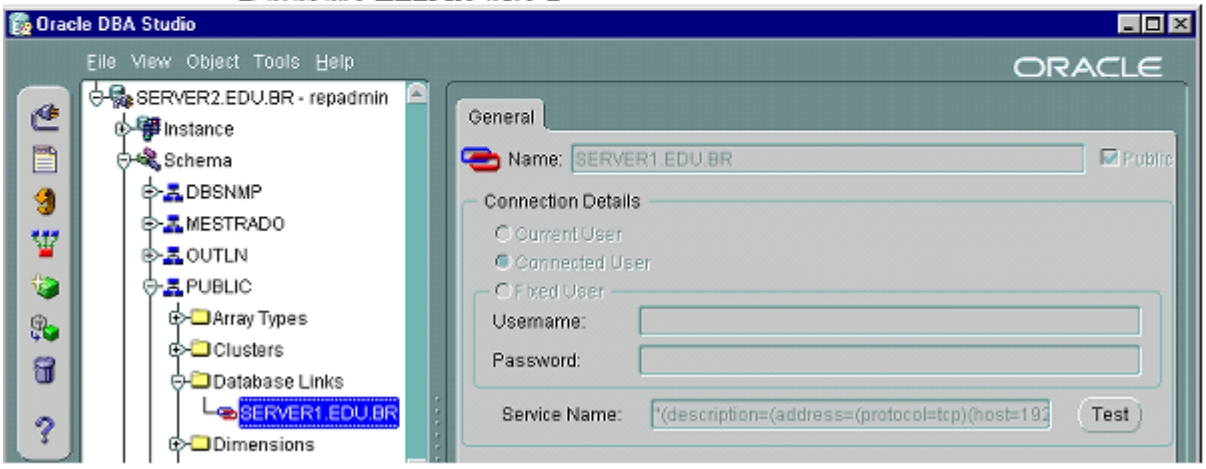
A configuração realizada no sistema de gerência de banco de dados ORACLE para implementar um ambiente de replicação foi complementar a instalação realizada na primeira análise - não sendo necessário proceder a instalação do sistema de gerência de banco de dados. As configuração específicas para a replicação dos dados são:

- Verificar os parâmetros de inicialização do banco de dados, localizado no diretório /home/oracle/dbs. Os parâmetros configurados são:
    - Job_queue_processes = 4
    - Job_queue_interval = 60
    - Shared_pool_size = 6291456
    - Distributed_transaction = 10
    - Global_names = true
    - Open_links = 4
- Executar os scripts como usuário sys: catproc.sql, catrep.sql, catsnap.sql e dbmssnap.sql, localizados no diretório /home/oracle/rdbms/admin. Para isto, acessar o monitor sql, digitando sqlplus sys/change_on_install, no prompt digitar connect to server1 e executar os scripst digitando @/rdbms/admin/comando;
- Criar a tabela EMP no site 1, para isto, acessar o monitor sql, digitando sqlplus mestrado/mestrado001 e a partir do prompt digitar connect to server1. Para criar a tabela utilizar o comando: create table EMP (ENO char(2) not null, ENAME char(30) not null, TITLE char(20) not null, SAL dec(7,0) not null, PNO char(2) not null, RESP char(20) not null, DUR dec(2,0) not null, primary key (ENO, PNO));
- Criar um database link entre os sites, desta forma, no site 1 deve ser criado um database link para o site 2 e no site 2 deve ser criado um database link para o site 1. A figura 5.37 mostra os database links entre os sites;
- A configuração do ambiente é feita através do aplicativo Visual Studio, instalado no computador cliente. Deve-se decidir qual o tipo de replicação desejada - se multimaster ou snapshot. Nesta analise é estudada a replicação multimaster. A primeira configuração da replicação multimaster é a seleção dos sites

participantes da replicação, o administrador do processo de replicação - usuário REPADMIN, e o intervalo de tempo para propagar as modificações e para limpar os logs. A figura 5.38 mostra a configuração dos sites master. A figura 5.39 mostra a tela para a definição dos site (obs: parte da imagem das telas foi cortada de modo a melhorar a apresentação). Outras duas telas de configuração opcional (schema a replicar e customização dos sites) não são apresentadas pelo fato de não serem utilizadas neste teste;



Database Link no Site 1



Database Link no Site 2

Figura 5.37. Database Links entre os Sites

Figura 5.38. Seleção da Configuração dos Master Sites

- A próxima configuração é criar o master group. Neste estudo foi criado o grupo REP_GRUPO. No grupo mestre são configurados os objetos e os sites participantes da replicação. A figura 5.40 mostra a tela de propriedades do grupo. Observar que o campo: Master Definition Site informa que este site é o responsável por gerênciar e configurar a replicação. O campo: Submit Stop Request define o início e a parada do processo de replicação. A figura 5.41 mostra os objetos replicados - tabela EMP, pertencente ao esquema MESTRADO. E a figura 5.42 mostra os sites participantes da replicação - server1.edu.br e server2.edu.br. A partir do término da configuração, o grupo e os objetos replicados são propagados para os sites participantes. A figura 5.43 mostra o grupo de replicação no site 2. Observe que o campo: Master Definition Site contém o valor No, informado que este site não controla a configuração da replicação e o campo Submit Stop Request está desabilitado, informando que este site não pode iniciar ou interromper o processo de replicação;

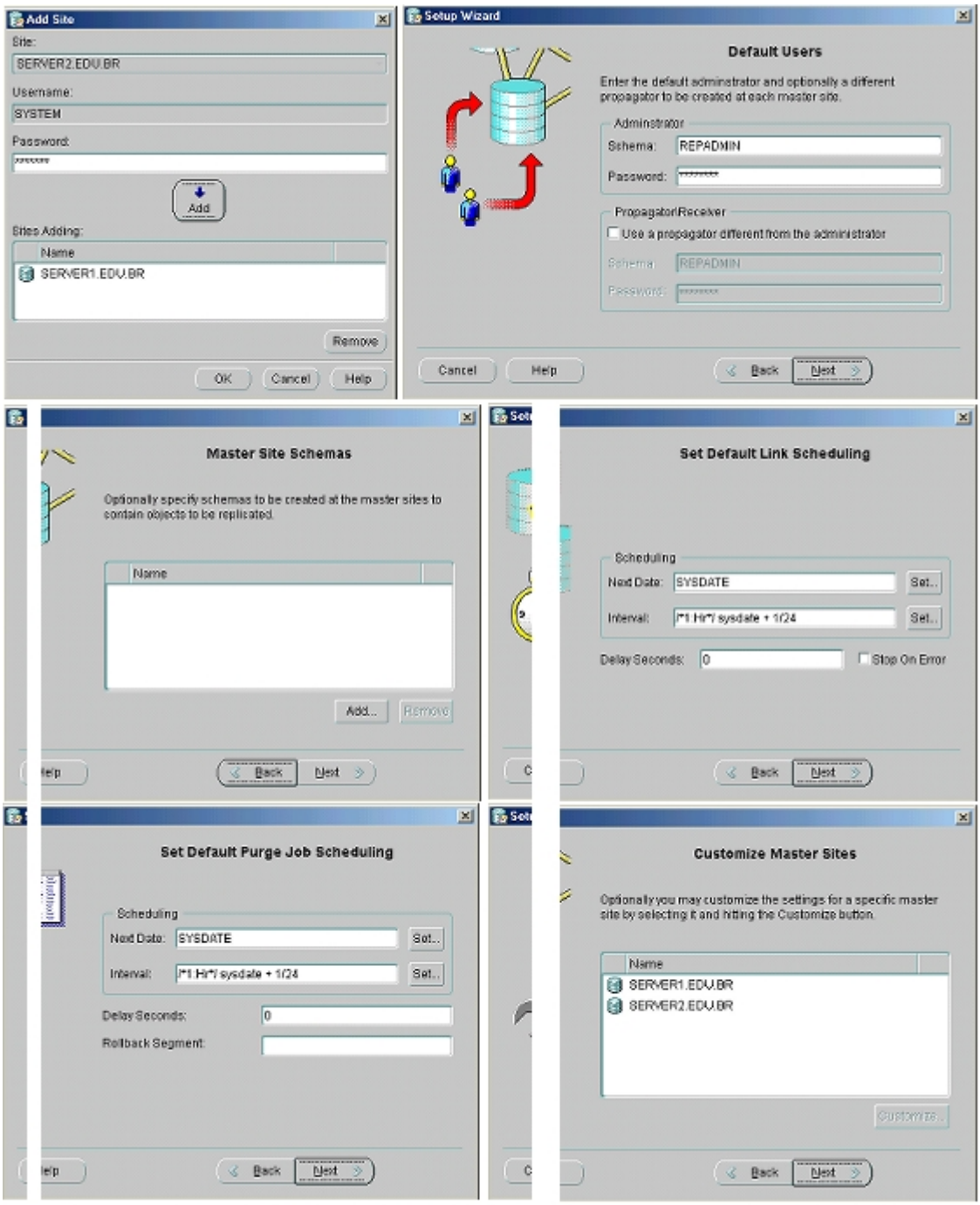


Figura 5.39. Configuração dos Master Sites no SGBD Oracle

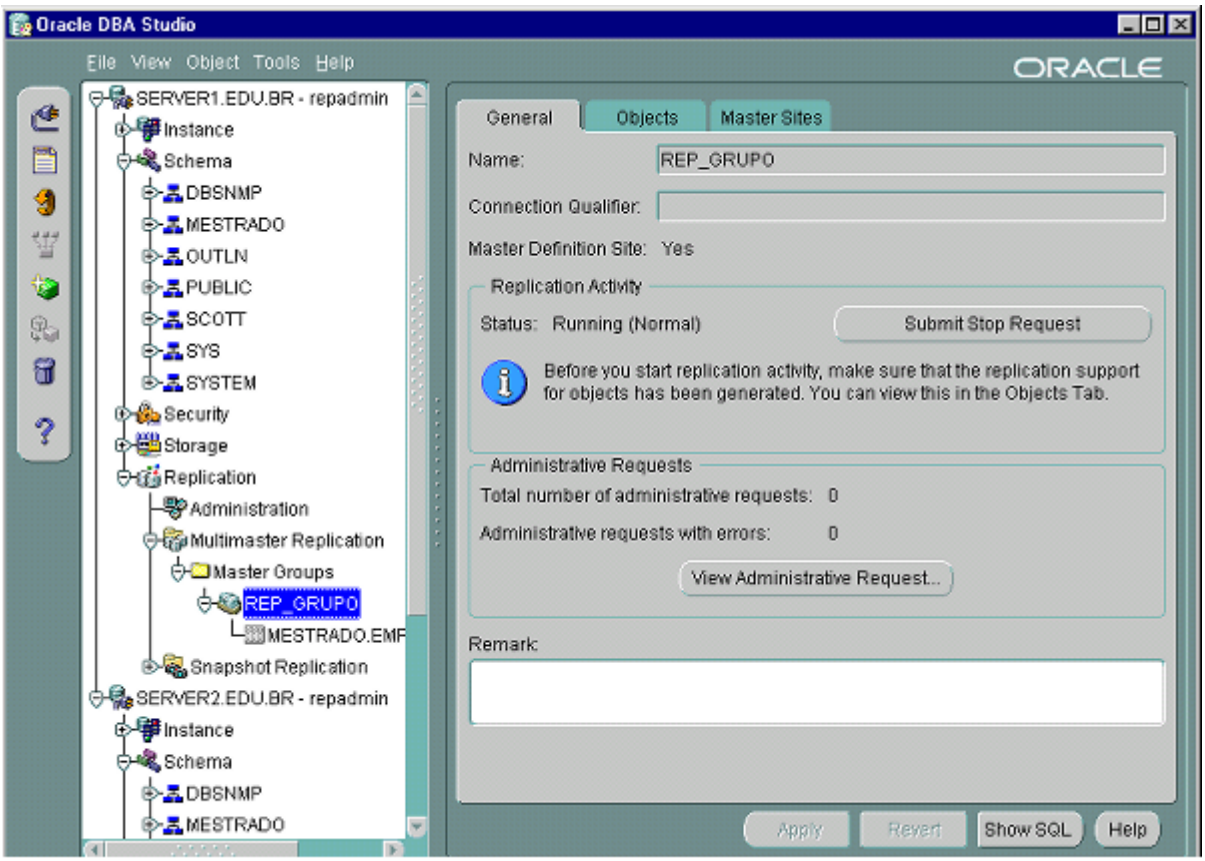


Figura 5.40. Propriedades do Grupo de Replicação

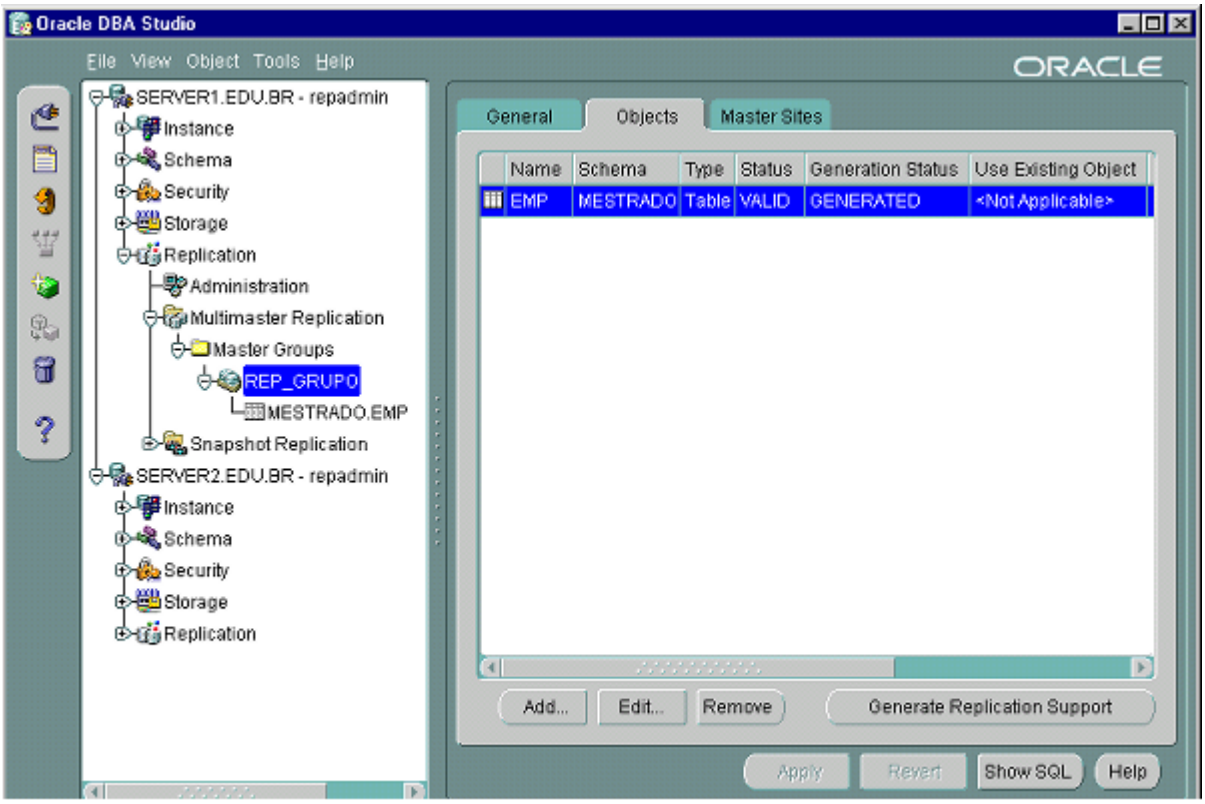


Figura 5.41. Objetos Replicados no Grupo REP_GRUPO

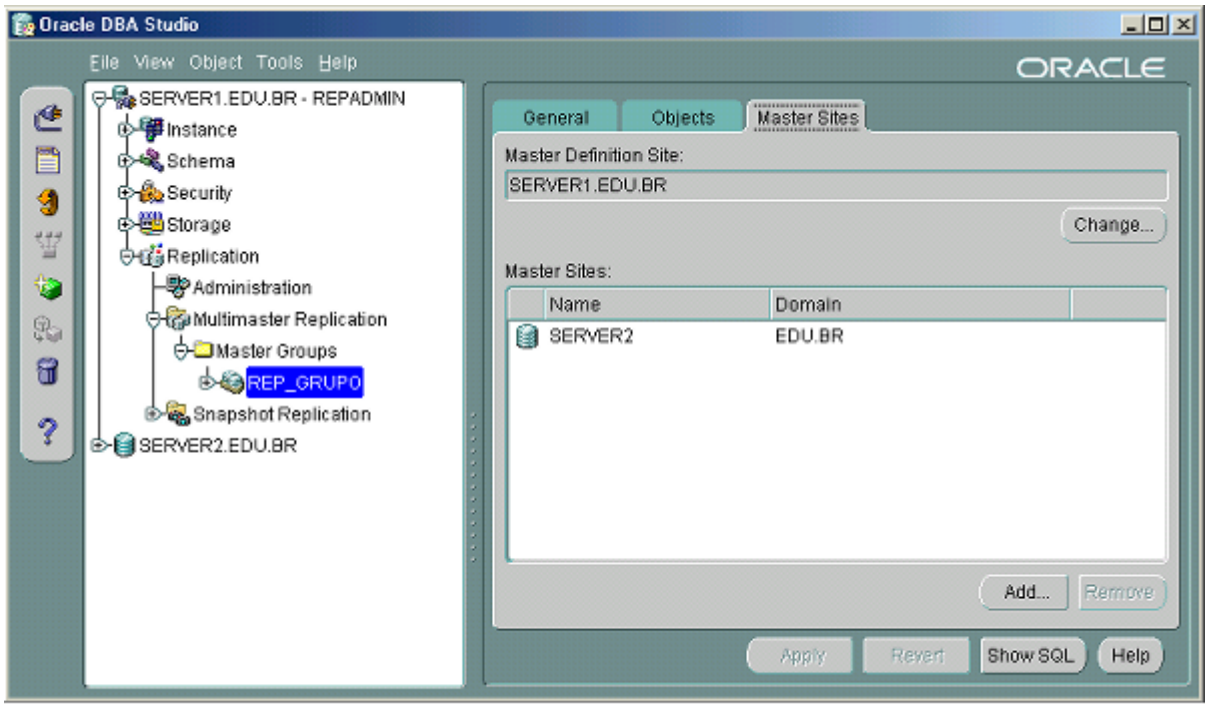


Figura 5.42. Sites Participantes da Replicação

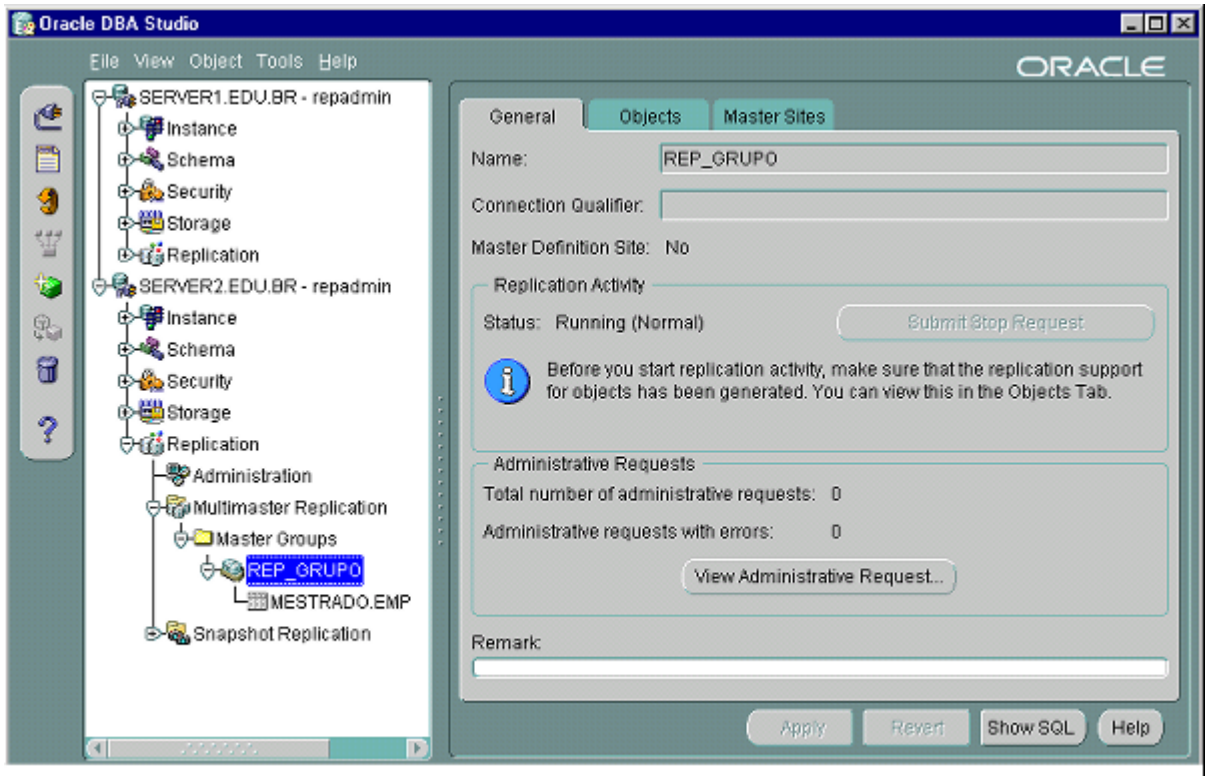


Figura 5.43. Grupo de Replicação no Site 2

Após estas configurações, o processo de configuração está completo. A figura 5.44 mostra a topologia de replicação entre os sites. Para testar a replicação dos dados, foram inseridos um conjunto de dados no site 1 (server1.edu.br) com o endereço IP de

192.168.0.251 e verificado se os mesmos eram replicados para o site 2, com endereço IP de 192.168.0.252. A figura 5.45 mostra as operações de seleção e inserção no site 1 e a figura 5.46 as operações de seleção no site 2.

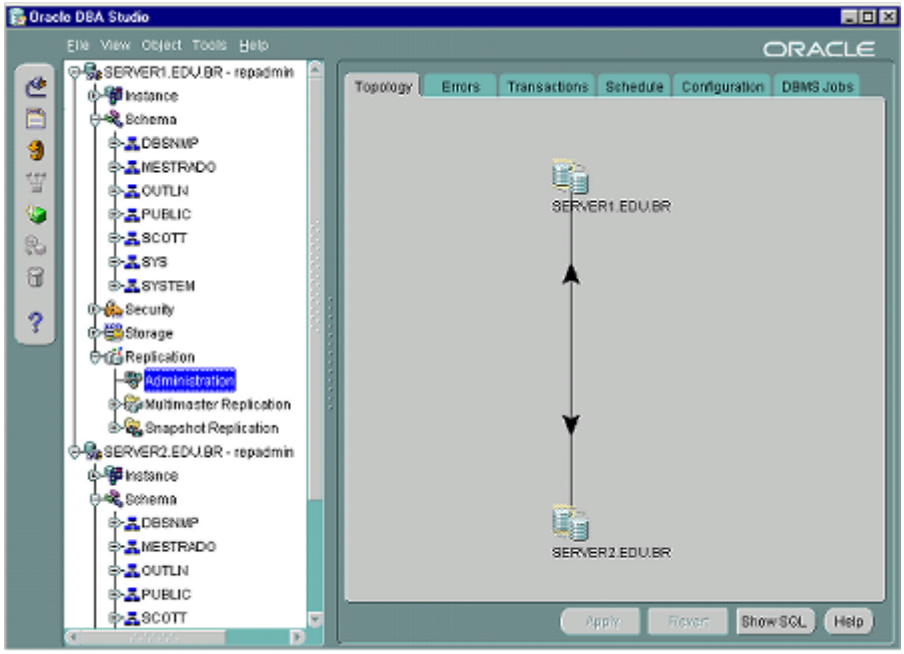


Figura 5.44. Topologia da Replicação

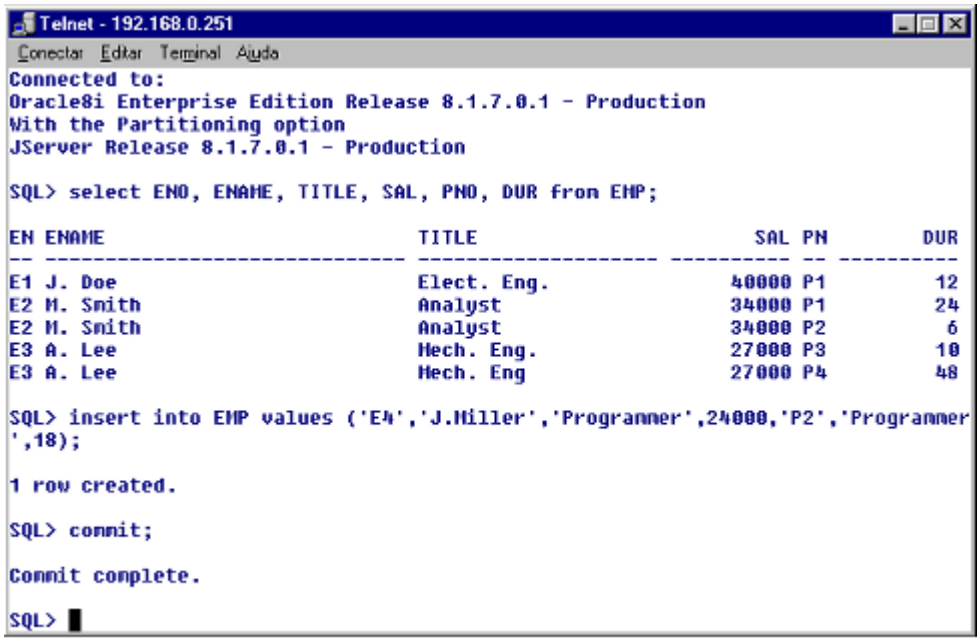


Figura 5.45. Operações de Seleção e Inserção no Site1

Na figura 5.45 foi realizada inicialmente uma operação de seleção sobre a tabela EMP no site 1 e que traz como resultado a existencia de cinco linhas. Da mesma forma, a

figura 5.46 mostra a execução de uma operação de seleção no site2, apresentando como resultado as mesmas cinco linhas. A operação de inserção de dados no site 1 - mostrado na figura 5.45 mostra a criação de uma nova linha. Uma nova operação de seleção, realizada no site 2 e demonstrada na figura 5.46, apresenta como resultado a existência de seis linhas, permitindo concluir que a replicação foi realizada.

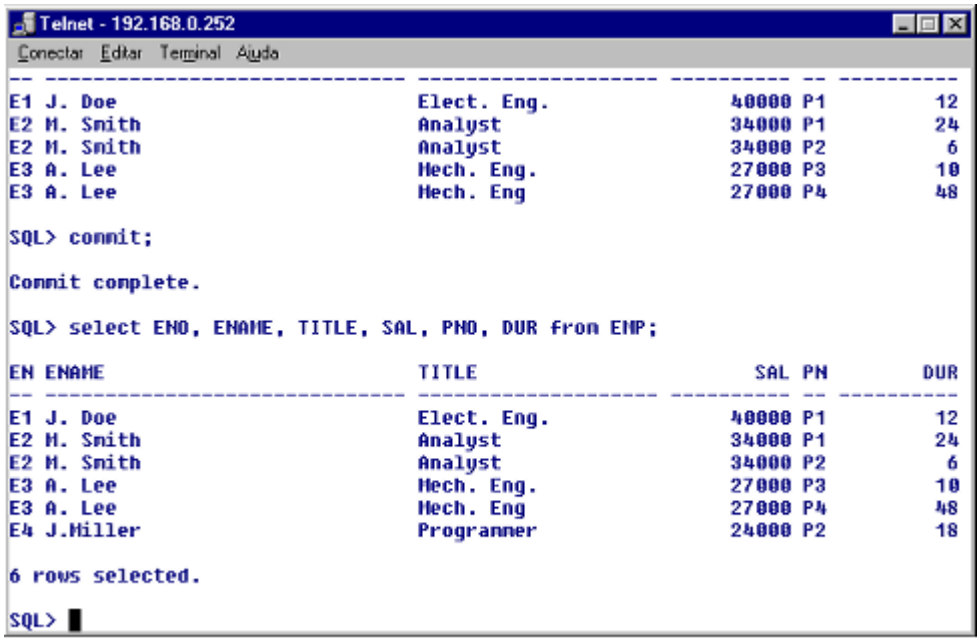


Figura 5.46. Operações de Seleção no Site 2

5.2.3 Conclusões da Análise 2

Esta análise investigou a implementação da propriedade de transparência de replicação nos SGBD Oracle e DB2. As conclusões obtidas são:

1. Para obter a propriedade de transparência de replicação os SGBDs implementam:
   a. A comunicação entre os sites origem e destino é feita por meio do registro dos bancos de dados. Neste estudo, foi necessário registrar os dois bancos de dados locais entre si;
   b. O registro dos bancos de dados é feito através de conexões entre os banco de dados - database links, de modo idêntico a realizada na primeria análise. Na configuração é cadastrando um nome para a conexão, o endereço IP ou o nome do servidor remoto e o nome do serviço - associado a uma porta do

protocolo TCP/IP. As portas utilizadas por cada SGBD são as mesmas da primeira análise;

c. Definida a conexão entre os bancos de dados, é realizada a definição dos locais participantes do ambiente de replicação. Deve ser definido um site controlador que tem a função de gerência o processo de replicação;

d. Após a definição dos sites participantes deve ser configurado os objetos a serem replicados e o intervalo de tempo para realizar a cópia das modificações para os sites - propagação dos dados;

e. Realizada a configuração, os esquemas dos bancos de dados e os objetos a replicar são copiados entre os sites, caso não existam são criados nos sites remotos. A gerência e coordenação entre os sites é feita pelo site controlador;

2. Ambos os SGBDs analisados tem um processo semelhante de gerência a replicação dos dados - variando somente na forma de configurar o ambiente. No SGBD Oracle, dois ambientes são possíveis - replicação Multimaster e replicação Snapshot. A replicação Multimaster permite cópias completas dos objetos e manipulação dos dados em qualquer site participante. A replicação Snapshot permite fragmentação dos objetos e a escolha entre replicação somente para leitura ou manutenção dos objetos em qualquer site. No SGBD DB2 tem um ambiente de replicação único, permitindo a configuração de replicação completa, somente para leitura e fragmentada dos objetos.

A tabela 5.5 apresenta um resumo dos comandos para configurar um ambiente de replicação de dados.

Tabela 5.5. Comandos para Configurar Replicação de Dados

Tarefa	DB2	Oracle
Catalogar site	Catalog tcpip node	Utilitário net8 - nomeação de serviço
Catalogar banco de dados	Catalog database	Aplicativo Visual Studio - database link
Configurar replicação	Aplicativo Control Center criar replication source e replication subscription	Aplicativo Visual Studio - replication
Iniciar replicação	Asnccp asnapply	Aplicativo Visual Studio - replication activity

6. CONCLUSÃO

A análise das propriedades de transparência de localização e nomeação dos SGBD Oracle, DB2, Sybase e DB2 e da propriedade de transparência de replicação dos SGBD Oracle e DB2, realizada nesta dissertação investiga os modos de implementação dos ambiente de distribuição e/ou de replicação de dados utilizando-se estes sistemas de gerência de bancos de dados, sob o sistema operacional Linux, distribuição Red Hat, versão 6.2. Esta pesquisa contribui para o conhecimento das características de cada SGBD analisado, descrevendo as características e o suporte fornecido pelos sistemas de banco de dados, bem como a forma como cada sistema deve ser configurado para implementar a distribuição ou a replicação dos dados. As análises desenvolvidas permitem conhecer os sistemas e suas características e concluir que as versões utilizadas implementam um ambiente distribuído - no caso dos quatro SGBDs, e um ambiente replicado - no caso dos SGBDs Oracle e DB2.

Esta pesquisa demonstra que todos os sistemas podem ser utilizados sob o sistema operacional linux, tornando este ambiente uma alternativa interessante - dadas as suas características técnicas e comerciais, para a implementação de um sistema de banco de dados distribuído e/ou replicado.

A investigação realizada nesta dissertação permite conhecer os detalhes e soluções adotadas pelos fabricantes destes SGBD, no que diz respeito a implementação de um sistema de banco de dados distribuído e replicado. Permite, da mesma forma, estabelecer uma relação entre a teoria dos sistemas de banco de dados distribuídos com sua implementação utilização prática - aplicada no ambiente das empresas e instituições.

Este estudo obteve sucesso na configuração de um ambiente de distribuição e replicação implementado sob o sistema operacional linux e contribui para o estudo dos sistemas de banco de dados distribuídos e das aplicações para o sistema operacional linux. Possibilita estabelecer uma relação teoria-prática no estudo dos sistemas de banco de dados, em especial, no estudo dos bancos de dados distribuídos. Com relação aos SGBD estudados, podemos concluir que:

a. Os SGBDs estudados podem ser instalados e configurados sob o SO Linux, não ocorrendo problemas na execução das aplicações;

b. Os SGBD analisados permitem a implementação de um ambiente de distribuição de dados;
c. Os SGBD Oracle e DB2 implementam um ambiente de replicação de dados peer-to-peer, multidatabase e homogêneo;
d. O sistema operacional linux, em particular a distribuição Red Hat 6.2, constituí-se uma alternativa que deve ser analisada para a implementação e o uso destes SGBD e de um ambiente de banco de dados distribuído;
e. Na avaliação de desempenho dos SGBD na realização de consultas distribuídas, o SGBD Oracle teve o menor custo de realização das funções count e na consulta sem o uso de cursores. O SGBD DB2 teve menor custo de realização de consultas distribuídas com o uso de cursores.

Como recomendações para trabalhos futuros, sugerimos:

a. Ampliação do estudo de caso número 2, inserindo no estudo os SGBD Ingres e Sybase;
b. Implementação de um ambiente de replicação e fragmentação de dados;
c. O estudo das propriedades de transparência de localização, nomeação e replicação sob sistemas de bancos de dados não comerciais, como o Mysql, Postgres, Interbase e Firebird;
d. Investigação da performance na realização de consultas em bancos de dados distribuídas, ampliando o estudo de caso número 1, especificamente a investigação dos resultados apresentados, no que se refere aos tempos de realização das consultas;
e. Implementação de um ambiente de consolidação de dados, modificando o estudo de caso número 1 e fazendo com que os dados de cada site local sejam replicados e consolidados em um banco de dados único;
f. Investigar o por que dos resultados obtidos nas avaliações de performance, procurando detectar as diferenças de implementação de cada SGBD que possam contribuir para este resultado.

REFERENCIAL BIBLIOGRÁFICO

ATZENI, Paolo; CERI, Stefano; PARABOSCHI, Stefano; TORLONE, Ricardo. Database Systemas. Concepts, Languages and Architectures. Editora McGrawHill. England. 2000. Chap. 10, pag. 349-393.

CHAKRAVARTHI, Upen S.; GRANT, John; MINKER, Jack. Logic-Based Approach to Semantic Query Optimization. ACM Transactions in Database Systems , Vol 15, No. 2. June 1990. Disponível por www em http://www.acm.org

COMPUTERWORLD. Oracle Corporation lidera vendas no mercado de bases de dados. IDG-International Data Group. 2001. Disponível por www em http://www.computerword.iol.pt/resources Acesso em 02/2002.

DOMOKOS, Pal. Ingres II Howto. Versão 1.01. Dezembro de 1999. Disponível por www em http://new.linuxnow.com. Acesso em 04/2001.

ELMASRI, Rames; NAVATHE, Shamkant B. Fundamentals of Database Systems. Terceira Edição. Editora Addison.Wesley. Massachusetts. 2000. Chap. 18, pag. 585-628. Chap. 24, pag. 765-799. Chap. 25, pag. 801-840.

IBM, International Business Machine Corporation. DB2 Replication Guide and Reference Version 7. 2000. USA. Cap. 1, pag. 3-18. Disponível por www em http://www.ibm.com

IBM, International Business Machines Corporation. DB2 Administration Guide. Disponível em: http://webdocs.caspur.it/ibm/web/udb-6.1/db2d0/db2d0.htm. Acesso em 02/2001.

IBM, International Business Machines Corporation. DB2 Universal Database. DB2 Software Lab. Canadá. Maio de 1999.

CA - Computer Associates International Inc. Ingres II System Reference Guide. New York. 1998.

CA - Computer Associates International Inc. Ingres II Database Administrator Guide. New York. 1998.

CA - Computer Associates International Inc . Ingres/Star User Guide. New York. 2000.

CA - Computer Associates International Inc . Ingres/Replicator User Guide. New York. 1998.

MOHAN, C. IBM´s Relational DBMS Products: Feature and Technology. Data Base Technology Institute. São Jose. CA. 1993.

ORACLE, Oracle Corporation. Oracle 8i Distributed Database Systems, Release 2. Dezembro 1999.

ORACLE, Oracle Corporation. Oracle 8i Replication, Release 2. Dezembro 1999.

ORACLE Corporation. Concepts, Relese 2. Dezembro 1999.

ORACLE, Oracle Corporation. Oracle 8i Instalattion Guide. Release 3 (8.1.7) for Linux Intel. Dezembro 2000.

ÖZSU, M. Tammer; VALDURIEZ, Patrick. Principles of Distributed Database Systems. Segunda Edição. Editora Prentice Hall. New Jersey. 1999. Chap. 1, pag. 2-24. Chap. 4, pag. 75-100. Chap. 7, pag. 188-202. Chap. 9, pag. 228-271.

ÖZSU, M. Tammer. Distributed Database. University of Alberta. 2000. Disponível por www em www._____

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. Database Management Systems. Segunda Edição. Editora McGrawn-Hill. Singapura. 2000. Chap. 21, pag. 597-641. Chap. 13, pag. 359-373.

SCOTT, Dan. DB2 Version 7.1 for Linux HOWTO. Disponível em: http://www.linuxdoc.org. Acesso em 10/2001.

SHETH, Amit P.; LARSON, James A. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys, Vol. 22, No. 3, September 1990. Disponível por www em http://www.acm.org

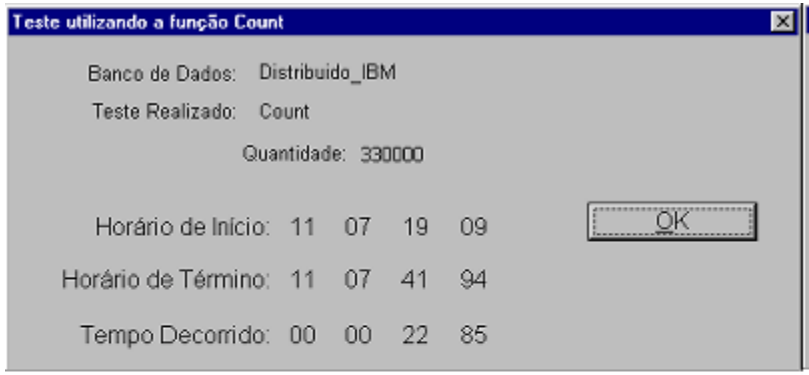
SYBASE, Sybase Inc. Instalation Guide . Junho 2001. Disponível por www em http://www.sybase.com. Acesso em 06/2001

SYBASE, Sybase Inc. Sybase Adaptive Server Enterprise System Admnistration Guide. Setembro 1997. Disponível por www em http://www.sybase.com. Acesso em 10/2001

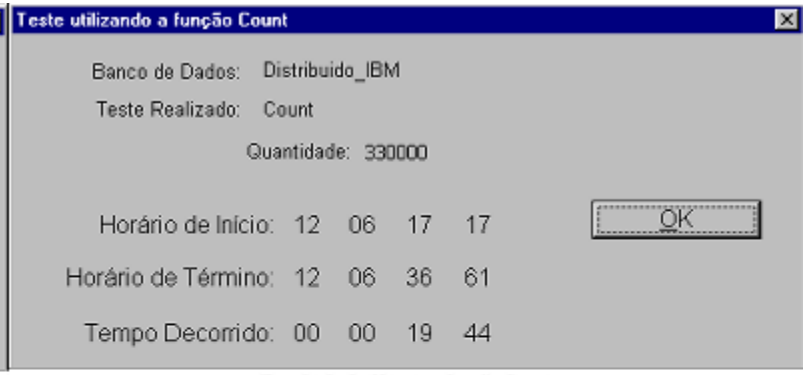
SYBASE, Sybase Inc. Replication Server Admnistration Guide. Novembro 1995. Disponível por www em http://www.sybase.com. Acesso em 10/2001

SYBASE, Sybase Inc. Replication Server Design Guide. Abril 1997. Disponível por www em http://www.sybase.com. Acesso em 11/2001

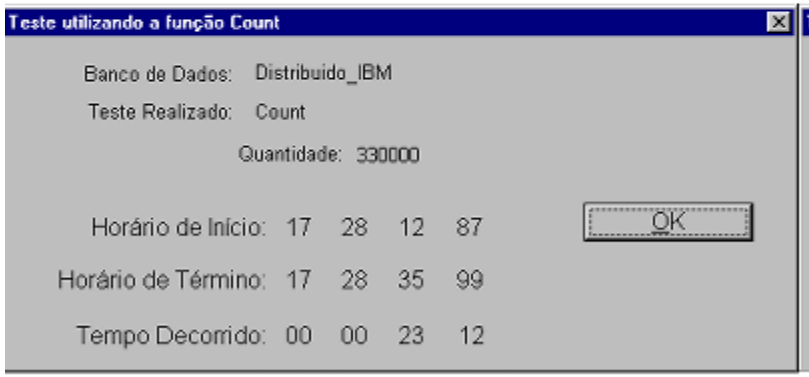
Anexo A
Testes com SGBD DB2



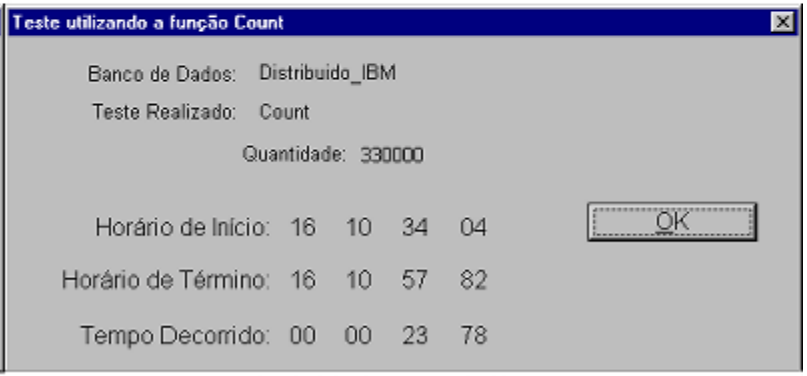
Resultado do 1° teste - função Count



Resultado do 2° teste - função Count

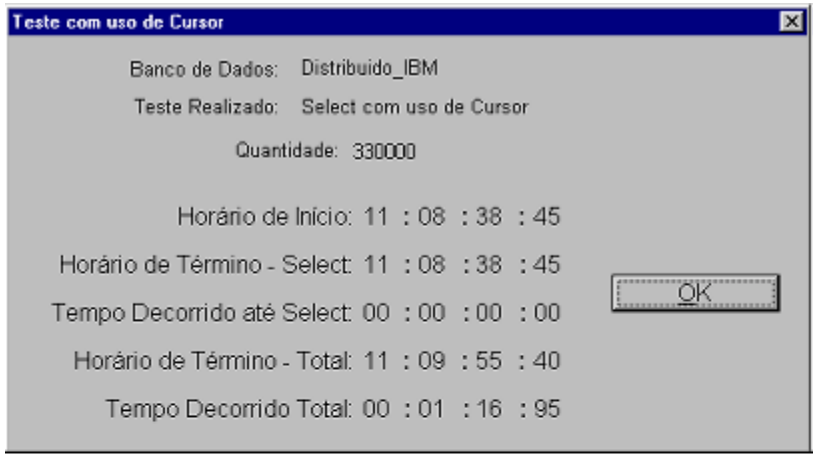


Resultado do 3° teste - função Count

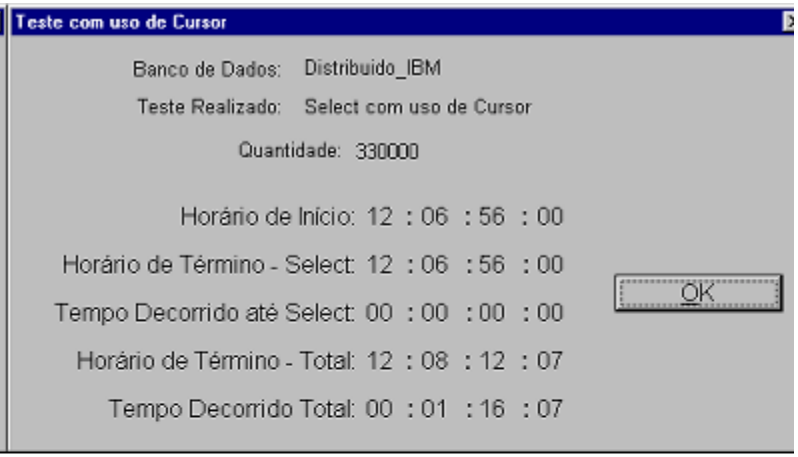


Resultado do 4° teste - função Count

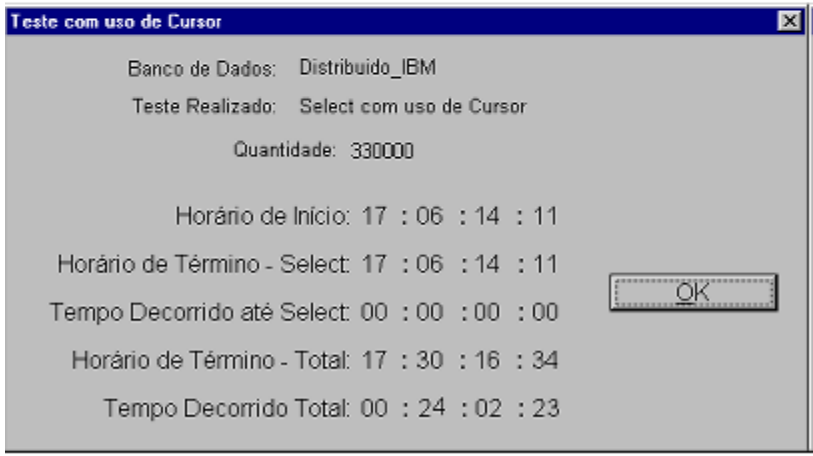
Resultado dos Testes com a Função Count



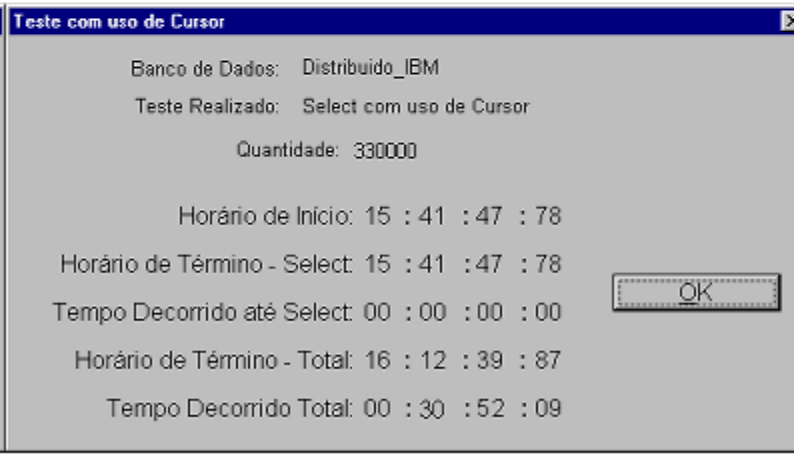
Resultado do 1° teste - Select com Cursor



Resultado do 2° teste - Select com Cursor

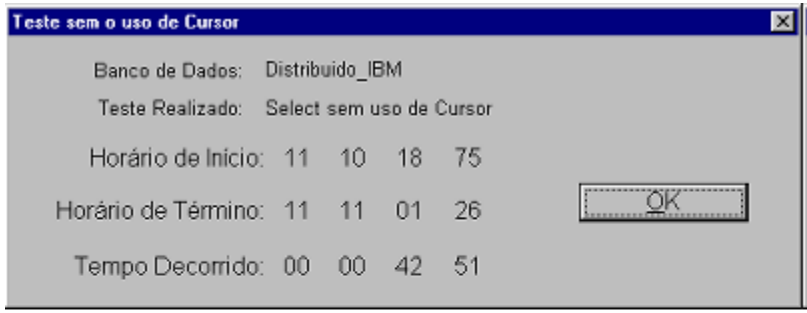


Resultado do 3° teste - Select com Cursor

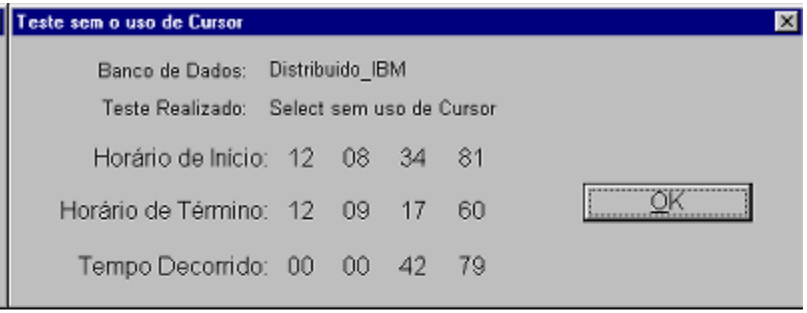


Resultado do 4° teste - Select com Cursor

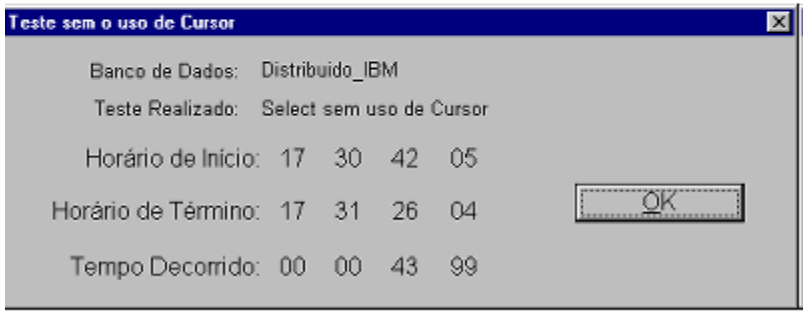
Resultados dos Testes de Select com Cursor



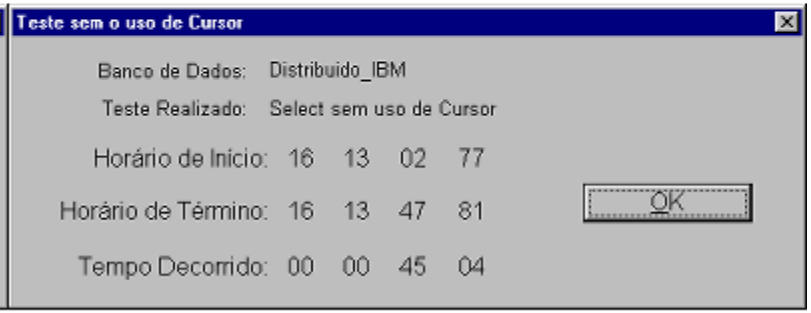
Resultado do 1° teste - Select sem Cursor



Resultado do 2° teste - Select sem Cursor

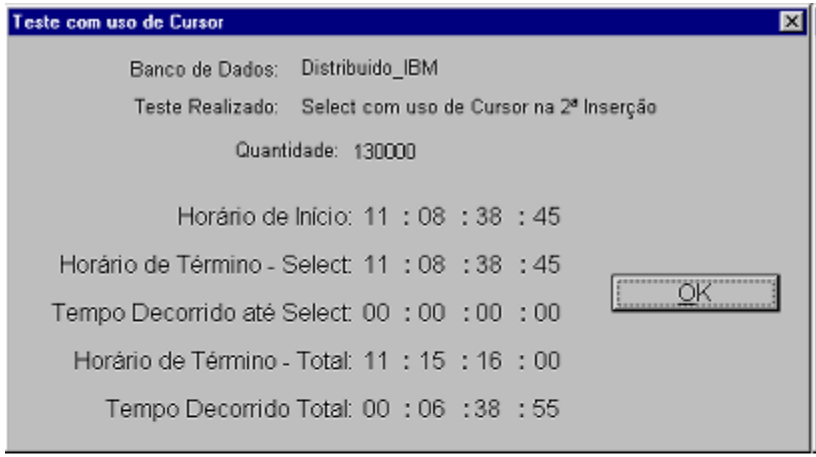


Resultado do 3° teste - Select sem Cursor

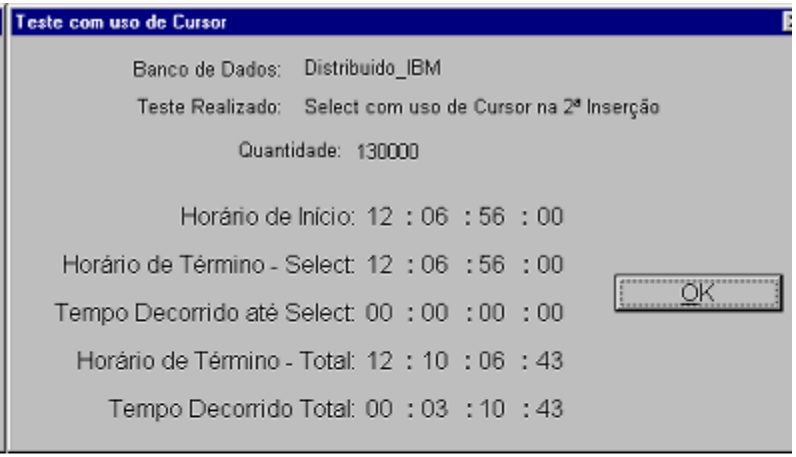


Resultado do 4° teste - Select sem Cursor

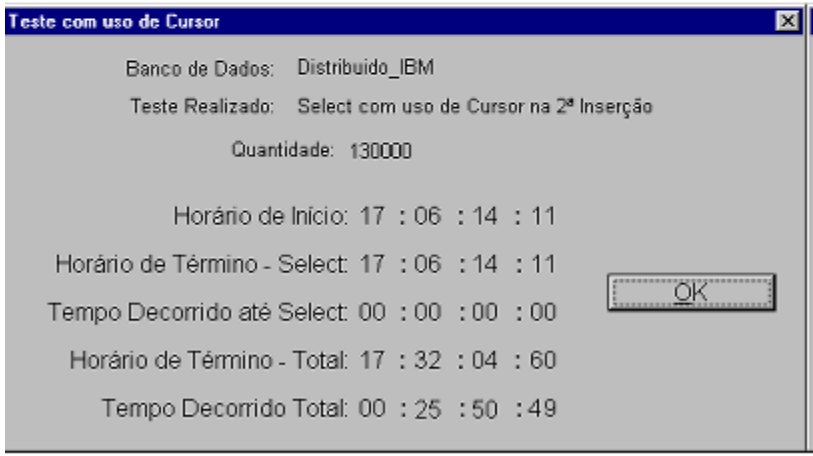
Resultados dos Testes de Select sem Cursor



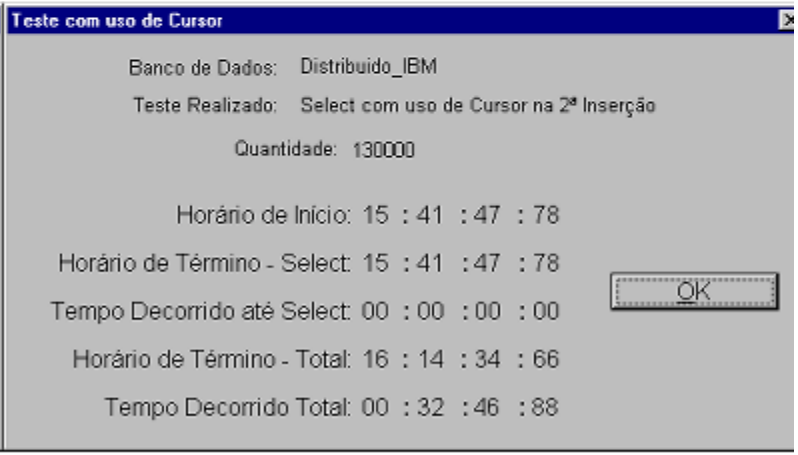
Resultado do 1ª teste - Select com Cursor - 2ª inserção



Resultado do 2ª teste - Select sem Cursor - 2ª inserção



Resultado do 3ª teste - Select sem Cursor - 2ª inserção



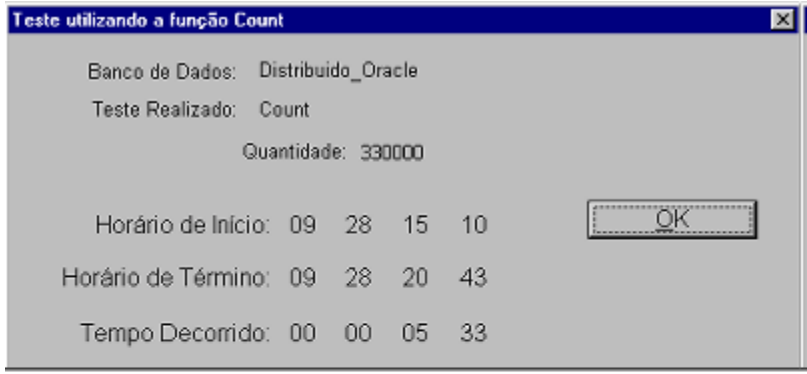
Resultado do 4ª teste - Select sem Cursor - 2ª inserção

Resultado dos Testes com Select sem Cursor na 2ª Inserção

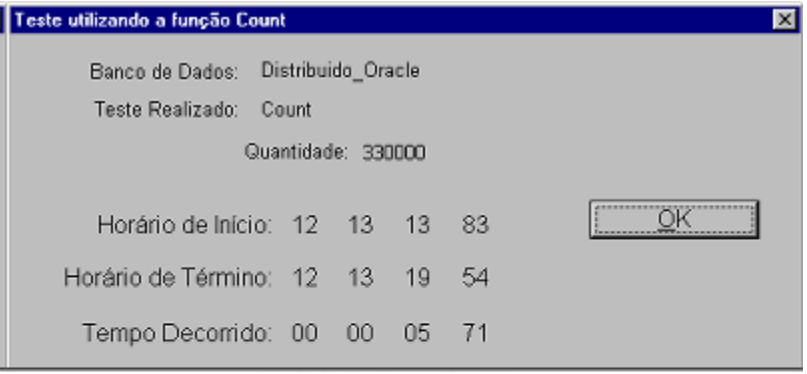
Teste	Início	Fim	Resultado
Função Count	11:07:19:09	11:07:41:94	22:85
	12:06:17:17	12:06:36:61	19:44
	17:28:12:87	17:28:35:99	23:12
	16:10:34:04	16:10:57:82	23:78
Select c/Cursor	11:08:38:45	11:09:55:40	01:16:95
	12:06:56:00	12:08:12:07	01:16:07
	17:06:14:11	17:30:16:34	24:02:23
	15:41:47:78	16:12:39:87	30:52:09
Select sem Cursor	11:10:18:75	11:11:01:26	42:51
	12:08:34:81	12:09:17:60	42:79
	17:30:42:05	17:31:26:04	43:99
	16:13:02:77	16:13:47:81	45:04
Select na 2ª inserção	11:08:38:45	11:15:16:00	06:38:55
	12:06:56:00	12:10:06:43	03:10:43
	17:06:14:11	17:32:04:60	25:50:49
	15:41:47:78	16:14:34:66	32:46:88

Resultado testes - SGBD DB2

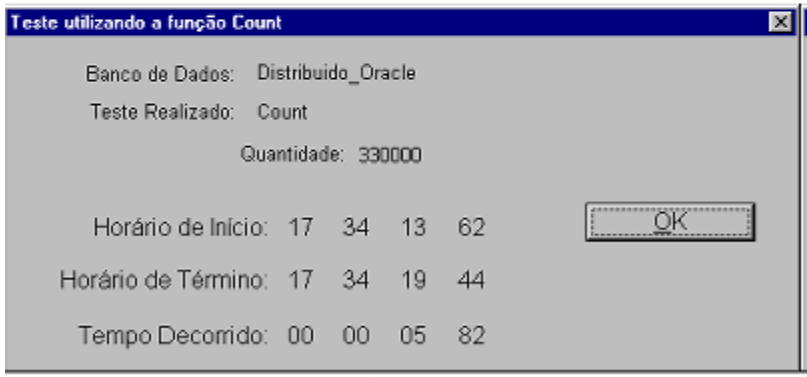
Anexo B
Testes com SGBD Oracle



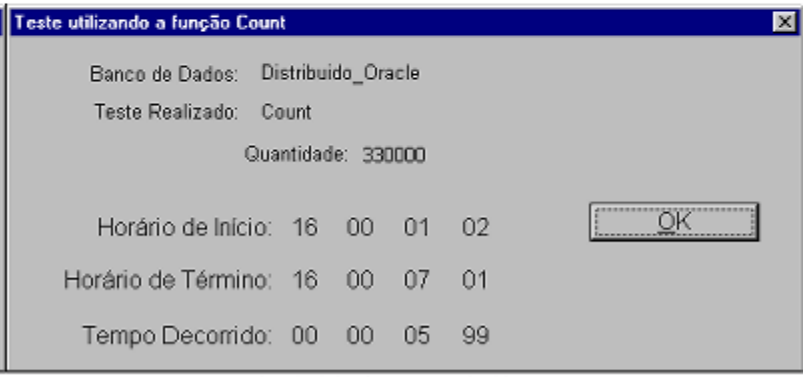
Resultado do 1° teste - função Count



Resultado do 2° teste - função Count

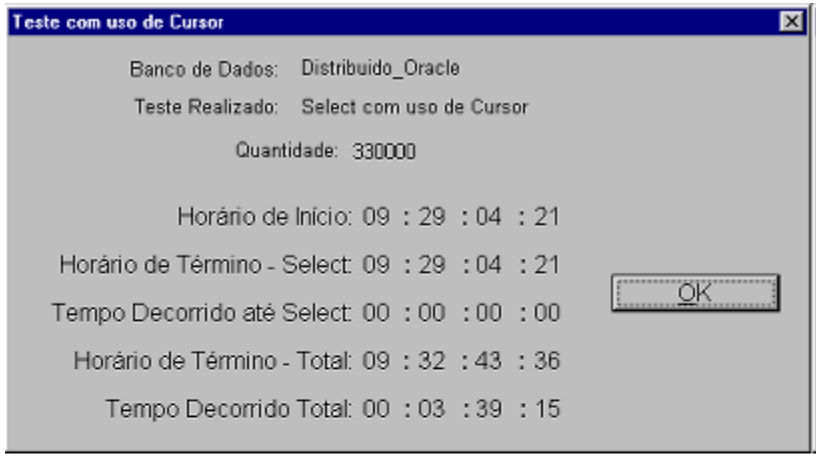


Resultado do 3° teste - função Count

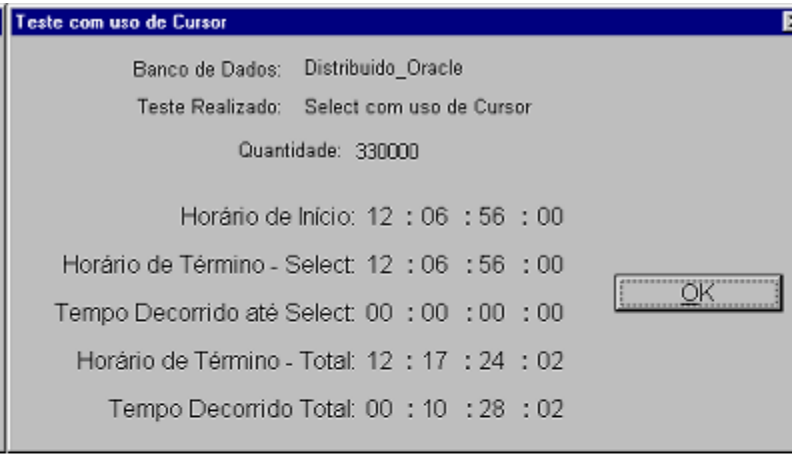


Resultado do 4° teste - função Count

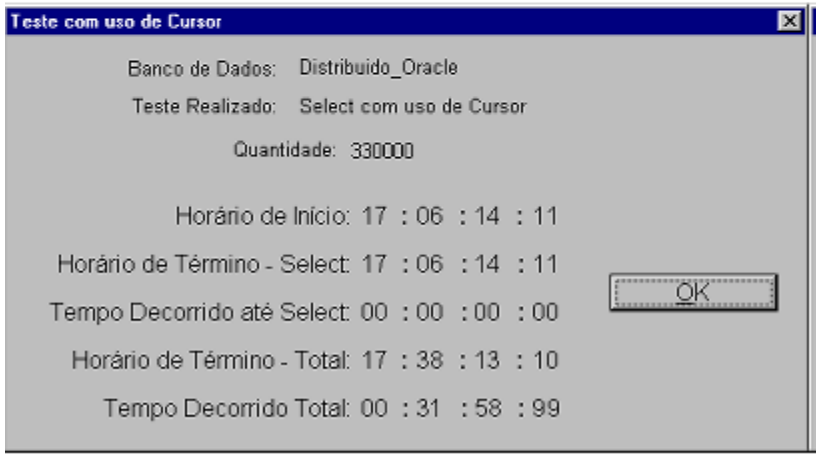
Resultados dos Testes da Função Count



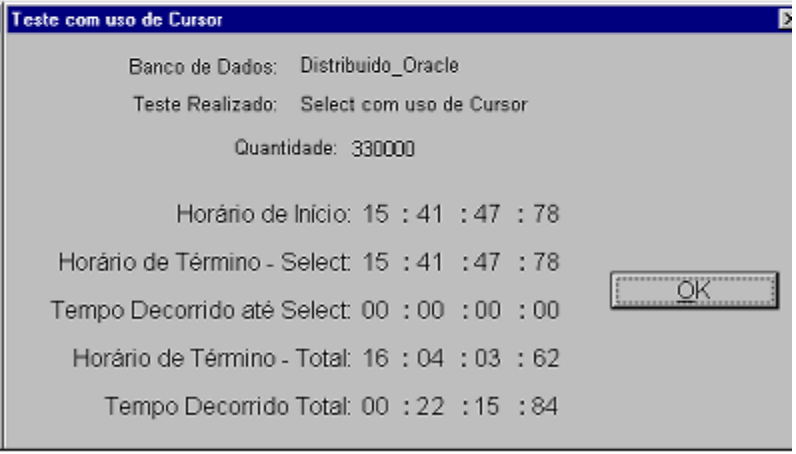
Resultado do 1° teste - Select com Cursor



Resultado do 2° teste - Select com Cursor

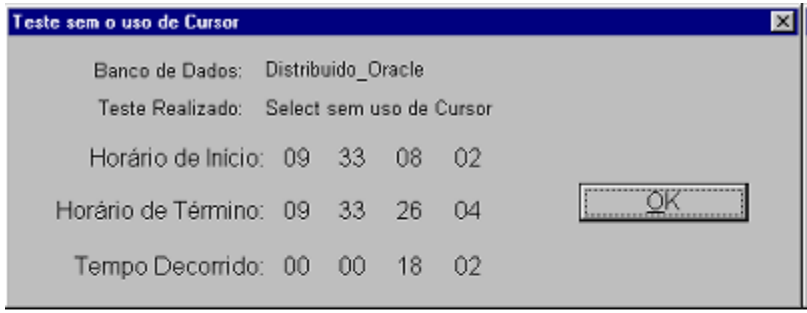


Resultado do 3° teste - Select com Cursor

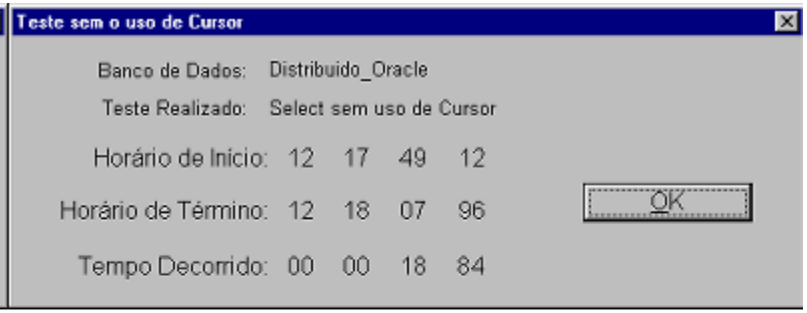


Resultado do 4° teste - Select com Cursor

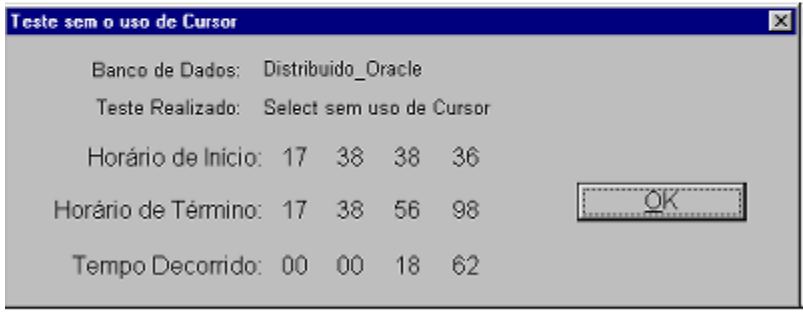
Resultados dos Testes com o uso de Cursor



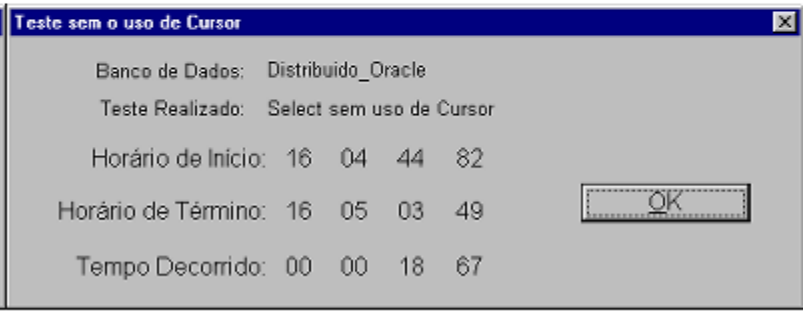
Resultado do 1° teste - Select sem Cursor



Resultado do 2° teste - Select sem Cursor

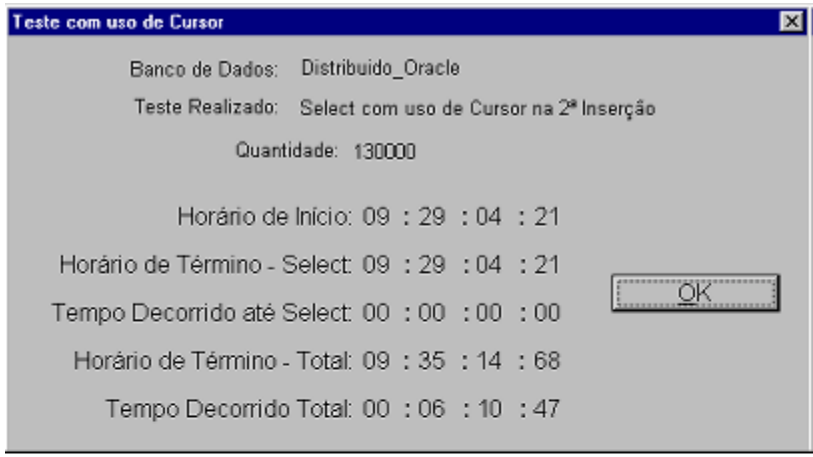


Resultado do 3° teste - Select sem Cursor

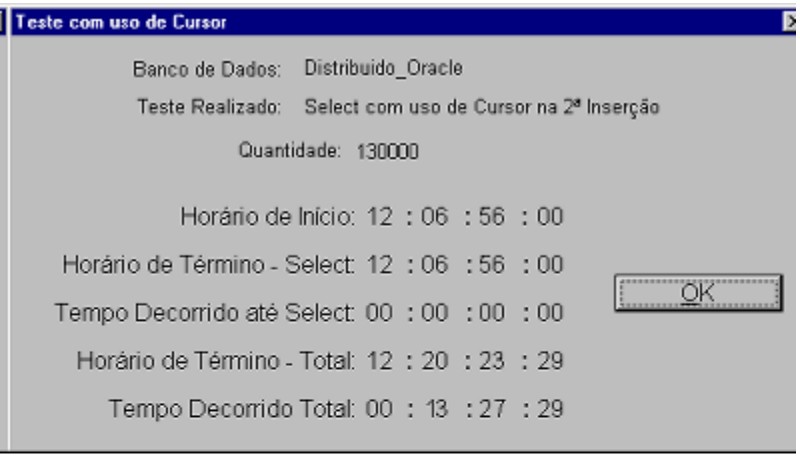


Resultado do 4° teste - Select sem Cursor

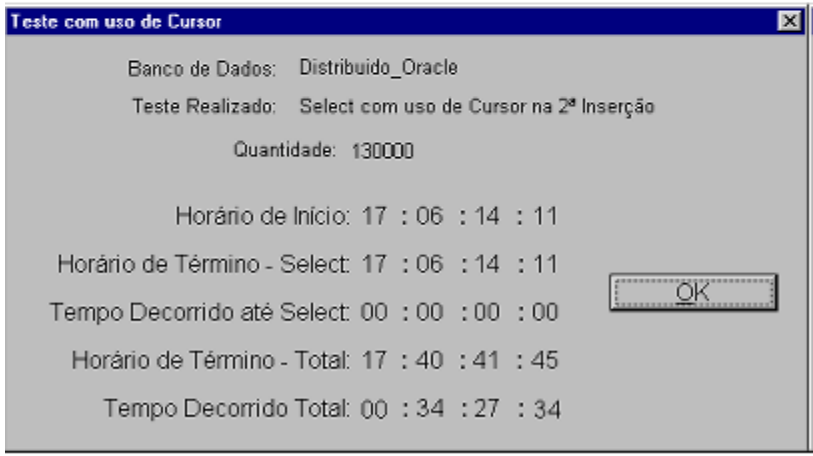
Resultados dos Testes sem o uso de Cursor



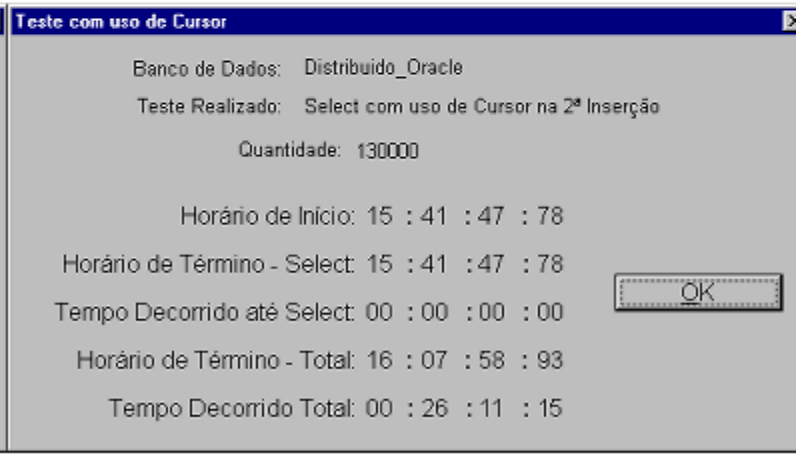
Resultado do 1º teste - Select com Cursor - 2ª inserção



Resultado do 2º teste - Select sem Cursor - 2ª inserção



Resultado do 3º teste - Select sem Cursor - 2ª inserção



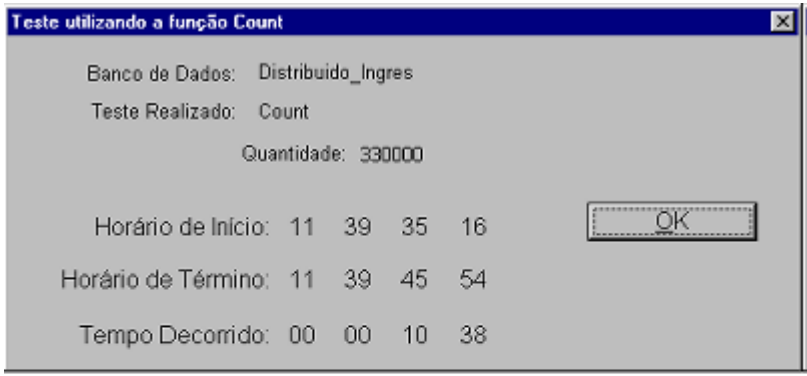
Resultado do 4º teste - Select sem Cursor - 2ª inserção

Resultados dos Testes sem Cursor - 2ª Inserção

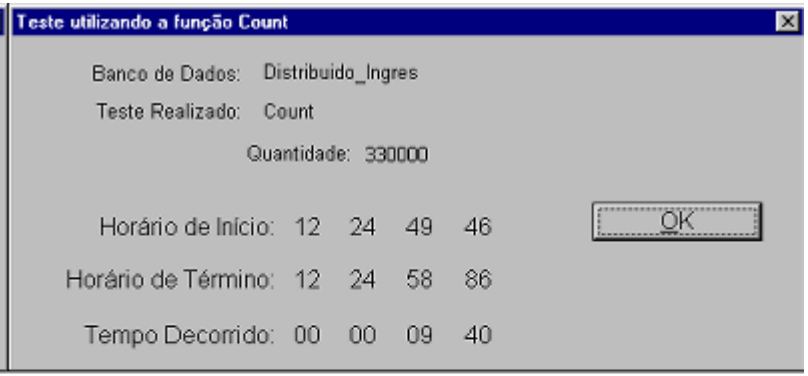
Teste	Início	Fim	Resultado
Função Count	09:28:15:10	09:28:20:43	05:33
	12:13:13:83	12:13:19:54	05:71
	17:34:13:62	17:34:19:44	05:82
	16:00:01:02	16:00:07:01	05:99
Select c/Cursor	09:29:04:21	09:32:43:36	03:39:15
	12:06:56:00	12:17:24:02	10:28:02
	17:06:14:11	17:38:13:10	31:58:99
	15:41:47:78	16:04:03:62	22:15:84
Select sem Cursor	09:33:08:02	09:33:26:04	18:02
	12:17:49:12	12:18:07:96	18:84
	17:38:38:36	17:38:56:98	18:62
	16:04:44:82	16:05:03:49	18:67
Select na 2ª inserção	09:29:04:21	09:35:14:68	06:10:47
	12:06:56:00	12:20:23:29	13:27:29
	17:06:14:11	17:40:41:45	34:27:34
	15:41:47:78	16:07:58:93	26:11:15

Resultado testes - SGBD Oracle

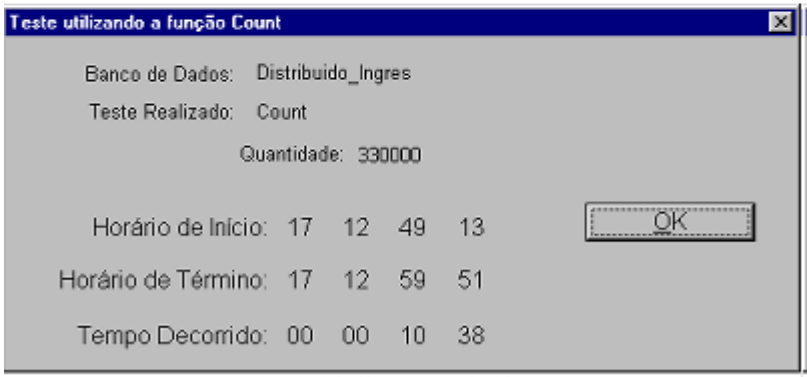
Anexo C
Testes com SGBD Ingres



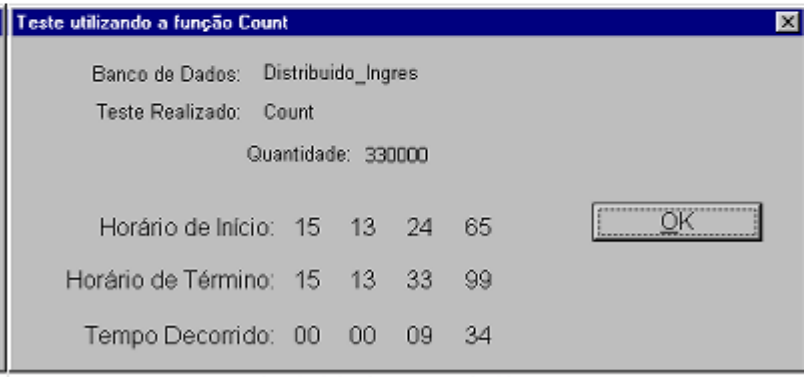
Resultado do 1º teste - função Count



Resultado do 2º teste - função Count

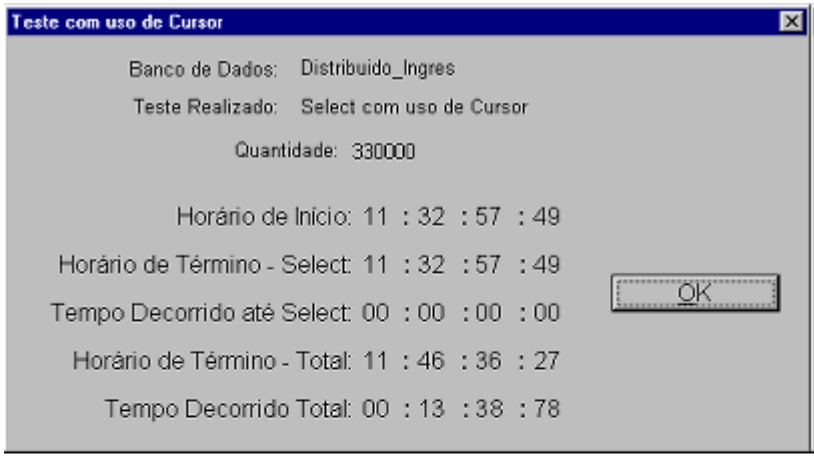


Resultado do 3º teste - função Count

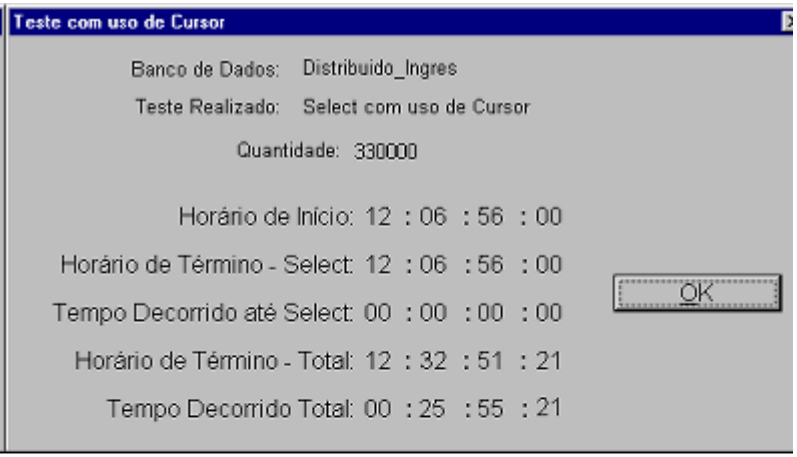


Resultado do 4º teste - função Count

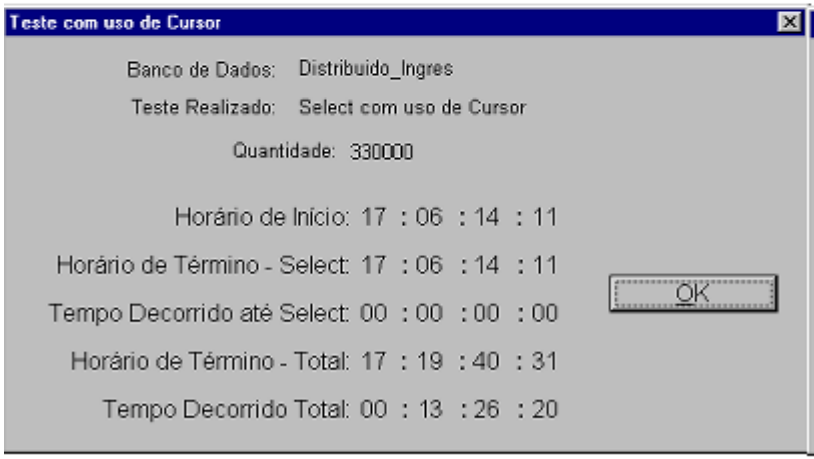
Resultados dos Testes com a Função Cursor



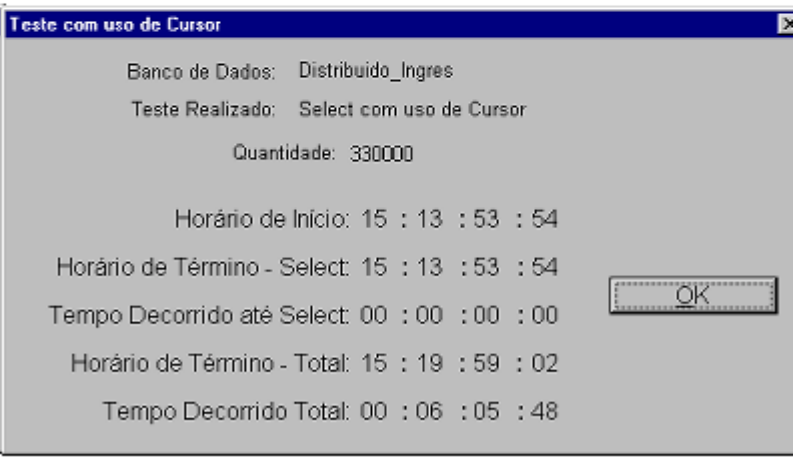
Resultado do 1° teste - Select com Cursor



Resultado do 2° teste - Select com Cursor

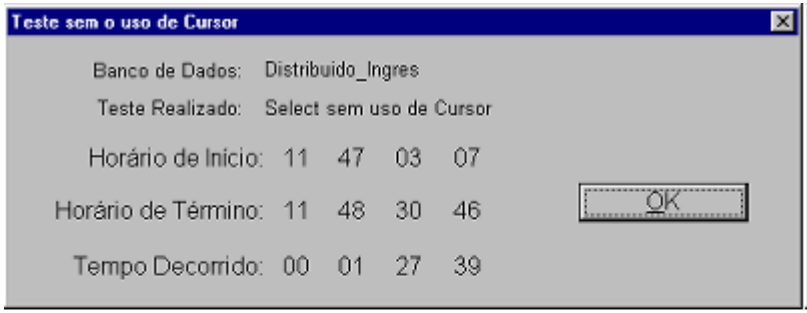


Resultado do 3° teste - Select com Cursor

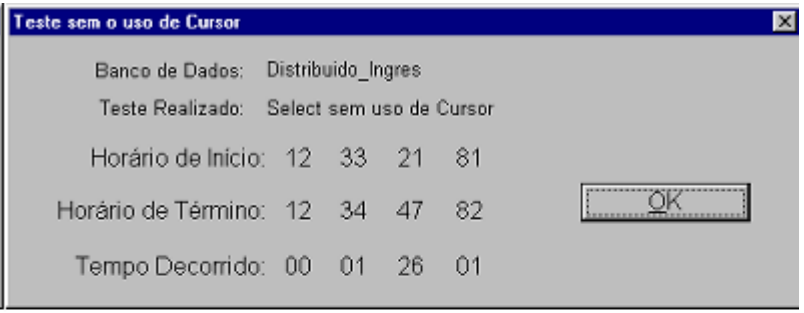


Resultado do 4° teste - Select com Cursor

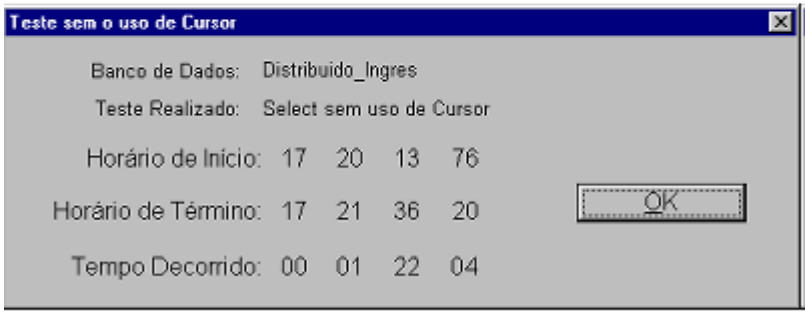
Resultados dos Testes de Seleção com Cursor



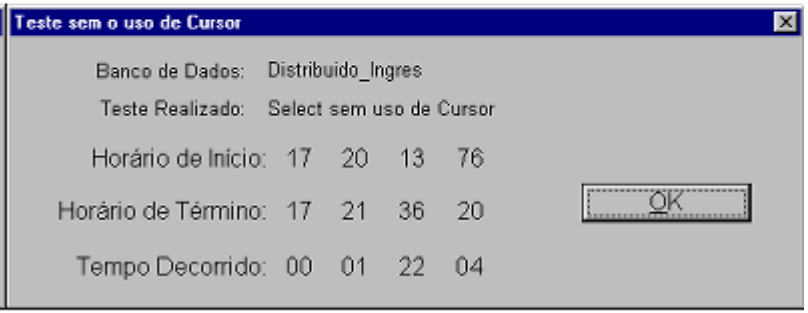
Resultado do 1° teste - Select sem Cursor



Resultado do 2° teste - Select sem Cursor

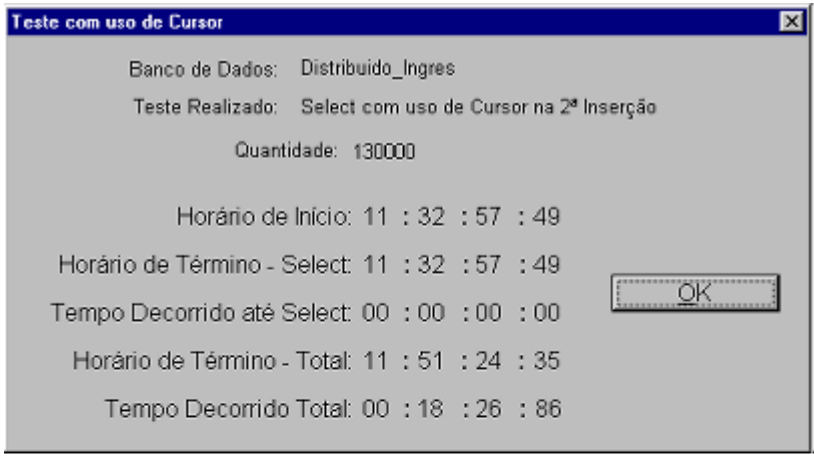


Resultado do 3° teste - Select sem Cursor

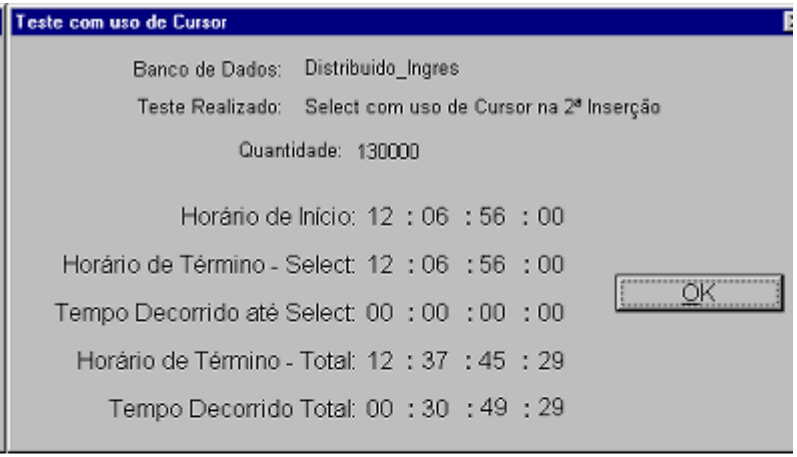


Resultado do 4° teste - Select sem Cursor

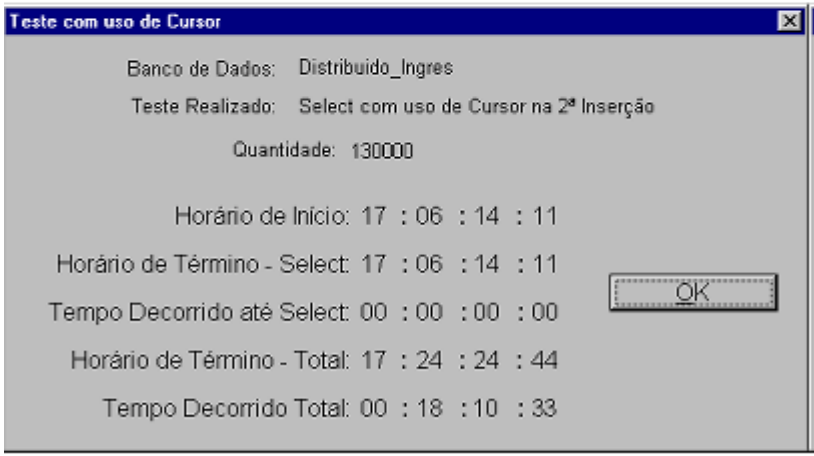
Resultados dos Teste de Seleção sem Cursor



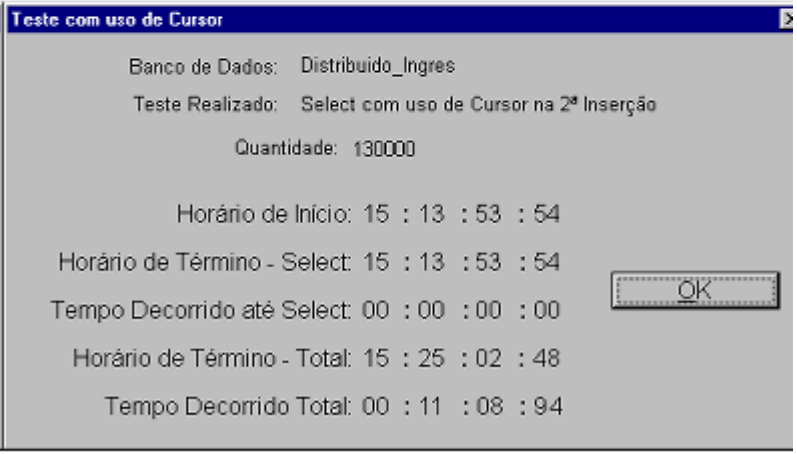
Resultado do 1ª teste - Select com Cursor - 2ª inserção



Resultado do 2ª teste - Select sem Cursor - 2ª inserção



Resultado do 3ª teste - Select sem Cursor - 2ª inserção



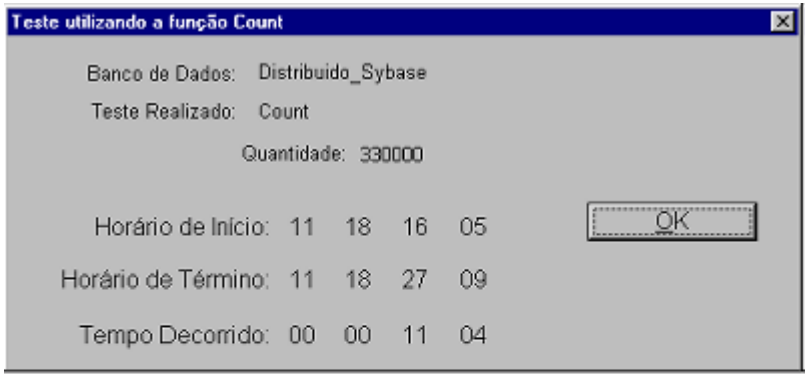
Resultado do 4ª teste - Select sem Cursor - 2ª inserção

Resultados dos Testes de Seleção sem Cursor - 2ª Inserção

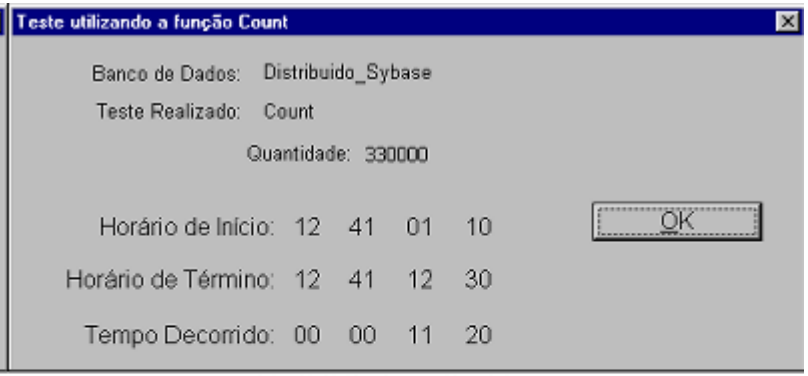
Teste	Início	Fim	Resultado
Função Count	11:39:35:16	11:39:45:54	10:38
	12:24:49:46	12:24:58:86	09:40
	17:12:49:13	17:12:59:51	10:38
	15:13:24:65	15:13:33:99	09:34
Select c/Cursor	11:32:57:49	11:46:36:27	13:38:78
	12:06:56:00	12:32:51:21	25:55:21
	17:06:14:11	17:19:40:31	13:26:20
	15:13:53:54	15:19:59:02	06:05:48
Select sem Cursor	11:47:03:07	11:48:30:46	01:27:39
	12:33:21:81	12:34:47:82	01:26:01
	17:20:13:76	17:21:36:20	01:22:04
	15:20:32:14	15:21:59:19	01:27:05
Select na 2ª inserção	11:32:57:49	11:51:24:35	18:26:86
	12:06:56:00	12:37:45:29	30:49:29
	17:06:14:11	17:24:24:44	18:10:33
	15:13:53:54	15:25:02:48	11:08:94

Resultado testes - SGBD Ingres

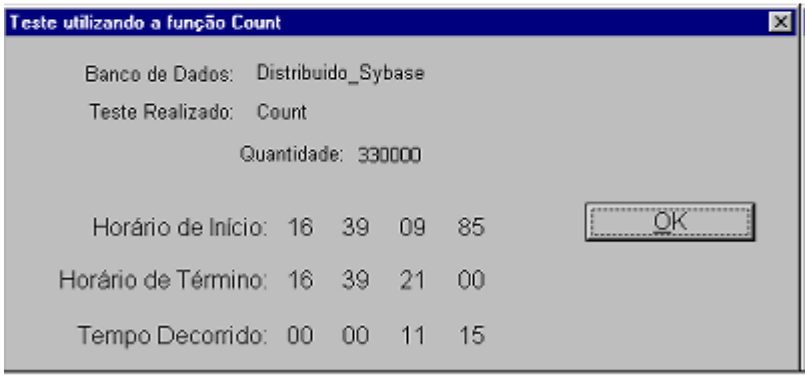
Anexo D
Testes com SGBD Sybase



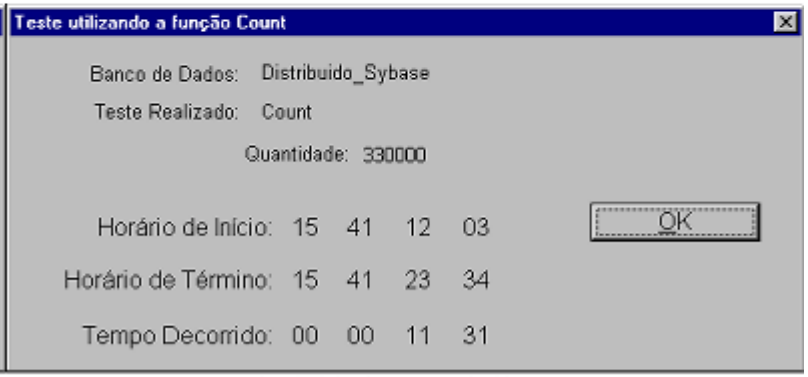
Resultado do 1º teste - função Count



Resultado do 2º teste - função Count

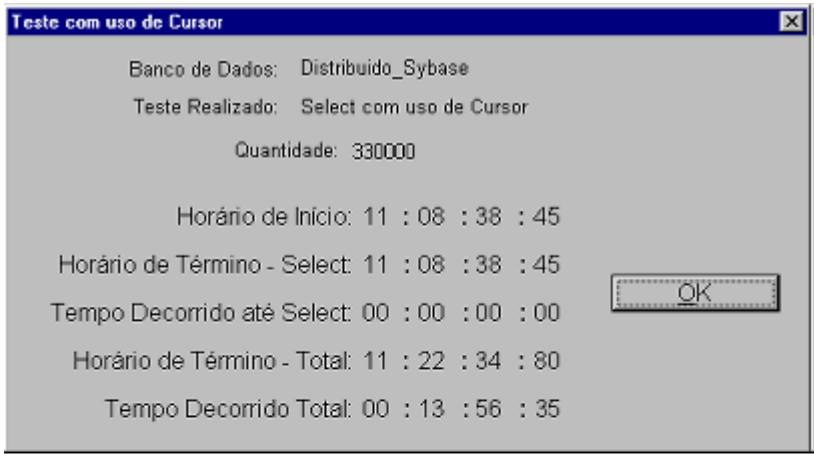


Resultado do 3º teste - função Count

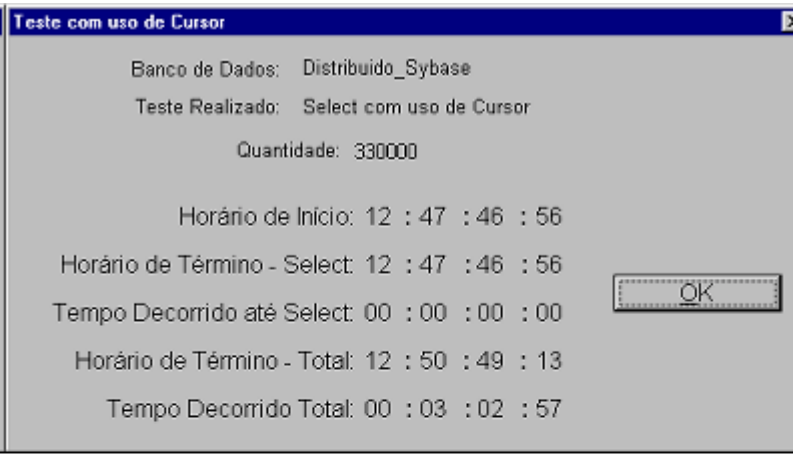


Resultado do 4º teste - função Count

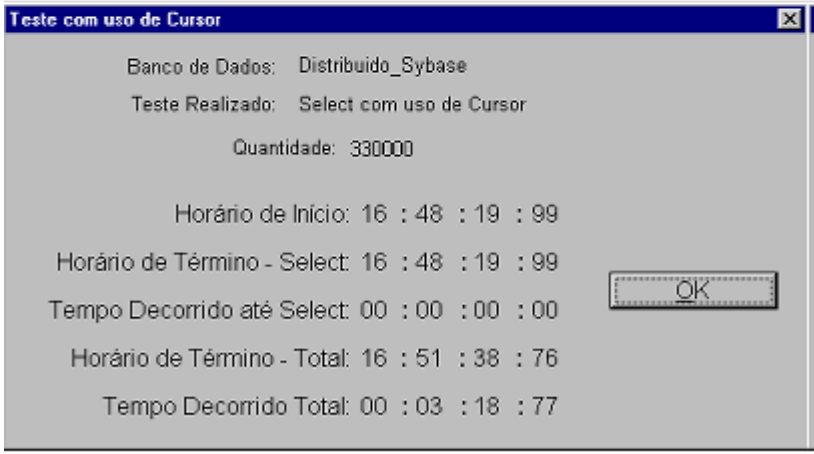
Resultados dos Testes da Função Count



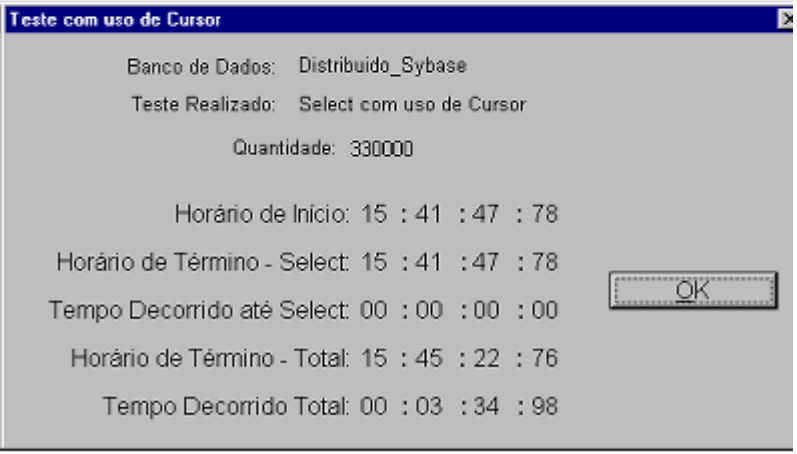
Resultado do 1° teste - Select com Cursor



Resultado do 2° teste - Select com Cursor

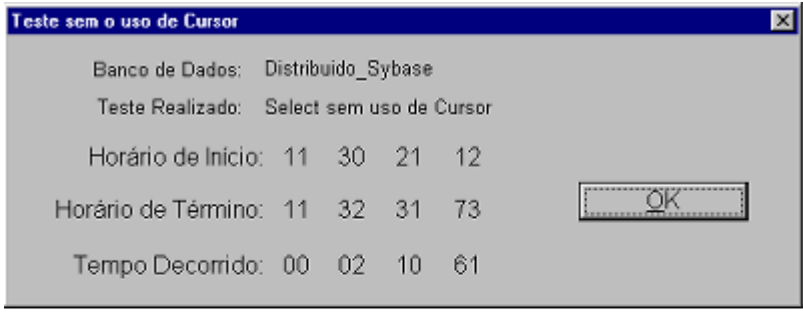


Resultado do 3° teste - Select com Cursor

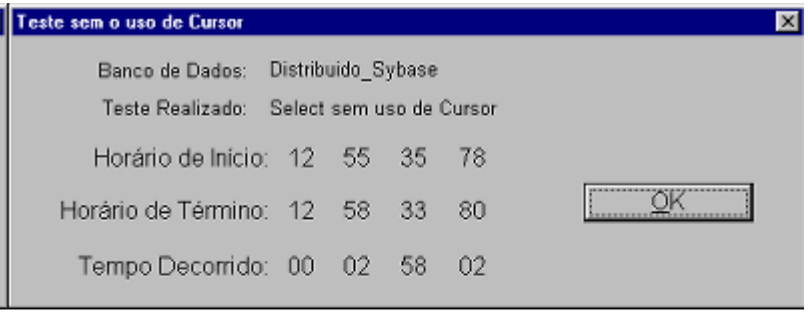


Resultado do 4° teste - Select com Cursor

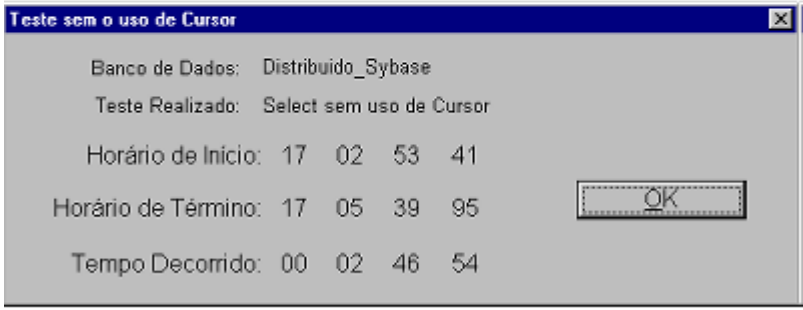
Resultados dos Testes de Seleção com Cursor



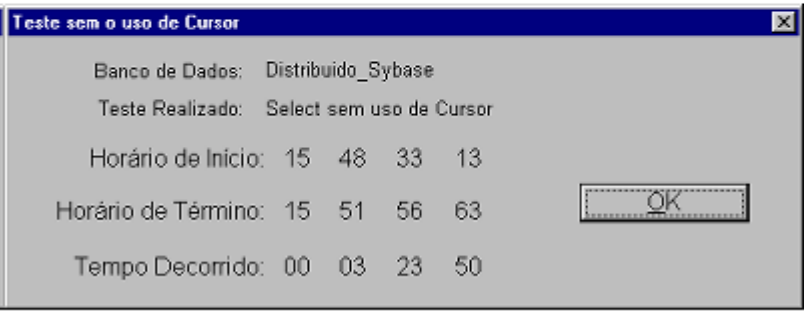
Resultado do 1° teste - Select sem Cursor



Resultado do 2° teste - Select sem Cursor

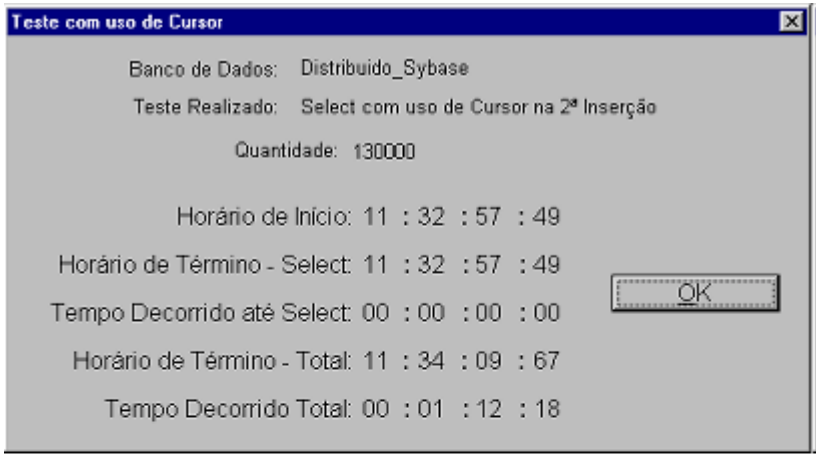


Resultado do 3° teste - Select sem Cursor

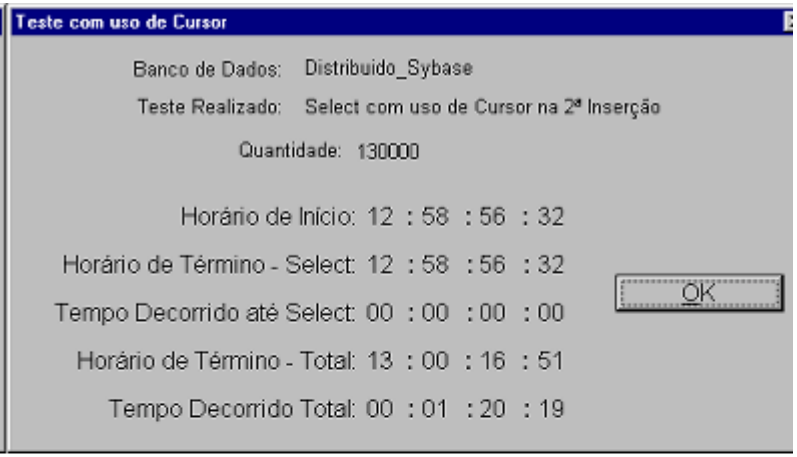


Resultado do 4° teste - Select sem Cursor

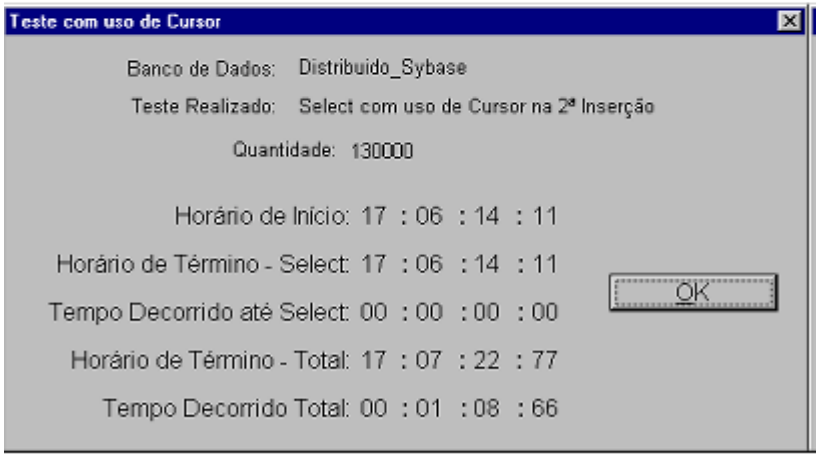
Resultados dos Testes de Seleção sem uso de Cursor



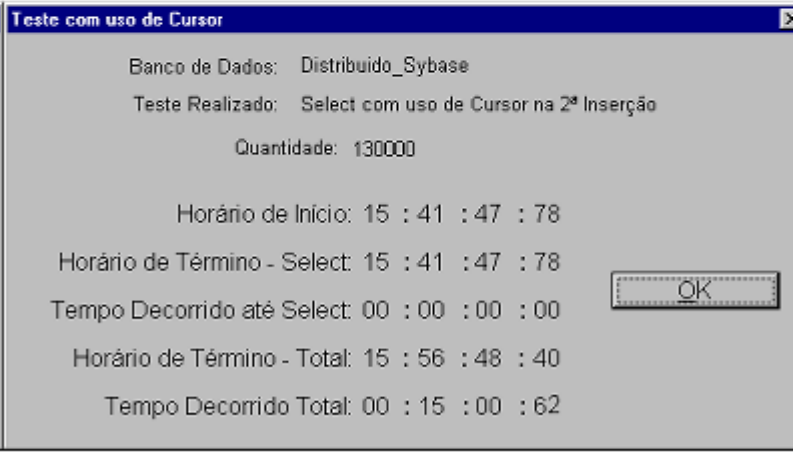
Resultado do 1º teste - Select com Cursor - 2ª inserção



Resultado do 2º teste - Select sem Cursor - 2ª inserção



Resultado do 3º teste - Select sem Cursor - 2ª inserção



Resultado do 4º teste - Select sem Cursor - 2ª inserção

Resultados dos Testes de Seleção sem Cursor - 2ª Inserção

Teste	Início	Fim	Resultado
Função Count	11:18:16:05	11:18:27:09	11:04
	12:41:01:10	12:41:12:30	11:20
	16:39:09:85	16:39:21:00	11:15
	15:41:12:03	15:41:23:34	11:31
Select c/Cursor	11:08:38:45	11:22:34:80	13:56:35
	12:47:46:56	12:50:49:13	03:02:57
	16:48:19:99	16:51:38:76	03:18:77
	15:41:47:78	15:45:22:76	03:34:98
Select sem Cursor	11:30:21:12	11:32:31:73	02:10:61
	12:55:35:78	12:58:33:80	02:58:02
	17:02:53:41	17:05:39:95	02:46:54
	15:48:33:13	15:51:56:63	03:23:50
Select na 2ª inserção	11:32:57:49	11:34:09:67	01:12:18
	12:58:56:32	13:00:16:51	01:20:19
	17:06:14:11	17:07:22:77	01:08:66
	15:41:47:78	15:56:48:40	15:00:62

Resultado testes - SGBD Sybase