

Sistemas de Gerência de Bancos de Dados Distribuídos  
são Factíveis?

Marco Antonio Casanova

Arnaldo Vieira Moura

**Palavras-chave:** sistemas de gerência de bancos de dados distribuídos, processamento de consultas, gerência de transações, controle de concorrência, controle de integridade.

MARCO ANTONIO CASANOVA é Pesquisador do Centro Científico da IBM em Brasília. Formou-se em Engenharia Eletrônica pelo Instituto Militar de Engenharia em 1974, obteve o grau de Mestre em Ciências pela Pontifícia Universidade Católica do Rio de Janeiro em 1976, e os graus de Mestre em Ciências (1977) e Doutor em Filosofia (1979), ambos em Matemática Aplicada, pela Universidade de Harvard. Seus interesses acadêmicos incluem teoria de banco de dados, sistemas de gerência de bancos de dados, e projeto e análise de algoritmos concorrentes.

ARNALDO VIEIRA MOURA é Pesquisador do Centro Científico da IBM em Brasília. Formou-se em Engenharia Eletrônica pelo Instituto Tecnológico de Aeronáutica em 1973, obteve o mestrado em Matemática também pelo mesmo Instituto em 1976 e o doutorado em Ciência da Computação pela Universidade da Califórnia, Berkeley, em 1980. Suas principais áreas de interesse incluem linguagens formais, complexidade de algoritmos e projeto e análise de algoritmos concorrentes.

Sistemas de Gerência de Bancos de Dados Distribuídos  
são Factíveis?

Marco Antonio Casanova  
Arnaldo Vieira Moura

Centro Científico de Brasília  
IBM do Brasil  
Caixa Postal 853  
70.000, Brasília, DF  
tel.(061)225-7670 r.5203

## 1. INTRODUÇÃO

Esta seção apresenta os conceitos mais básicos acerca de sistemas de gerência de bancos de dados distribuídos e desenvolve argumentos indicando quando bancos de dados distribuídos são uma alternativa interessante.

### 1.1. O que é um Sistema de Gerência de Bancos de Dados Distribuídos

Sistemas de gerência de bancos de dados distribuídos (SGBDDs) estendem as facilidades usuais de gerência de dados de tal forma que o armazenamento de um banco de dados possa ser dividido ao longo dos nós de uma rede de comunicação de dados, sem que com isto os usuários percam uma visão integrada do banco.

A criação de SGBDDs contribui de forma significativa para o aumento da produtividade em desenvolvimento de "software", um fator importante desde longa data. De fato, tais sistemas simplificam a tarefa de se definir aplicações que requerem a compartilhamento de informação entre usuários, programas ou organizações onde os usuários da informação, ou mesmo as fontes de informação, estão geograficamente dispersas. Aplicações com estas características incluem, por exemplo, sistemas de controle de inventário, contabilidade ou pessoal de grandes empresas, sistemas de consulta a saldos bancários, outros sistemas voltados para clientes e bancos de dados censitários.

A idéia de SGBDDs é atrativa sob muitos aspectos. Sob ponto de vista administrativo, tais sistemas permitem que cada setor de uma organização geograficamente dispersa mantenha controle de seus próprios dados, mesmo oferecendo compartilhamento a nível global no uso destes dados. Do ponto de vista econômico, SGBDDs podem diminuir os custos de comunicações, que hoje em dia tendem a ser maiores do que o próprio custo de equipamento, com o tradicional declínio dos custos de "hardware". Finalmente, SGBDDs também são atrativos de um ponto de vista técnico pois facilitam o crescimento modular do sistema (em contraste principalmente com um sistema centralizado de grande porte), aumentam a confiabilidade através da replicação das partes críticas do banco em mais de um nó, e podem aumentar a eficiência através de um critério judicioso de particionamento e replicação que coloque os dados próximos do local

onde são mais frequentemente usados (em contraste com acesso remoto a um banco de dados centralizado).

Por outro lado, não é difícil argumentar que sistemas com esta arquitetura levantam problemas de implementação sérios, têm um custo de desenvolvimento elevado, consomem recursos e podem ter uma performance duvidosa.

**Nota:** usaremos as seguintes abreviações:

- BDD - banco de dados distribuído;
- SGBD - sistema de gerência de bancos de dados (centralizado ou distribuído);
- SGBDD - sistema de gerência de bancos de dados distribuídos.

## 1.2. Por Que Bancos de Dados Distribuídos?

A arquitetura de sistemas utilizando banco de dados tem sido tradicionalmente centralizada. Ou seja, o banco de dados reside em um único computador onde são executados todos os programas acessando o banco. Os usuários podem, por sua vez, estar ligados diretamente, ou através de uma rede, ao computador onde o banco reside. Sistemas com esta arquitetura amadureceram durante a década de 70 e são instalados a uma razão da ordem de 10 por dia.

Usualmente, alinham-se a favor de sistemas centralizados argumentos que incluem fatores de economia de escala - o custo por unidade de trabalho decresce à medida que o tamanho do processador cresce (Lei de Grosch) - facilidade de controle de segurança, integridade e implantação de padrões, além de disponibilidade de dados para a gerência.

Estes argumentos precisam ser reavaliados, no entanto. Em primeiro lugar, a centralização dos dados e de responsabilidades choca-se com o objetivo de tornar os dados mais facilmente disponíveis ao usuário final em aplicações geograficamente dispersas. A relação entre os custos de processamento e de comunicações alteraram-se com a queda acentuada do custo de processadores, mas não do custo de transmissão de dados. Ou seja, a estratégia de trazer os dados a um processador central pode ser mais cara do que trazer capacidade computacional ao local de geração ou uso dos dados. Finalmente, sistemas centralizados apresentam vulnerabilidade maior a falhas e nem sempre permitem um crescimento gradativo da capacidade computacional instalada de forma simples e adequada.

Invertendo-se a discussão acima obtém-se argumentos a favor de bancos de dados distribuídos (BDDs).

Em detalhe, os argumentos que tornam BDDs atrativos podem ser postos da seguinte forma. BDDs podem refletir a estrutura organizacional ou geográfica do empreendimento dando maior autonomia e responsabilidade local ao usuário, mas preservando uma visão unificada dos dados. Do lado tecnológico, o desenvolvimento de redes de comunicação de dados permitiu a interligação de um grande número de processadores independentes de forma confiável e com custo previsível. Do ponto de vista puramente econômico, o preço/performance de equipamentos de menor porte tem melhorado substancialmente, obliterando o argumento a favor de equipamentos de grande porte. Além disto, BDDs podem diminuir os custos de comunicação se a maior parte dos acessos gerados em um nó puderem ser resolvidos localmente, sem acesso a dados armazenados em nós remotos. Finalmente, BDDs podem ser projetados de tal forma a melhorar a disponibilidade e confiabilidade do sistema através da replicação de dados, além de permitirem um crescimento modular da aplicação simplesmente acrescentando-se novos processadores e novos módulos do banco ao sistema.

Em contrapartida, o desenvolvimento de SGBDDs de uso genérico não é um problema simples. Em um SGBDD, o conhecimento do estado global do sistema é necessário para se processar consultas e para controle de concorrência, enquanto que não só os dados mas também o controle e informação sobre o estado do sistema estão distribuídos. Portanto, SGBDDs diferem significativamente de SGBDs centralizados do ponto de vista técnico, e um SGBDD não pode ser entendido como a simples replicação de SGBDs centralizados em vários nós.

### 1.3. Arquitetura Genérica para SGBDs Distribuídos

De um ponto de vista bem geral, um SGBD distribuído pode ser visto como uma federação de SGBDs centralizados, autônomos, chamados de SGBDs locais, que são interligados por uma camada de software chamada de SGBD da rede ou SGBD global ("network data base management system").

Um SGBD local é, para todos os efeitos, um SGBD centralizado gerenciando de forma autônoma o banco de dados local, exceto que poderá receber comandos tanto de usuários locais quanto da cópia local do SGBD global. O SGBD local faz uso do sistema operacional local que prove as seguintes facilidades básicas: métodos de acesso, gerência de processos e gerência de memória. (6)

A coletividade dos bancos locais constitui, então, uma implementação do banco distribuído.

O SGBD global roda como uma aplicação sob o sistema operacional da rede de comunicação de dados e, portanto, pertence à camada de aplicação na nomenclatura do modelo de referência da ISO. Isto significa que todos os problemas de comunicação de dados e distribuição de recursos é transparente ao SGBD global.

#### 1.4. Tipos de SGBDs Distribuídos

SGBDs distribuídos podem ser classificados em dois grandes grupos. Um SGBD distribuído será chamado de homogêneo (em "software") se os SGBDs locais são semelhantes, caso contrário será chamado de heterogêneo. (Uma classificação semelhante pode ser feita do ponto de vista de "hardware", mas não é importante). Mais precisamente, um SGBD distribuído é homogêneo se todos os seus SGBDs locais:

- oferecem interfaces idênticas ou, pelo menos, da mesma família;
- fornecem os mesmos serviços aos usuários em diferentes nós.

SGBDs distribuídos homogêneos aparecem com mais frequência quando a aplicação a que se destinam não existia antes. Conversamente, SGBDs distribuídos heterogêneos surgem usualmente quando há necessidade de integrar sistemas já existentes. A escolha entre uma arquitetura ou outra é influenciada pelo aproveitamento de "hardware" e "software" já existentes e pelo próprio hábito e grau de cooperação esperado dos usuários em caso de uma mudança para um sistema diferente. A alternativa óbvia, naturalmente, seria adotar uma arquitetura híbrida.

#### 1.5. Classes de Usuários de um SGBD Distribuído

Para apresentação de certas características de SGBDs distribuídos, convém classificar os seus usuários da seguinte forma. Por um lado, os usuários de um SGBD distribuído podem ser agrupados em:

- usuários globais, que observam o banco de dados distribuído como um todo e acessam os dados através das interfaces do SGBD global;
- usuários locais que têm contato apenas com o banco de dados local ao nó onde residem e interagem apenas com o SGBD local.

Ortogonalmente, os usuários de um SGBD (centralizado ou distribuído) podem ser classificados em quatro grandes grupos:

- o administrador do banco, ("database administrator") responsável pela definição e manutenção do banco (que naturalmente pode ser uma equipe; por tradição mantivemos aqui o singular);
- analistas e programadores de aplicação, responsáveis pelo desenvolvimento de aplicações sobre o banco de dados;
- usuários casuais, como gerentes, que usualmente não são programadores treinados e fazem uso do banco irregularmente;
- usuários paramétricos, como caixas de banco, que fazem uso do banco de dados através de transações paramétricas pré-programadas.

## 2. REQUISITOS FUNCIONAIS DE UM SGBD DISTRIBUÍDO

Do ponto de vista do usuário, um SGBD é uma ferramenta de "software" para armazenamento e acesso aos dados operacionais de um empreendimento. Um SGBD distribuído é utilizado quando a forma mais adequada de armazenamento dos dados é ao longo dos nós de uma rede. Um SGBD distribuído deverá facilitar a gerência dos dados em si, o desenvolvimento de aplicações relativas ao empreendimento, além de facilitar a utilização dos dados para fins de planejamento gerencial.

Nesta seção serão apresentadas as características funcionais de um SGBD distribuído necessárias para alcançar estes objetivos gerais. Os requisitos funcionais aqui apresentados terão grande influência no desenrolar dos capítulos seguintes, aplicando-se tanto a SGBDs centralizados, quanto a SGBDs distribuídos, exceto em certos casos. Muitos dos requisitos selecionados, embora originalmente introduzidos para SGBDs seguindo o modelo relacional de dados, refletem preocupações mais gerais e independentes do modelo usado.

### 2.1. Independência Física de Dados

A forma como o banco de dados está armazenado não deve ser visível aos programadores e analistas de aplicação, muito menos aos usuários casuais, sendo responsabilidade exclusiva do administrador do banco defini-la. Em outros termos, os detalhes de armazenamento do banco devem ser transparentes (ou mesmo irrelevantes) ao desenvolvimento de programas de aplicação e ao uso casual do banco, já que a este nível apenas a forma com que os dados estão logicamente estruturados importa. Além disto, espera-se que um bom sistema de gerência de banco de dados permita mudar a forma de armazenar o banco sem alterar os programas de aplicação.

Ou seja, deve ser possível alcançar o que se convencionou chamar de independência física de dados.

### 2.2. Independência de Localização e Replicação

Em um SGBD distribuído, independência física de dados adquire um significado especial. Este requisito exige que o fato do banco ser distribuído seja um problema de implementação e, portanto, transparente aos usuários (exceto, é claro, na variação do tempo de acesso). Isto significa que devem ser transparentes aos usuários tanto a localização das várias partes do banco de dados ("location transparency"), quanto ao fato destas partes estarem replicadas ou não ("replication transparency"). O sistema deve, então, ser responsável por localizar os dados e atualizar todas as cópias. Além disto, se os arquivos forem movidos de um nó para

outro, ou divididos, os usuários não devem tomar conhecimento do fato (estas são formas de reestruturar um banco de dados distribuído).

Em resumo, *os usuários globais deverão ver o banco de dados distribuído como se fosse centralizado.*

Chamaremos estas características de independência de localização e replicação.

### 2.3. Autonomia Local

Este requisito está intrinsecamente ligado à estruturação de um SGBD distribuído em uma federação de SGBDs locais autônomos interligados pelo SGBD global. Mais precisamente, na arquitetura genérica adotada exige-se que cada SGBD local mantenha sua autonomia, no seguinte sentido:

- cada SGBD local deve manter controle sobre seus próprios dados, pois uma das motivações para banco de dados distribuídos era justamente a distribuição da responsabilidade dos dados para os próprios usuários locais;
- programas que acessem dados locais devem ser executados localmente, sem que seja necessário consultar outros nós. Não deve haver, portanto, um controle central do banco, nem os dados necessários ao funcionamento do sistema devem ser centralizados (como em um diretório único centralizado).

Como consequência deste requisito, *um usuário local deverá acessar os dados locais como se constituíssem um banco de dados centralizado independente.*

### 2.4. Interfaces de Muito Alto Nível

A linguagem para acesso aos dados armazenados no banco deve ser de muito alto nível, ou seja, com as seguintes características:

- a linguagem deve ser não-procedural no sentido do usuário especificar *que* dados devem ser acessados e não *como* eles devem ser acessados (isto é problema do sistema);
- os comandos de acesso ao banco, oferecidos pela linguagem, devem manipular conjuntos de objetos e não apenas um objeto de cada vez;
- os comandos devem ser completamente independentes dos detalhes de armazenamento do banco e da existência de caminhos de acesso pré-definidos.

Estas características são evidentemente importantes para usuários casuais, com pouco treinamento em processamento de dados. Mas há duas outras razões de peso para se exigir interfaces de alto nível. Primeiro, a produtividade de analistas e programadores de aplicação aumentará, pois poderão se concentrar primordialmente na aplicação em si e não na sua

implementação (a situação é a mesma quando linguagens de programação de alto nível começaram a aparecer). Segundo, linguagens de muito alto nível podem potencialmente aumentar a eficiência do sistema, no seguinte sentido. Cada comando para acesso ao banco exige a intervenção do SGBD, o que gera um custo adicional considerável. Se o comando manipula conjuntos de objetos de cada vez, este custo adicional é, portanto, diluído em um volume maior de trabalho útil. Este argumento é especialmente importante quando o comando gera acessos a dados remotos exigindo o envio de mensagens através da rede.

## 2.5. Otimização Automática

O uso de interfaces de alto nível perderia o impacto se o processamento de comandos para acesso aos dados fosse ineficiente. O SGBD deve, portanto, conter um otimizador para selecionar os caminhos de menor custo para acessar os dados.

A construção de otimizadores eficientes foi, de fato, um dos principais problemas enfrentados no projeto de SGBDs recentes, especialmente aqueles seguindo o modelo relacional. As soluções apresentadas provaram, no entanto, serem bastante satisfatórias, viabilizando assim o uso de interfaces de alto nível.

## 2.6. Reestruturação Lógica do Banco e Suporte a Visões

Os requisitos de independência física de dados e independência de localização e replicação implicam em que a forma de armazenamento do banco pode ser modificada sem que seja necessário alterar os programas de aplicação. No entanto, reestruturações deste tipo são necessárias para otimizar a forma de armazenamento quando o perfil de utilização do banco muda.

Por outro lado, modificações nas estruturas lógicas do banco (ou seja, na forma como os usuários vêem a estruturação dos dados) são necessárias quando a aplicação muda conceitualmente. O SGBD deve, então, fornecer meios para modificar a estrutura lógica de um banco já existente e criar a nova versão dos dados a partir da antiga.

Reestruturações deste tipo podem causar impacto nos programas de aplicação. Uma estratégia para minorar o impacto de tais mudanças seria criar visões através das quais os programas de aplicação acessam o banco. Assim, se a estrutura lógica do banco mudar, em certos casos basta adaptar a definição das visões, sem que com isto os programas de aplicação recebam o impacto das mudanças. É interessante então que o SGBD suporte o mecanismo de visões em complemento a reestruturações no banco

## 2.7. Segurança dos Dados

Uma aplicação baseada em um banco de dados facilita enormemente o acesso aos dados operacionais do empreendimento, o que traz o efeito adverso de facilitar acessos não autorizados a dados classificados. O SGBD deverá, necessariamente, prover meios para definir critérios de autorização para acesso aos dados e meios para assegurar que as regras de acesso serão cumpridas.

## 2.8. Suporte à Administração dos Dados

Um banco de dados é, em geral, uma estrutura complexa com centenas de tipos de objetos diferentes, armazenados de diversas formas. A tarefa de administrar um banco, especialmente se é distribuído, exige ferramentas especiais para ser efetivamente executada. O SGBD deve, então, fornecer um dicionário ou diretório, onde é armazenada a descrição do banco, ferramentas para acesso a este dicionário, além de utilitários para manutenção do banco.

Em especial, o acesso ao dicionário não deverá ser exclusividade do administrador do banco, já que é importante que os usuários casuais, programadores e analistas de aplicação conheçam a definição dos tipos de objetos armazenados no banco.

# 3. ESPECIFICAÇÃO DAS INTERFACES DE UM SGBD DISTRIBUÍDO

Nesta seção serão estudadas as características das interfaces oferecidas tanto a usuários locais quanto a usuários globais, e os efeitos sobre as interfaces resultantes do SGBD distribuído ser homogêneo ou heterogêneo.

## 3.1. Interfaces Globais e Locais

Conforme visto, um SGBD distribuído é constituído de uma coleção de SGBDs locais interligados por um SGBD global. Em cada nó, os usuários locais são servidos pelo SGBD local implementado naquele nó, e os usuários globais (residentes naquele nó) são servidos pela cópia local do SGBD global. Há, portanto, duas classes de interfaces em um SGBD distribuído:

- as interfaces globais, oferecidas pelo SGBD global aos usuários globais;
- as interfaces locais, oferecidas pelos SGBDs locais aos usuários locais.

Como consequência de dois dentre os requisitos básicos que SGBDs distribuídos devem satisfazer, a especificação destas duas classes de interfaces se confunde, no entanto, com a descrição das interfaces de um SGBD centralizado. De fato, lembremos que o requisito de Independência de Localização e Replicação implica em que os usuários globais de um SGBD distribuído deverão ver o banco de dados distribuído como se fosse centralizado. Logo, o SGBD global deverá se comportar como um SGBD centralizado perante estes usuários. Já o requisito de Autonomia Local implica em que um usuário local deverá acessar os dados locais como se constituíssem um banco de dados centralizado independente. Ou seja, o SGBD local é, para efeito dos usuários locais, um SGBD centralizado autônomo.

Assim, para especificar as características das interfaces oferecidas tanto a usuários locais quanto a usuários globais, basta estudar os tipos de interfaces comumente oferecidas por SGBDs centralizados. Este será o assunto do parágrafo seguinte.

### 3.2. Especificação das Interfaces

Recordemos que os usuários locais ou globais podem ser classificados em quatro grandes grupos: o administrador do banco, analistas e programadores de aplicação, usuários casuais, e usuários paramétricos. Para satisfazer as necessidades destas classes de usuários, tradicionalmente um SGBD centralizado oferece:

- uma linguagem de definição de dados (LDD) usada para definir novos bancos de dados;
- uma ou mais linguagens de manipulação de dados (LMDs) usadas para recuperar e modificar os dados armazenados no banco;
- opcionalmente, uma linguagem de geração de relatórios (LGR) que, como o nome indica, é apropriada para extrair relatórios do banco de dados;
- utilitários para manutenção do banco.

A LDD conterá comandos para definir as estruturas lógicas do banco e indicar como estas deverão ser armazenadas fisicamente. A LDD poderá também conter outros tipos de comandos como, por exemplo, comandos para definir critérios de autorização de acesso aos dados. A LDD e os utilitários são as ferramentas de que dispõe o administrador do banco para executar as suas tarefas.

Uma LMD pode ser oferecida como uma linguagem independente para acesso ao banco através de terminais, ou ser oferecida como uma extensão de uma linguagem de programação já existente, chamada de linguagem hospedeira. A primeira versão é apropriada à formulação de acessos não antecipados em geral, típicos de usuários casuais. A segunda versão é utilizada no desenvolvimento de programas de aplicação que implementam transações repetitivas, definidas a priori (como desconto de cheques, transferência de fundos, ou consulta a saldo).

Uma LMD, tipicamente, conterá comandos para recuperar, inserir, remover e atualizar dados armazenados no banco. Outros comandos de controle também existirão indicando ao sistema quando começa e termina uma transação, quando os dados de uma transação podem se tornar visíveis a outros usuários, etc.

A LDD e as LMDs oferecidas por um SGBD são baseadas no mesmo modelo de dados, que fixa os tipos de estruturas lógicas, como árvores, tabelas, etc... que serão usados para organizar o banco de dados do ponto de vista conceitual.

Um usuário paramétrico não utiliza diretamente as interfaces do SGBD, mas sim transações paramétricas pertencentes a uma aplicação desenvolvida sobre o banco de dados. Uma transação é por sua vez definida por um programa de aplicação (um procedimento com parâmetros de entrada) escrito em uma determinada linguagem hospedeira e contendo comandos da LMD. Em geral o termo "transação" tem o sentido mais preciso de uma coleção de comandos que deve ser processada pelo SGBD como se fosse atômica, ou seja, o banco deve refletir o resultado de todos os seus comandos ou de nenhum deles.

Isto conclui a discussão sobre as interfaces de um SGBD.

### 3.3. Influência do Tipo de SGBD Distribuído sobre as Interfaces

Como em um SGBD distribuído homogêneo todos os SGBDs locais oferecem interfaces idênticas, estes últimos usam, então, o mesmo modelo de dados, a mesma LDD e as mesmas LMDs. Logo, uma vez fixadas as interfaces locais, é natural que o SGBD global também ofereça estas mesmas interfaces. Assim, qualquer usuário, local ou global, poderá acessar tanto dados locais quanto dados remotos através da mesma linguagem de manipulação de dados.

Este não é o caso, porém, para sistemas heterogêneos pois SGBDs locais potencialmente usam modelos de dados e LMDs diferentes. Uma opção seria o SGBD da rede oferecer ao usuário global, residente em um dado nó, uma visão do banco de dados distribuído no mesmo modelo de dados que o banco local, e permitir que este usuário acesse dados definidos nesta visão através da própria LMD local. Esta opção é interessante pois não é necessário ensinar uma nova LMD aos usuários residentes em um determinado nó para que possam acessar dados remotos.

Nesta opção, o SGBD global possui, na verdade, uma interface diferente para cada nó. Isto não quer dizer que o SGBD global não suporte uma LMD independente das LMDs oferecidas pelos SGBDs locais, chamada LMD pivot, quer seja pela existência de uma nova classe de usuários globais, quer seja para simplificar a estruturação do sistema.

## 4. PRINCIPAIS FUNÇÕES DE UM SGBD DISTRIBUÍDO

As principais funções de um SGBD (centralizado ou distribuído) podem ser grupadas em sete grandes módulos:

- Armazenamento do Diretório de Dados
- Armazenamento do Banco de Dados
- Processamento de Comandos da Linguagem de Manipulação de Dados
- Gerência de Transações
- Controle de Integridade
- Controle de Concorrência
- Controle de Acesso ao Banco

Este agrupamento de funções corresponde diretamente à organização do livro, sendo cada função discutida em um capítulo separado.

Nesta seção, cada uma destas funções será brevemente abordada, criando-se assim um guia para leitura dos outros capítulos. Como a discussão é bastante genérica, aplica-se tanto a SGBDs centralizados quanto distribuídos, exceto em certos casos. Antes, porém, um refinamento da arquitetura de SGBDs distribuídos descrita na Seção 1.1.3, será apresentado, bem como um esquema simplificado do ciclo de acesso a um banco de dados distribuído.

### 4.1. Refinamento da Arquitetura de um SGBD Distribuído

De acordo com a arquitetura genérica adotada, um SGBD distribuído consiste de uma coleção de SGBDs locais interligados pelo SGBD global. O SGBD global pode, por sua vez, ser refinado em três grandes componentes:

1. diretório de dados global (DDG): contém a descrição do banco de dados distribuído. O critério usado para sua distribuição/duplicação é crucial para a performance do sistema;
2. gerente de transações (GT): interpreta e controla o processamento de consultas e transações acessando o BDD;
3. gerente de dados (GD): interface com o SGBD local, fazendo as traduções necessárias no caso de sistemas heterogêneos.

Cada SGBD local possui também um diretório de dados local descrevendo o banco de dados local.

Cada nó da rede conterá então um SGBD local e uma cópia do SGBD global, caso armazene parte do banco, ou apenas o gerente de transações, caso não armazene parte do banco. Um nó poderá conter ou não parte do diretório global, já que a estratégia de alocação deste último não é fixada "a priori" neste nosso cenário.

#### 4.2. Ciclo de Processamento em um SGBD Distribuído

Um ciclo típico de acesso ao banco de dados distribuído na arquitetura descrita na seção anterior seria (ver Figura 1):

1. uma transação  $T$  operando no nó  $i$  (ou usuário acessando o banco através do nó  $i$ ) executa um comando para acessar o banco;
2. o gerente de transações do nó  $i$  intercepta o comando, acessa o diretório global (que pode estar em outro nó) e cria um plano de acesso ao BDD para obter os dados necessários, ou seja, cria uma seqüência de comandos a serem enviados aos outros nós e para o próprio banco local;
3. o gerente de transações do nó  $i$  envia comandos aos nós envolvidos e coordena a sua execução;
4. o gerente de dados de um nó  $j$  envolvido no processamento recebe comandos para o banco local e se encarrega de chamar o SGBD local para executá-los (se for necessário, o gerente de dados traduz os comandos para a linguagem de manipulação de dados local);
5. o gerente de dados do nó  $j$  devolve os dados pedidos ao gerente de transações do nó  $i$ ;
6. o gerente de transações do nó  $i$  completa o processamento do comando submetido, passando os dados para a transação (ou para o usuário).

O acesso ao banco de dados local seguiria o seguinte ciclo esquematizado:

1. uma transação local executa um comando para acessar o banco local, ou o gerente de dados repassa um comando remoto ao SGBD local;
2. SGBD local, por intermédio do sistema operacional, acessa o diretório de dados local;
3. SGBD local analisa o comando;

4. SGBD local chama o sistema operacional para transferir dados do banco de dados local para memória principal;
5. SGBD local extrai dos dados transferidos aqueles necessários para responder ao comando.

O resto desta seção discute quais são as principais funções de um SGBD distribuído e como elas são implementadas.

#### 4.3. Armazenamento do Diretório de Dados

O diretório de dados, na sua forma mais simples, contém toda informação sobre o sistema e sobre os bancos de dados e transações já definidos. Esta informação é armazenada sob forma interna para ser usada pelos outros componentes do SGBD e, naturalmente, é fundamental para o funcionamento do sistema.

Este papel básico do diretório pode ser expandido com outras facilidades para que se torne a principal ferramenta de administração do banco e a principal fonte de informação sobre o significado dos dados para os usuários.

Nota: em todo este texto, usaremos o termo dicionário de dados para um subsistema cobrindo ambos os tipos de facilidades, reservando o termo diretório de dados para um subsistema que abranja apenas as funções básicas.

Em um SGBD distribuído haverá um diretório de dados global, que descreve o banco de dados distribuído e é usado pelas cópias do SGBD global, e um diretório local para cada SGBD local, descrevendo o banco local. Independentemente do fato de ser global ou local, as seguintes observações podem ser feitas acerca do diretório.

A linguagem de definição de dados (LDD) implementada pelo SGBD constitui-se na principal interface com o dicionário de dados, no seguinte sentido. O diretório contém, inicialmente, apenas informação sobre as suas próprias estruturas e outras transações e usuários especiais. A descrição de novos bancos de dados é então acrescentada ao diretório como resultado do processamento de comandos da LDD; novos usuários são catalogados através de comandos de autorização da LDD; e novas transações são acrescentadas também através de comandos especiais de catalogação.

Há duas formas de implementar o diretório. A rigor, o diretório nada mais é do que um banco de dados e um conjunto de transações e, portanto, pode ser implementado como um banco de dados usando os mesmos mecanismos que os bancos comuns. Ganha-se com isto em uniformidade de interfaces e em compartilhamento de código, já que a própria linguagem de manipulação de dados do SGBD pode ser utilizada para manter o diretório. Por outro lado, perde-se em performance.

De fato, o diretório, embora sendo um banco de dados, é acessado com extrema frequência. Portanto merece técnicas de armazenamento e acesso especiais. Com isto ganha-se em performance, mas perde-se em complexidade.

Estas últimas observações são especialmente importantes no que se refere ao diretório global. Como qualquer outro banco, ele poderá ser distribuído com replicação ou particionamento. No entanto, como acessos ao diretório são necessários a cada consulta, a sua implementação deve ser cuidadosa para minimizar o tráfego adicional de mensagens gerado para acessá-lo.

#### 4.4. Armazenamento do Banco de Dados

A função de armazenamento do banco de dados se refere às estruturas de arquivo, métodos de acesso, técnicas de compressão, etc., oferecidas como opções para organização física dos dados em memória secundária. Esta é uma função dos SGBDs locais já que, uma vez definidos os critérios de distribuição do banco, o problema de armazenamento físico é puramente local.

As opções de armazenamento oferecidas pelo SGBD tornam-se aparentes sob forma de comandos especiais da LDD usados para descrição da organização física do banco de dados. Esta parte da linguagem é, às vezes, denominada de linguagem de definição interna de dados.

A implementação do subsistema de armazenamento interage fortemente com o subsistema responsável pelo processamento de comandos, considerando-se que o problema básico de um SGBD local consiste em receber um comando do usuário (local ou remoto) e traduzí-lo em acessos físicos a registros armazenados em memória secundária. Este problema é exacerbado quando a linguagem de manipulação de dados usada é de muito alto nível, conforme um dos requisitos funcionais impostos a SGBDs distribuídos.

Neste contexto, a arquitetura do subsistema de armazenamento pode ser dividida em dois níveis. O nível mais baixo, que chamaremos de nível físico, consiste de uma coleção de métodos de acesso primários cujas principais funções são:

- esconder dos subsistemas superiores todos os detalhes de armazenamento referentes aos periféricos;
- oferecer alternativas básicas para armazenamento e recuperação de registros físicos, através da definição de estruturas de dados e algoritmos para manipulação destas estruturas.

Este primeiro nível, na verdade, não costuma ser parte do SGBD. Ou seja, o SGBD aproveita os próprios métodos de acesso primários do sistema operacional subjacente, talvez complementando-os com outros mais sofisticados.

O segundo nível do subsistema de armazenamento, que chamaremos de nível interno ou lógico, pode ser encarado como um SGBD independente, definido com os seguintes objetivos:

- oferecer aos subsistemas superiores uma interface melhor do que simplesmente chamadas para as rotinas dos métodos de acesso, mas bem mais simples do que a LMD do sistema completo;
- implementar a base de outras funções do sistema como, por exemplo, controle de integridade.

Este segundo nível oferece então um passo intermediário entre os comandos da LMD do sistema e as rotinas dos métodos de acesso.

#### 4.5. Processamento de Comandos da Linguagem de Manipulação de Dados

Por processamento de comandos da linguagem de manipulação de dados (LMD) entenderemos o problema de processar comandos tanto para recuperação de dados quanto para atualização do banco, formulados por um usuário de forma interativa, ou resultantes da execução de transações. Mas excluiremos o problema de implementar o conceito de transação como coleção atômica de comandos, que é discutido na seção seguinte.

Independentemente do sistema ser centralizado ou distribuído, o processamento de comandos poderá ser feito de forma interpretativa, ou através de compilação. A estratégia de compilação provou ser bastante mais eficiente, mas cria certas dificuldades adicionais. De fato, considere o seguinte cenário: um comando é compilado, assumindo uma dada organização física do banco, e o código objeto armazenado para posterior execução; o banco de dados tem sua organização alterada; o código objeto é executado, resultando em erro (pois a organização do banco mudou). Para evitar este problema e manter a capacidade de reestruturar dinamicamente o banco é necessário, então, um mecanismo de validação em tempo de execução que recompile automaticamente o comando caso necessário. Note que, em uma estratégia interpretativa, este problema não ocorre.

Em um SGBD distribuído, o processamento de comandos é subdividido nas seguintes fases:

1. análise sintática: inclui a tradução dos nomes das estruturas do banco de dados para uma forma interna com auxílio do diretório de dados;
2. otimização: seleciona um plano de execução para o comando;
3. distribuição do plano de execução;
4. execução dos subcomandos pelos SGBDs locais e criação do resultado final.

Há três tarefas particularmente difíceis. A primeira, otimização, requer escolher que cópias dos dados deverão ser usadas, fragmentar o comando original em subcomandos e definir a estratégia de movimentação dos resultados parciais (um extremo seria, por exemplo, mover o resultado de todos os subcomandos para um dado nó, que não precisa ser o nó em que o usuário está, e aí resolver a consulta).

A segunda tarefa, execução dos subcomandos por cada SGBD local, coincide com o problema de processar comandos em um SGBD centralizado. Como esta é uma tarefa complexa, o subsistema correspondente costuma ser estruturado em várias camadas, ou máquinas virtuais, cada uma implementando uma interface que se assemelha a uma LMD de mais alto nível que a anterior. Conjugando-se esta arquitetura com a discussão sobre armazenamento de bancos, a máquina mais interna corresponderia então ao nível físico do subsistema de armazenamento, a máquina imediatamente superior corresponderia ao nível interno do subsistema de armazenamento, e assim por diante até a última máquina, que ofereceria como interface a própria LMD do sistema.

A terceira tarefa que merece comentários nesta fase introdutória surge quando o SGBD distribuído é heterogêneo e, em cada nó, o SGBD global oferece a própria LMD local como interface. Isto significa que comandos para acessar o banco de dados distribuído poderão ser formulados em várias LMDs diferentes. Cada um destes comandos deverá, então, ser fragmentado em subcomandos que serão, por sua vez, traduzidos para a LMD do SGBD local em que serão processados.

Esta tarefa de tradução é função do gerente de dados local. Supondo-se que o banco esteja armazenado em  $n$  nós, serão necessários  $n(n - 1)$  tradutores, já que deverá haver um tradutor para cada par de LMDs locais diferentes.

Esta situação pode ser melhorada instituindo-se um modelo de dados e uma LMD pivots, intermediários e invisíveis, em princípio, aos usuários. Um comando formulado contra o banco distribuído seria então traduzido para a LMD pivot e fragmentado em subcomandos ainda na LMD pivot. Assim, seria necessário a construção de apenas  $n$  tradutores.

Finalmente, note que quando atualizações estão envolvidas, o problema recai na tradução de atualizações em visões, que é extremamente difícil.

#### 4.6. Gerência de Transações

As funções de gerência de transações são:

- implementar o conceito de transação;
- gerenciar as atividades e recursos do sistema;
- monitorizar o início e término das atividades do sistema.

Destas atividades, a mais importante é a implementação do conceito de atomicidade. Esta propriedade é caracterizada por uma seqüência de comandos bem delimitada de tal sorte que o sistema deve garantir que, ou todos os comandos na seqüência sejam completamente executados, ou o banco não reflete o resultado da execução de nenhum deles, mesmo que o sistema falhe antes da completa execução de todos os comandos da seqüência. O gerente de transações, portanto, transforma o produto final do processador de consultas em uma unidade atômica de trabalho, implementando os comandos de iniciar, migrar, terminar, cancelar, e reiniciar transações.

Em um SGBD distribuído, a implementação destes comandos é consideravelmente mais difícil pois uma transação pode modificar dados armazenados em mais de um nó, o que significa que vários nós têm que cooperar no processo. Por outro lado, por requisitos de projeto, toda informação e mesmo o próprio controle do processo não devem estar centralizados em um único nó.

Com relação aos outros componentes do SGBD, o gerente de transações usa os subsistemas de controle de concorrência e de controle de integridade para executar as suas tarefas.

#### 4.7. Controle de Concorrência

Controle de concorrência visa a garantir que, em toda execução simultânea de um grupo de transações, cada uma seja executada como se fosse a única do sistema. Isto significa que, em uma execução concorrente, transações não devem gerar interferências que levem a anomalias de sincronização. As anomalias mais comumente encontradas são perda de atualizações, perda de consistência do banco e acesso a dados inconsistentes.

Mais precisamente, uma técnica de controle de concorrência deve garantir que toda execução concorrente de um conjunto de transações seja serializável, ou seja, equivalente a alguma execução das transações em que cada transação é completamente processada antes da próxima começar (i.e., a execução é serial).

Há três classes básicas de técnicas de controle de concorrência: técnicas de bloqueio, pré-ordenação ou mistas. As técnicas de bloqueio exigem que um dado seja bloqueado pela transação antes de ser lido ou modificado (embora isto não seja suficiente, conforme veremos). Estas técnicas em geral criam problemas de bloqueio mútuo ("deadlock") que são especialmente difíceis de resolver em um ambiente distribuído.

Nas técnicas de pré-ordenação, a ordem das transações é escolhida "a priori" e as transações são executadas concorrentemente como se fossem processadas serialmente na ordem escolhida. Em geral estas técnicas evitam problemas de bloqueio mútuo, mas criam problemas de reinício cíclico de transações e postergação indefinida de transações ("cyclic restart" e "indefinite postponement").

As técnicas mistas, como o nome indica, tentam combinar as vantagens de bloqueio e pré-ordenação.

#### 4.8. Controle de Integridade

Controle de integridade endereça os seguintes problemas:

- implementar o cancelamento, recuperação e término de transações;
- trazer o banco de dados, em caso de falhas, a um estado consistente que reflita apenas o efeito de todas as transações já concluídas;
- recuperar regiões danificadas do banco de dados.

As funções de controle de integridade estão implementadas tanto como parte do SGBD global, quanto como parte dos SGBDs locais. Por exemplo, a recuperação de regiões do banco danificadas por falhas nos periféricos é uma função do SGBD local. Já as funções relacionadas a transações e a manter a consistência do banco é uma função do SGBD global, pois envolve o banco de dados distribuído como um todo.

Os tipos de falhas podem ser classificados em: falhas (de "hardware" ou "software") no processador local, falhas nos periféricos que armazenam o banco e falhas na rede de comunicação de dados.

Nota: controle de integridade se refere, portanto, à integridade física dos dados neste texto. Não inclui a manutenção automática dos critérios de consistência lógica do banco que porventura tenham sido definidos para o banco (critérios tais como "todo salário deve ser maior do que o mínimo").

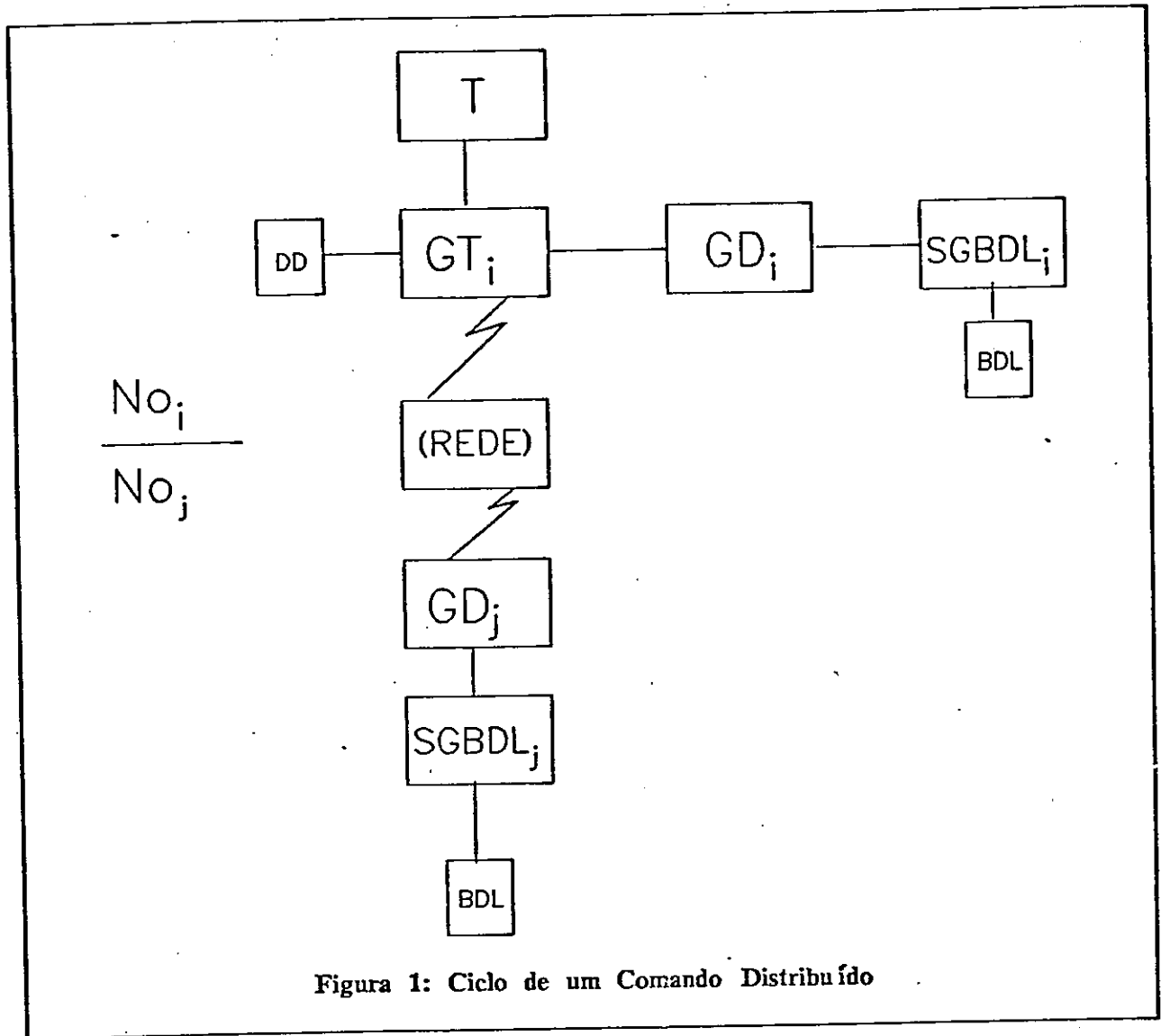
#### 4.9. Controle de Acesso ao Banco

O objetivo da função de controle de acesso é implementar mecanismos que garantam a segurança dos dados armazenados no banco, permitindo que informação seja lida ou modificada apenas por usuários autorizados.

O controle de acesso pode usar um mecanismo de definição de visões, restringindo a que partes do banco de dados cada grupo de usuários pode ter acesso. Um usuário, ou transação, acessaria o banco de dados através da sua visão. O sistema seria, então, encarregado de traduzir acessos à visão em acessos aos objetos realmente armazenados pelo banco.

Independentemente do mecanismo de visões, o sistema deve oferecer um mecanismo de autorização de privilégios. Como parte da LDD, existiriam comandos de autorização indicando para cada usuário que privilégios (leitura, modificação, inserção, etc.) possui e como estes

privilégios se relacionam com as estruturas lógicas do banco. Ao entrar no sistema (ou começar a executar, no caso de transações), haveria um mecanismo de autenticação do usuário (ou transação) estabelecendo a sua identidade perante o sistema. Finalmente, a cada acesso ao banco (ou outra unidade de trabalho mais conveniente, principalmente no caso de transações), o sistema verificaria se o usuário possui os privilégios necessários à execução do acesso.



## REFERÊNCIAS BIBLIOGRÁFICAS

Adiba, M., e B. Lindsay [1980]. "Databases snapshots", IBM Research Report RJ2772, IBM Research Laboratory, San Jose, Calif.

Adiba, M., et all. [1980]. "Polipheme: An experience in distributed database system design and implementation", *Proc. of the Int. Symp. on Distr. Databases*, North Holland, Amsterdam, 67-98.

Alsberg, P.A., e J.D. Day [1976]. "A principle for resilient sharing of distributed resources", *Proc. of the Second Int. Conf. on Software Engineering*, San Francisco, Calif., 562-570.

Astrahan, M.M., et all. [1976]. "System R, a relational approach to database management", *ACM Transactions on Database Systems* 1:2, 97-137.

Astrahan, M.M., W. Kim e M. Schkolnick [1980]. "Evaluation of the System R access path selection mechanism", IBM Research Report RJ2797, IBM Research Laboratory, San Jose, Calif.

Astrahan, M.M., et all. [1981]. "A history and evaluation of System R", *Communications of the ACM* 24:10, 632-646.

Attar, R., P.A. Bernstein e N. Goodman [1981]. "Site initialization and back-up in a distributed database system", Tech. Rep. TR-13-81, Aiken Comp. Lab., Harvard Univ., Cambridge, Mass.

Bernstein, P.A., et al. [1978]. "The concurrency control mechanism of SDD-1: A system for distributed databases (the fully redundant case)", *IEEE Trans. on Software Engineering* 4:3., 154-168.

Bernstein, P.A., et al. [1979]. "Concurrency control in SDD-1: A system for distributed databases, part 1: description", Tech. Rep. CCA-79-03, Computer Corporation of America, Cambridge, Mass.

Bernstein, P.A., D.W. Shipman e W.S. Wong [1979]. "Formal aspects of serializability in database concurrency control", *IEEE Trans. on Software Engineering* 5:3., 203-215.

Bernstein, P.A., e N. Goodman [1980a]. "Fundamental algorithms for concurrency control in distributed database systems", Tech. Rep. CCA-80-05, Computer Corporation of America, Cambridge, Mass.

Bernstein, P.A., e N. Goodman [1980b]. "Timestamped based algorithms for concurrency control in distributed data base systems", *Proc. Int. Conf. on Very Large Data Bases*, Montreal, Canada.

Bernstein, P.A., N. Goodman e M.Y. Lai [1980]. "Two part proof schema for database concurrency control", *Proc. 5th. Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, Calif.

Bernstein, P.A., e D.W. Shipman [1980]. "The correctness of a concurrency control mechanism in a system for distributed databases (SDD-1)", *ACM Transactions on Database Systems* 5:1, 52-68.

Bernstein, P.A., D.W. Shipman e J.B. Rothnie Jr [1980]. "Concurrency control in a system for distributed databases (SDD-1)", *ACM Transactions on Database Systems* 5:1, 18-25.

Bernstein, P.A., e D.W. Chiu [1981]. "Using semi-joins to solve relational queries" *Journal of the ACM* 28:1, 25-40.

Bernstein, P.A., e N. Goodman [1981]. "Concurrency control in distributed databases systems", *Computing Surveys* 13:2, 186-221.

Bernstein, P.A., N. Goodman e M.Y. Lai [1983]. "Analysing concurrency control algorithms when user and system operations differ" (to appear in *IEEE Trans. on Software Engineering*).

Bjork, L. [1973]. "Recovery scenario for a DB/DC system", *Proc. ACM National Conf.*, 142-146.

Blasgen, M.W., e K.P. Eswaran [1976]. "On the evaluation of queries in a relational data base system", IBM Research Report RJ1745, IBM Research Laboratory, San Jose, Calif.

Blasgen, M.W., et all. [1979]. "System R: An architectural update", IBM Research Report RJ2581, IBM Research Laboratory, San Jose, Calif.

Borr, A. [1981]. "Transaction monitoring in ENCOMPAS: reliable distributed transaction processing", *Proc. Int. Conf. on Very Large Data Bases*, Cannes, França.

Bracchi, G., et all. [1980]. "Distributed query processing" em *Distributed Data Bases: An Advanced Course*, I. W. Draffan e F. Poole (eds.), Cambridge University Press, Cambridge, Inglaterra.

Chamberlin, D.D., et all. [1976]. "SEQUEL 2: a unified approach to data definition, manipulation and control", *IBM Journal of Research and Development* 20:6, 560-575.

Chamberlin, D.D., et all. [1981]. "Support for repetitive transactions and ad hoc queries in System R", *ACM Transactions on Database Systems* 6:1, 70-94.

Champine, G.A. [1978]. "Six approaches to distributed data bases" em *Distributed Data Base Management*, P.A. Bernstein, J.B. Rothnie e D.W. Shipman (eds.), IEEE Computer Society, Long Beach, Calif., 45-48.

Dayal, U. [1983a], "Processing queries with quantifiers: a horticultural approach", *Proc. ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, Atlanta, Georgia, 125-136.

Dayal, U. [1983b]. "Processing queries over generalization hierarchies in a multibase system", *Proc. Int. Conf. on Very Large Data Bases*, Florence, Italia, 342-353.

Draffan, L.W., e F. Poole [1980]. "The classification of distributed data base management systems" em *Distributed Data Bases: An Advanced Course*, I. W. Draffan e F. Poole (eds.), Cambridge University Press, Cambridge, Inglaterra.

Ellis, C. [1977]. "A robust algorithm for updating duplicated databases", *Proc. 2th. Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, Calif.

Epstein, R., M. Stonebraker e E. Wong [1978]. "Distributed query processing in a relational database system", *Proc. SIGMOD Int. Conf. on the Manag. of Data*, Austin, Texas.

Eswaran, K.P., et all. [1976]. "The notions of consistency and predicate locks in a relational database system", *Communications of the ACM* 19:11, 624-634.

Filkelstein, S. [1982]. "Common expression analysis in database applications", *Proc. SIGMOD Int. Conf. on the Manag. of Data*, Orlando, Florida, 235-245.

Fussel, D., et all. [1981]. "Deadlock removal using partial rollback in database systems", *Proc. SIGMOD Int. Conf. on the Manag. of Data*, Ann Arbor, Michigan, 65-73.

Furtado, A.L., e M.A. Casanova "Updating relational views", em *Query Processing in Database Systems*, W. Kim, D. Reiter, e D. Batory (eds.), Springer-Verlag (a ser publicado).

Garcia-Molina, H. [1978]. "Performance comparison of two algorithms for distributed databases", *Proc. 3th. Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, Calif.

Gifford, D.K. [1979]. "Weighted voting for replicated data", *Proc. ACM SIGOPS Symposium on Operating Systems Principles*.

Gligor, V.D., e S.H. Shattuck [1980]. "On deadlock detection in distributed systems", *IEEE Trans. on Software Engineering* 6:5, 435-440.

Goodman, N., et all [1979]. "Query processing in SDD-1: A system for distributed databases", Tech. Rep. CCA-79-06, Computer Corporation of America, Cambridge, Mass.

Gorski, J. [1979]. "A formal model for transaction backup in a database environment", Institute for Informatics, Tech. Univ. of Gdansk, Polonia (Draft).

Gray, J.N., R.A. Lorie e G.R. Putzolu [1975]. "Granularity of locks in a shared database", *Proc. Int. Conf. on Very Large Data Bases*, Framingham, Mass., 428-451.

Gray, J.N. [1976]. "Granularity of locks and degrees of consistency in a shared database" em *Modeling in Data Base Management Systems*, G.M. Nijssen (ed.), North Holland, Amsterdam, 365-394.

Gray, J.N. [1978]. "Notes on Databases Operating Systems" em *Operating Systems: An Advanced Course*, R. Bayer, R.M. Graham e G. Seegmuller (eds.), Lecture Notes in Computer Science Vol. 60, Springer Verlag, New York, 391-481.

Gray, J.N. [1979]. "A discussion on distributed systems", *Proc. Congresso AICA*, Bari, Italia.

Gray, J.N., et all [1979]. "The recovery manager of the System R database manager", *Computing Surveys* 13:2, 233-242.

Gray, J.N. [1980]. "A Transaction model", IBM Research Report RJ2895, IBM Research Laboratory, San Jose, Calif.

Gross, J.M., et al [1980]. "Distributed data base design and administration", em *Distributed Data Bases: An Advanced Course*, I. W. Draffan e F. Poole (eds.), Cambridge University Press, Cambridge, Inglaterra.

Hadzilacos, V. [1982]. "An algorithm for minimizing rollback costs", *Proc. ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, Los Angeles, Calif., 93-97.

Hammer, M., e D. Shipman [1979]. "Reliability mechanism for SDD-1: A system for distributed databases", *ACM Transactions on Database Systems* 5:4, 431-466.

Held, G.D., M. Stonebraker e E. Wong [1975]. "INGRES: A relational database system", *AFIPS 1975 National Computer Conference*, Vol. 44, AFIPS Press, Montvale, N.J., 342-353.

Hevner, A.R., e S.B. Yao [1978]. "Query processing on a distributed database", em *Tutorial: Distributed data base management*, P.A. Bernstein, J.B. Rothnie e D.W. Shipman (eds.), IEEE Computer Society, Long Beach, Calif., 69-86.

Horowitz, E., e S. Sahni [1976]. *Fundamental of Data Structures*, Computer Science Press, Woodland Hills, Calif.

Jarke, M., e J. Koch [1983]. "A fast method to evaluate quantified queries", *Proc. ACM-SIGMOD Annual Meeting*, San Jose, Calif., 40-54.

Kim, W. [1981]. "Query optimization for relational database systems", IBM Research Report RJ3081, IBM Research Laboratory, San Jose, Calif.

Klug, A. [1983]. "Locking expressions for increased database concurrency", *Journal of the ACM* 30:1, 36-54.

Knuth, D.E. [1968] *The Art of Computer Programming*, Vol. 1: *Fundamental Algorithms*, Addison-Wesley, Reading, Mass.

Kohler, W.H. [1981]. "A survey of techniques for synchronization and recovery in decentralized computer systems", *Computer Surveys* 13:2, 149-184.

Korth, H.F. [1982]. "Deadlock freedom using edge locks", *ACM Transactions on Database Systems* 7:4, 632-652.

Korth, H.F. [1983]. "Locking primitives in a database system", *Journal of the ACM* 30:1, 55-79.

Lamport, L. [1978]. "Time, clocks and ordering of events in a distributed system", *Communications of the ACM* 21:7, 558-565.

Lampson, B.W. [1980]. "Replicated Commit", *Proc. of the Workshop on Fundamental Issues in Distr. Computing*, Pala Mesa, Calif.

Lampson, B.W., e H.E. Sturgis [1976]. "Crash recovery in a distributed data storage system", Tech. Rep. Computer Science Lab., Xerox Palo Alto Research Center, Palo Alto, Calif., (to appear in *Communications of the ACM*).

LeLann, G. [1978]. "Algorithms for distributed data sharing systems which use tickets", *Proc. 3th. Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, Calif., 259-272.

Leung, J.Y.T., e E.K. Lai [1979]. "On minimum cost recovery from system deadlock", *IEEE Transactions on Computers* 28:9, 671-677.

Lindsay, B.G., et all. [1979]. "Notes on distributed database systems", IBM Research Report RJ2571, IBM Research Laboratory, San Jose, Calif.

Lindsay, B.G. [1980]. "Single and multi-site recovery facilities" em *Distributed Data Bases: An Advanced Course*, I. W. Draffan e F. Poole (eds.), Cambridge University Press, Cambridge, Inglaterra.

Lindsay, B.G., e P.G. Selinger [1980]. "Site autonomy issues in R\*: a distributed database management system", IBM Research Report RJ2927, IBM Research Laboratory, San Jose, Calif.

Lindsay, B.G. [1981]. "Object naming and catalog management for a distributed database manager", *Proc. 2nd. Int. Conf. on Distr. Comp. Systems*, Paris, França.

Lindsay, B.G. [1983]. "Computation and communication in R\*: a distributed database manager", IBM Research Report RJ3740, IBM Research Laboratory, San Jose, Calif.

Liskov, B., e R. Scheifler [1982]. "Guardians and actions: linguistic support for robust distributed programs", *Proc. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages and Systems*, Albuquerque, Novo Mexico, 7-19.

Lorie, R.A. [1977]. "Physical integrity in a large segmented database", *ACM Transactions on Database Systems* 2:1, 91-104.

Lynch, N.A. [1983]. "Multilevel atomicity - a new correctness criterion for database concurrency control", *ACM Transactions on Database Systems* 8:4, 484-502.

Menascé, D.A., e R.R. Muntz [1979]. "Locking and deadlock detection in distributed databases", *IEEE Trans. on Software Engineering* 5:3, 195-201.

Menascé, D.A., e T. Nakanishi [1982]. "Optimistic versus pessimistic concurrency control mechanisms in database management systems", *Information Systems* 7:1, 13-27.

Menascé, D.A., e D. Schwabe [1984]. *Redes de Computadores: Aspectos Técnicos e Operacionais*, Ed. Campus, Rio de Janeiro.

Milenkovic, M. [1979]. "Update synchronization in multiaccess database systems", Ph.D. Dissertation, Dept. of Electrical and Computing Eng., Univ. of Massachusetts, Amherst, Mass.

Mohan, C. [1980]. "Distributed data base management: some thoughts and analyses", *Proc. ACM Annual Conf.*, Nashville, Tenn., 399-410.

Mohan, C., R. Strong e S. Finkelstein [1983]. "Method for distributed transaction commit and recovery using bizantine agreement within clusters of processors", *Proc. ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, Montreal, Canada.

Montgomery, W.A. [1978]. "Robust concurrency control for a distributed information system", MIT/LCS/TR-205, Lab. for Computer Science, M.I.T., Cambridge, Mass.

Moss, M. [1981]. "Nested transactions: an approach to reliable distributed computing", MIT/LCS/TR-260, Lab. for Computer Science, M.I.T., Cambridge, Mass.

Murthy, K., J. Kam e M.S. Krishnammorthy [1983]. "An approximation algorithm to the file allocation problem in computer networks", *Proc. ACM-SIGMOD Annual Meeting*, San Jose, Calif., 40-54.

Nakanishi, T. [1981]. "Análise de desempenho de mecanismos de controle de concorrência em bancos de dados", Tese de Doutorado, Departamento de Informática, PUC/RJ, Rio de Janeiro.

Neuhold, E.J., e B. Walter [1982]. "An overview of the architecture of the distributed data base system "POREL"" em *Distributed data bases*, H.-J. Schneider (ed.), North-Holland, Amsterdam.

Ng, P. [1982]. "Distributed compilation and recompilation of database Queries", IBM Research Report RJ3375, IBM Research Laboratory, San Jose, Calif.

Obermarck, R. [1980]. "Global deadlock detection algorithm", IBM Research Report RJ2845, IBM Research Laboratory, San Jose, Calif.

Obermarck, R. [1982]. "Distributed deadlock detection algorithm", *ACM Transactions on Database Systems* 7:2, 187-208.

Papadimitriou, C.H. [1979]. "Serializability of concurrent databases updates", *Journal of the ACM* 24:4, 631-653.

Papadimitriou, C.H., P.A. Bernstein e J.B. Rothnie [1977]. "Some computational problems related to database concurrency control", *Proc. Conf. Theoretical Computer Science*, Waterloo, Canada.

Pelagatti, G., e F.A. Schreiber [1980]. "Distributed data base applications: model of an access strategy in a distributed data base" em *Distributed Data Bases: An Advanced Course*, I. W. Draffan e F. Poole (eds.), Cambridge University Press, Cambridge, Inglaterra.

Randell, B., et all. [1978]. "Reliability issues in computing system design", *Computing Surveys* 10:2, 123-165.

Randell, B. [1979]. "Reliable computing systems" em *Operating Systems: An Advanced Course*, R. Bayer, R.M. Graham e G. Seegmuller (eds.), Lecture Notes in Computer Science Vol. 60, Springer Verlag, New York, 391-481.

Rappaport, R.L. [1975]. "File structure design to facilitate on-line instantaneous updates", *Proc. SIGMOD Int. Conf. on the Manag. of Data*, 1-14.

Reed, D.P. [1978]. "Naming and synchronization in a decentralized computer system", Ph.D. Dissertation, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass.

Reed, D.P. [1979]. "Implementing atomic actions on decentralized data", *Proc. ACM SIGOPS Symposium on Operating Systems Principles*, (to appear in *Communications of the ACM*).

Richard, P. [1981]. "Evaluation of the size of a query expressed in relational algebra", *Proc. SIGMOD Int. Conf. on the Manag. of Data*, Ann Arbor, Michigan, 155-163.

Ries, D.R., e M. Stonebraker [1977]. "Effects of locking granularity in a database management system", *ACM Transactions on Database Systems* 2:3, 233-246.

Ries, D.R., e M. Stonebraker [1979]. "Locking granularity revised", *ACM Transactions on Database Systems* 4:2, 210-227.

Ries, D.R. [1979a]. "The effects of concurrency control on database management system performance", Ph.D. Dissertation, Univ. of California, Berkeley, Calif.

Ries, D.R. [1979b]. "The effects of concurrency control on the performance of a distributed database management system", *Proc. 4th. Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, Calif.

Rosenkrantz, D.J., R.E. Stearns e P.M. Lewis II [1978]. "System level concurrency control for distributed databases", *ACM Transactions on Database Systems* 3:2, 178-198.

Rosenthal, A., e D. Reiner [1982]. "An architecture for query optimization", *Proc. SIGMOD Int. Conf. on the Manag. of Data*, Orlando, Florida, 246-255.

Rothnie, J.B., e N. Goodman [1977a]. "An overview of the preliminary design of SDD-1: a system for distributed databases", *Proc. 2th. Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, Calif., 39-57.

Rothnie, J.B., e N. Goodman [1977b]. "A survey of research and development in distributed database management", *Proc. Int. Conf. on Very Large Data Bases*, Toquio, Japão

Rothnie, J.B., N. Goodman e T. Marill [1978]. "Database architecture in a network environment" em *Protocols and techniques for data communication networks*, F.F. Kuo (ed.), Prentice-Hall, Englewood Cliffs, N.J.

Rothnie, J.B., et all. [1980]. "Introduction to system for distributed databases (SDD-1)", *ACM Transactions on Database Systems* 5:1, 1-17.

Rypka, D.J., e A.P. Lucido [1979]. "Deadlock detection and avoidance for shared logical resources", *IEEE Trans. on Software Engineering* 5:5, 465-471.

Selinger, P.G., et. all. [1979]. "Access path selection in a relational database management system", *Proc. SIGMOD Int. Conf. on the Manag. of Data*, Boston, Mass., 23-34.

Selinger, P.G., e M. Adiba [1980]. "Access path selection in distributed database management systems", *Proc. Int. Conf. on Databases*, Aberdeen, Escócia. (também IBM Research Report RJ2883, IBM Research Laboratory, San Jose, Calif.).

Severance, D.G., e G.M. Lohman [1976]. "Differential files: their application to the maintenance of large databases", *ACM Transactions on Database Systems* 1:3, 256-267.

Schreiber, F.A., C. Baldissera e S. Ceri [1980]. "Distributed data base applications: an overview" em *Distributed Data Bases: An Advanced Course*, I. W. Draffan e F. Poole (eds.), Cambridge University Press, Cambridge, Inglaterra.

Shapiro, R.M., e R.E. Millstein [1977a]. "Reliability and fault recovery in distributed processing", *Oceans '77 Conf. Record*, Vol. II, Los Angeles, Calif.

Shapiro, R.M., e R.E. Millstein [1977b]. "NSW reliability plan", TR-7701-1411, Massachusetts Comp. Assoc., Wakefield, Mass.

Shapiro, R.M., e R.E. Millstein [1978]. "Failure recovery in a distributed data base system", *Proc. COMPCOM Spring '78*, 66-70.

Shertock, M.J. [1982]. "Distribution in an installed integrated on-line real-time data-base environment", *Proc. Int. Conf. on Management of Distributed Data Processing*, Cergy, França, 277-291.

Skeen, D. [1981]. "Nonblocking commit protocols", *Proc. SIGMOD Int. Conf. on the Manag. of Data*, Ann Arbor, Michigan, 133-142.

Skeen, D. [1982]. "A quorum based commit protocol", *Proc. 6th. Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, Calif., 69-90.

Skeen, D., e M. Stonebraker [1983]. "A formal model of crash recovery in a distributed system", *IEEE Trans. on Software Engineering* 9:3, 219-228.

Smith, D.D.P., e J.M. Smith [1979]. "Relational database machines" *Computer* 12:3, 28-38.

Smith, J.M., et all. [1981]. "MULTIBASE - Integrating heterogeneous distributed database systems", *Proc. AFIPS 1981 National Computer Conference*, Vol. 50, AFIPS Press, Montvale, N.J., 487-499.

Spaccapietra, S. [1980]. "Heterogeneous distributed data base systems" em *Distributed Data Bases: An Advanced Course*, I. W. Draffan e F. Poole (eds.), Cambridge University Press, Cambridge, Inglaterra.

Stearns, R.E., P.M. Lewis II e D.J. Rosenkrantz [1976]. "Concurrency control for database systems", *Proc. 17th Annual Symp. on Foundations Computer Science*, Houston, Texas, 19-32.

Stonebraker, M., e E. Neuhold [1977]. "A distributed data base version of INGRES", *Proc. 2th. Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, Calif.

Stonebraker, M., et all [1976]. "The design and implementation of INGRES", *ACM Transactions on Database Systems* 1:3, 189-222.

Stonebraker, M. [1980]. "Homogeneous distributed data base systems", em *Distributed Data Bases: An Advanced Course*, I. W. Draffan e F. Poole (eds.), Cambridge University Press, Cambridge, Inglaterra.

Stonebraker, M. [1979]. "Concurrency control and consistency of multiple copies of data in distributed INGRES", *IEEE Trans. on Software Engineering* 5:3, 180-194.

Sturgis, H., J. Michell e J. Israel [1980]. "Issues in the design and use of a distributed file system", *ACM Oper. Syst. Review* 14:3, 55-69.

Tannenbaum, A.S. [1981]. *Computer Networks*, Prentice-Hall, Englewood Cliffs, N.J.

Thomas, R.H. [1978]. "A solution to the concurrency control problem for multiple copy databases", *Proc. 1978 IEEE COMPCON*.

Thomas, R.H. [1979]. "A majority consensus approach to concurrency control for multiple copies databases", *ACM Transactions on Database Systems* 4:2, 180-209.

Traiger, I., et al. [1978]. "Transactions and consistency in a distributed data base system", IBM Research Report RJ2555, IBM Research Laboratory, San Jose, Calif.

Verhofstad, J.S.M. [1978]. "Recovery techniques for database systems", *Computer Surveys* 10:2, 167-195.

Williams, R., et al. [1981]. "R\*: An overview of the architecture", IBM Research Report RJ3325, IBM Research Laboratory, San Jose, Calif.

Wilms, P., e B. Lindsay [1981]. "A database authorization mechanism supporting individual and group authorization", *Proc. of the 2nd. Seminar on Distributed Data Sharing Systems*, (a ser publicado pela North Holland, Amsterdam).

Wah, B.W. [1984]. "File placement on distributed computer systems", *Computer* 17:1, 23-32.

Wong, E., e K. Youssefi [1976]. "Decomposition - a strategy for query processing", *ACM Transactions on Database Systems* 1:3, 223-241.

Wong, E. [1977]. "Retrieving dispersed data from SDD-1: a system for distributed databases", Tech. Rep. CCA-77-03, Computer Corporation of America, Cambridge, Mass.

Yu, C.T., e C.C. Chang [1983]. "On the design of a query processing strategy in a distributed database environment", *Proc. ACM-SIGMOD Annual Meeting*, San Jose, Calif., 30-39.