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

Heloise Manica

BANCOS DE DADOS DISTRIBUÍDOS HETEROGÊNEOS: ARQUITETURAS, TECNOLOGIAS E TENDÊNCIAS

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

Orientador:

Murilo Silva de Camargo

Florianópolis, fevereiro/2001

BANCOS DE DADOS DISTRIBUÍDOS HETEROGÊNEOS: ARQUITETURAS, TECNOLOGIAS E TENDÊNCIAS

Heloise Manica

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Dr. Eng. Murilo Silva de Camargo
(Orientador e Presidente da Banca)

Prof. Dr. Eng. Fernando A. Ostuni Gauthier
(Coordenador do Curso)

Banca Examinadora

Prof. Dr. Vitório Bruno Mazzola

Prof. Dr. Roberto Willrich

Prof. Dr. Rosvelter J. Coelho da Costa

"O Universo esta dentro de nós.
Portanto, devemos pedir tudo a nós mesmos..."

AGRADECIMENTOS

Agradeço a Deus por estar sempre ao meu lado.
Ao meu orientador Prof. Murilo Silva de Camargo
pela confiança em mim depositada.
Aos meus pais por sempre me apoiarem nas horas difíceis.
Aos meus irmãos pelo incentivo e finalmente,
a todos aqueles que caminharam comigo e
contribuíram para a realização deste trabalho.

RESUMO

Com o desenvolvimento das empresas e instituições, surge a necessidade de integração dos sistemas de bancos de dados preexistentes, autônomos e diferentes. A utilização de Sistemas de Bancos de Dados Heterogêneos, por sua capacidade de integrar arquiteturas diferentes, apresenta-se como uma alternativa interessante para a solução do problema de integração de dados. A tecnologia de Sistemas de Bancos de Dados Heterogêneos permite que sistemas de modelos, fabricantes e produtos diferentes possam ser integrados de forma a oferecer uma visão única ao usuário. No entanto, devido à autonomia e heterogeneidade dos sistemas existentes, este ambiente apresenta muitos desafios. Acessar e gerenciar dados originários de vários bancos de dados independentes gera vários problemas. Este trabalho apresenta um estudo e análise sobre os principais problemas e soluções propostas para a integração de bases de dados heterogêneas. Foram analisadas as principais abordagens utilizadas por diferentes autores para tratar problemas relacionados ao processamento/otimização de consultas, gerenciamento/concorrência de transações e interoperabilidade em ambientes autônomos e heterogêneos. Por fim, apresentamos comentários sobre alguns projetos de sistemas gerenciadores de bancos de dados heterogêneos que vem sendo desenvolvidos nos últimos anos.

Palavras-chave: Sistemas de bancos de dados heterogêneos, interoperabilidade, sistemas *multidatabase*, sistemas autônomos, integração de bases de dados.

ABSTRACT

The progress of the enterprises and institutions raises the need of integration of the preexistent, autonomous and different database systems. The use of Heterogeneous Databases Systems comes forward as an interesting alternative for the solution of data integration problem, because it can integrate different architectures. The technology of Heterogeneous Databases Systems allows that systems with different models, makers and products can be integrated in such a way that the user can get only one vision. However, due to the autonomy and heterogeneity of the existent systems, this environment causes many challenges, as well as to access, to manager data from several independent databases also generates several problems. This work presents a study on the main problems and solutions proposed for heterogeneous database integration. The main approaches used by different authors were analyzed in order to handle the problems concerning query processing/optimization, transaction management/concurrency, and interoperability in autonomous and heterogeneous environment. Finally, we present remarks about some projects of heterogeneous databases systems managers that have been developed recently.

Keywords: Heterogeneous database managers systems, interoperability, multidatabase systems, autonomous systems, database integration.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

INTRODUÇÃO

1.1 Considerações Iniciais

Durante os anos setenta, bancos de dados centralizados eram dominantes. Até então, a maioria das grandes organizações mantinha seus dados armazenados em bancos de dados distintos e independentes, que surgiram para atender necessidades características de seu ambiente.

Perante as mudanças da situação social e econômica das empresas e instituições, os sistemas de computação vêm sofrendo freqüentemente várias alterações. Quanto maior o sistema, maior é a dificuldade e os custos para sua substituição ou mesmo atualização.

A rápida evolução das redes de comunicação de dados, bem como a gradual redução de seus custos, permite a utilização cada vez mais freqüente de sistemas de banco de dados distribuídos para gerenciamento dos dados de empreendimentos cuja organização é geograficamente dispersa.

Uma classe especial desse tipo de banco de dados surge quando empresas se fundem e necessitam integrar bancos de dados pré-existentes para formar uma nova visão global dos dados do empreendimento. Sistemas gerenciadores de bancos de dados heterogêneos têm sido desenvolvidos como uma das soluções mais viáveis e baratas para esta classe de sistemas.

Para interligar essas bases de dados isoladas e heterogêneas faz-se necessária uma arquitetura que possibilite ao sistema gerenciador acessar os dados originários de bancos de dados pré-existentes e heterogêneos que se encontram espalhados pela rede.

Uma maneira de conseguir interoperabilidade entre bancos de dados heterogêneos é através de um sistema gerenciador de bancos de dados heterogêneos (SGBDH), que permita aos usuários acessar simultaneamente bancos de dados independentes usando uma linguagem de definição e de manipulação única.

É importante que a integração entre os sistemas preserve as características de cada sistema individual. Ou seja, a integração não deve violar as características de cada sistema autônomo. O problema de integração de bancos de dados é complexo, visto que os vários componentes podem estar descritos em modelos de dados diferentes (relacional, redes, etc) ou mesmo sendo gerenciados localmente por sistemas de gerência de bancos de dados diferentes.

As principais dificuldades encontradas no modelo de banco de dados heterogêneo estão [Özsu e Valduriez, 1999]: na definição de um modelo de integração, no processamento e otimização de consultas, na gerência e controle da concorrência de transações e na interoperabilidade entre os sistemas.

A maioria destes assuntos ainda não possui uma solução ideal. Atualmente existe um grande número de pesquisas sobre os sistemas de bancos de dados heterogêneos. O esforço para distinção e conceituação de soluções encontra-se expresso em diversos trabalhos.

Neste trabalho é apresentado um estudo dos principais problemas relacionados a sistemas de bancos de dados distribuídos heterogêneos, assim como as soluções que estão sendo propostas nos últimos anos. Os sistemas de bancos de dados heterogêneos considerados neste trabalho variam de acordo com as três dimensões [Özsu e Valduriez, 1999]: distribuição, autonomia e heterogeneidade.

A tecnologia de Banco de Dados evolui rapidamente, sendo que novas filosofias são adotadas e novos sistemas são desenvolvidos com uso de técnicas cada vez mais apuradas. As soluções pesquisadas em bancos de dados heterogêneos têm um objetivo em comum: fornecer transparência aos usuários locais e globais dos

vários bancos de dados participantes na federação. Esta transparência refere-se à localização física, às linguagens de manipulação, aos caminhos de acesso aos dados e nas diferentes maneiras de sua representação.

1.2 Metodologia

Para a realização deste trabalho, foram pesquisadas diversas bibliografias, tais como: livros, dissertações, teses, artigos, relatórios técnicos, documentos oficiais de congressos, Workshops e sites da Internet. O material utilizado foi obtido principalmente por meio de pesquisa em bibliotecas digitais e Comut.

1.3 Organização do Trabalho

O restante deste trabalho está organizado como indicado a seguir. O capítulo 2 apresenta uma revisão sobre os principais aspectos relacionados a bancos de dados distribuídos, tais como definição, arquitetura, terminologia, problemática, etc., a fim de facilitar a compreensão dos capítulos seguintes.

No capítulo 3 são apresentadas as diversas possibilidades para a distribuição dos dados. É descrita também, a definição de sistemas distribuídos heterogêneos, mostrando suas arquiteturas e terminologia.

O capítulo 4 apresenta as principais formas de integração de bancos de dados individuais já existentes e autônomos, através de modelos que especificam ou não um esquema conceitual global. Detalha também, o processo de integração e tradução de esquemas, destacando vantagens e desvantagens dos diferentes modelos apresentados.

Em seguida, os capítulos 5 e 6 descrevem, respectivamente, as abordagens referentes ao processamento e otimização de consultas em sistemas heterogêneos, salientando seus principais problemas e desafios.

No capítulo 7 são descritas características, modelos, problemática e propostas para o gerenciamento de transações.

O capítulo 8 descreve o controle da concorrência. Entre outros itens abordados, descrevem-se problemas e condições para um controle eficaz da concorrência entre transações locais e globais.

No capítulo 9 trata-se a questão da interoperabilidade entre os diferentes SGBDs, considerando a utilização do modelo orientado a objetos para lidar com interoperabilidade entre bancos de dados heterogêneos.

O capítulo 10 apresenta alguns dos projetos de sistemas gerenciadores de bancos de dados heterogêneos desenvolvidos nos últimos anos.

No capítulo 11 são apresentadas as conclusões deste estudo, destacando-se o interesse na linha de pesquisa relacionada à gerência de transações em sistemas gerenciadores de bancos de dados heterogêneos para um trabalho futuro.

Por fim, um glossário e as referências bibliográficas utilizadas para a realização deste trabalho.

Capítulo 2

BANCO DE DADOS DISTRIBUÍDO

2.1 O que é um Sistema de Banco de Dados Distribuído?

Define-se [Bell e Grimson, 1992] sistema de banco de dados distribuído (SBDD) como uma coleção de dados inter-relacionados que se encontram fisicamente distribuídos pelos nós de uma rede de computadores.

Um sistema gerenciador de banco de dados distribuído (SGBDD) é um software que gerencia um banco de dados distribuído de tal modo que os aspectos da distribuição ficam transparentes para o usuário. Assim, o usuário tem a ilusão de integração lógica de dados, sem requerer a integração física de banco de dados. O usuário de um sistema de banco de dados distribuído é incapaz de saber a origem das informações, tendo a impressão de estar acessando um único banco de dados. A figura 2.1.a ilustra a distribuição de dados em uma rede de computadores.

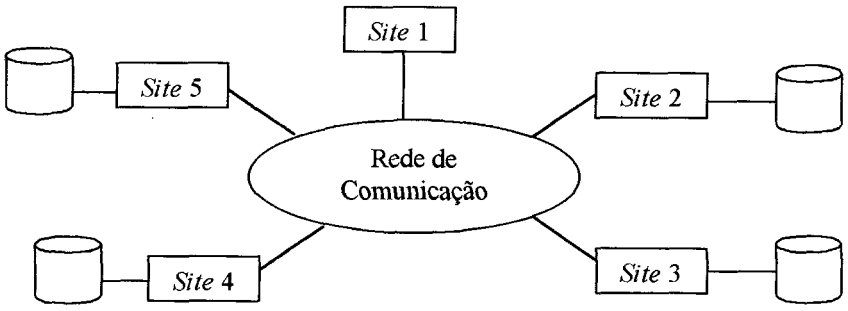

Figura 2.1.a: Banco de dados Distribuídos

O banco de dados centralizado é gerenciado por um único sistema de computador (*site* 2 na figura 2.1.b) e todas as requisições são encaminhadas para este *site*. A existência de uma rede de computadores não significa que o sistema é distribuído, pois não existe a distribuição física dos dados.

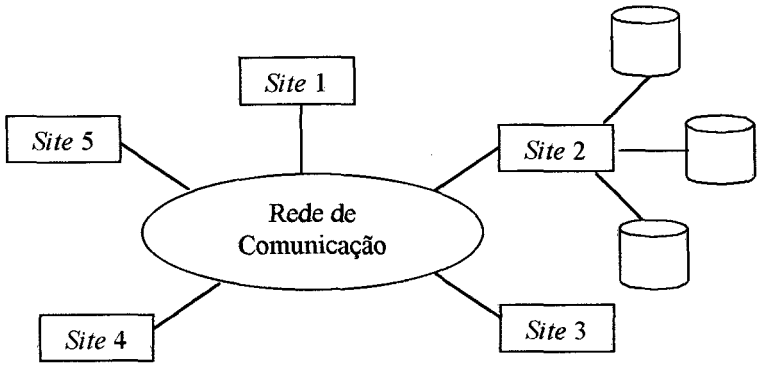

Figura 2.1.b: Banco de dados Centralizado em uma rede de computadores

Enquanto um sistema gerenciador de banco de dados centralizado (SGBD) gerencia um único banco de dados, o SGBDD é responsável pelo gerenciamento global e local [Özsu e Valduriez, 1999].

Cada nó de um banco de dados distribuído é capaz de processar transações *locais*, as quais acessam apenas dados daquele único nó, ou pode participar na execução de transações *globais*, que fazem acesso a dados em diversos nós. A execução de transações globais requer comunicações entre os nós, que podem ser feitos através de cabos, fibras óticas, linhas telefônicas, ligações de microondas, canais de satélite, etc..

Se o projeto de um sistema distribuído é executado *top-down*, isto é, sem um sistema já existente, é conveniente desenvolver um sistema homogêneo. Todavia, em alguns casos onde a criação do banco de dados distribuído for feita pela integração de vários bancos de dados já existentes (chamamos *bottom-up*), será necessário um SGBDD heterogêneo, capaz de fornecer interoperabilidade entre os bancos de dados locais.

2.2 Porque usar Sistema de Banco de Dados Distribuído?

Existem diversas razões para construir um banco de dados distribuído, como o partilhamento de dados, confiabilidade, disponibilidade e aceleração de processamento de consultas. Entretanto, juntamente com essas vantagens há diversas desvantagens, como o custo de desenvolvimento de software, maior potencial para existência de erros e crescente sobrecarga de processamento.

A principal vantagem de sistemas de bancos de dados distribuídos é a capacidade de dividir e fazer acesso a dados de uma maneira confiável e eficiente. Pois, se uma série de nós diferentes estão conectados, então um usuário em um nó pode ser capaz de fazer acesso a dados disponíveis em um outro nó. Cada nó é capaz de reter um grau de controle sobre dados armazenados localmente.

Caso ocorra uma falha em um dos nós do sistema distribuído, os nós remanescentes podem ser capazes de continuar operando, aumentando a confiabilidade e disponibilidade. Além disso, quando uma consulta envolve dados em diversos nós, é possível dividi-la em subconsultas que podem ser executadas em paralelo, acelerando seu processamento.

2.3 Características de um Sistema de Banco de Dados Distribuído

A seguir são descritas algumas características do SBDD, que fazem com que estes sistemas proporcionem inúmeras vantagens a seus usuários.

2.3.1 Gerenciamento transparente de dados distribuídos e replicados

Transparência é a separação da semântica de alto nível de um sistema dos detalhes de implementação. Um sistema transparente esconde detalhes de implementação dos usuários (figura 2.2a e figura 2.2b). Podemos citar vários tipos de transparência em banco de dados como por exemplo: transparência de distribuição, de replicação e de fragmentação dos dados.

Em um SBDD, os programas de aplicação não sabem e não precisam saber onde os dados estão localizados. O SGBDD encontra os dados, onde quer que eles estejam, sem o envolvimento do programa de aplicação.

Transparência de replicação refere-se ao fato de que os programas de aplicação não sabem se os dados estão replicados ou não. Se estiverem, o SGBDD assegurará que todas as cópias sejam atualizadas consistentemente, sem o envolvimento do programa de aplicação.

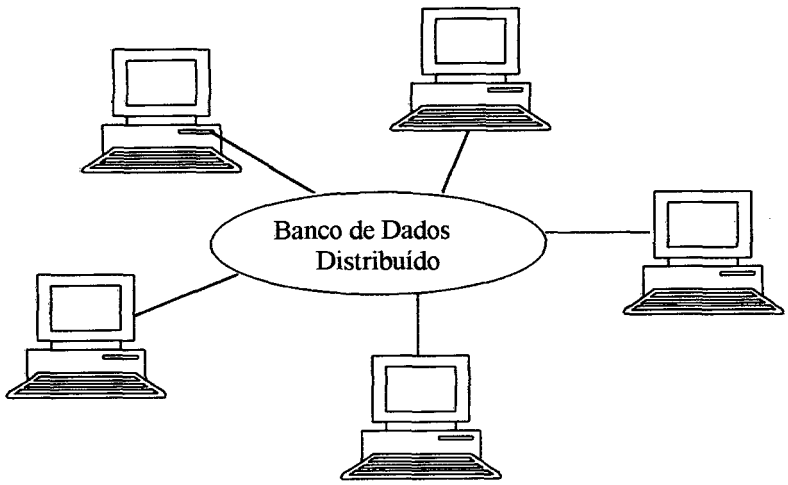

Figura 2.2.a: Banco de dados distribuído – visão do usuário

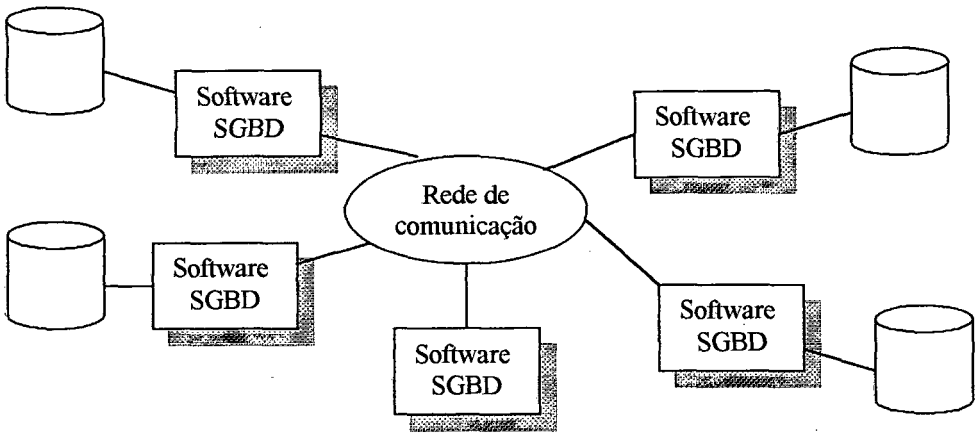

Figura 2.2.b: Banco de dados distribuído – realidade

2.3.2 Independência de dados

Independência dos dados [Özsu e Valduriez, 1999] é uma forma de transparência fundamental que se procura em um SGBD. Refere-se à impossibilidade das aplicações dos usuários em alterar a definição e organização dos dados e vice-versa. Ou seja, os dados são armazenados de forma independente dos programas que os utilizam.

2.3.3 Facilidade e economia para expansão do sistema

Em um ambiente distribuído, é muito mais fácil acomodar novas bases de dados. Muitas vezes custa menos distribuir um sistema de banco de dados em dois pequenos computadores do que utilizar apenas um sistema de banco de dados centralizado em uma única máquina mais potente.

2.3.4 Aumento do desempenho (*performance*)

O desempenho aumenta em SGBDD devido principalmente a dois fatores. Primeiro, o SGBDD fragmenta o esquema conceitual de modo que os dados são armazenados próximos aos pontos nos quais são mais utilizados. Assim, cada *site* possui apenas uma parte do banco de dados, minimizando a sobrecarga de processamento. Um segundo fator é o paralelismo. Quando uma consulta envolve dados em diversos nós, é possível dividi-la em subconsultas que podem ser executadas em paralelo, acelerando seu processamento.

2.3.5 Confiabilidade e disponibilidade através de transações distribuídas

Uma característica importante de sistemas de bancos de dados distribuídos é a capacidade de dividir e prover acesso a dados de uma maneira confiável e eficiente.

Uma falha em um dos nós, ou em um link de comunicação que faça com que um ou mais *sites* fiquem inalcançáveis, não prejudicará o sistema como um todo. Os nós remanescentes podem ser capazes de continuar operando, aumentando a confiabilidade e disponibilidade.

2.3.6 Autonomia

Dependendo do projeto de banco de dados distribuído, os administradores podem possuir diferentes graus de autonomia local. O termo autonomia refere-se à distribuição de controle e indica o grau no qual SGBDs individuais podem operar de forma independente. A possibilidade de autonomia local é uma das maiores vantagens destes sistemas, pois cada sistema gerenciador de banco de dados possui a capacidade de operar independentemente. Este é fator de grande importância em SGBDD, descrito com maiores detalhes no capítulo 3.

2.4 Fatores complicadores em sistemas de bancos de dados distribuídos

A complexidade em sistemas distribuídos aumenta devido a vários fatores. Um deles, refere-se à distribuição dos dados. Não é essencial que cada *site* da rede possua o banco de dados completo e sim que um banco de dados resida em mais de um *site*. Portanto, é necessário definir como será a distribuição e replicação (ou não) dos dados.

Um outro fator refere-se a falhas (de hardware ou software) nos *sites* ou na rede de comunicação que possam vir a ocorrer na execução de uma atualização de dados. O sistema deve ter certeza de que os efeitos serão refletidos nos dados residentes nestes sites indisponíveis assim que o sistema retornar ao normal.

A maior desvantagem do sistema de banco de dados distribuído é a complexidade adicional requerida para assegurar a própria coordenação entre os nós. Devido a esta complexidade é requerido hardware e software adicionais, o que leva a um aumento de custos de desenvolvimento, potencialidade de defeitos e da

sobrecarga de processamento. A tabela 2.1 resume os principais fatores complicadores em SGBD distribuídos.

Projeto	- como distribuir o banco de dados
	- distribuição dos dados replicados
Processamento de consultas	- conversão de transações de usuários em instruções de dados
	- problema de otimização
Controle de concorrência	- sincronização de acessos concorrentes
	- consistência e isolamento de efeitos de transações
	- gerenciamento de *deadlocks*
Confiabilidade	- como manter o sistema imune à falhas
	- atomicidade e durabilidade

Tabela 2.1 - SGBD distribuídos: fatores complicadores

2.5 Arquiteturas de SGBD Distribuídos

Uma arquitetura define a estrutura de um sistema: identificação, definição da função e o inter-relacionamento e interações entre os componentes do sistema. Na arquitetura data-lógica (figura 2.3) é formada pelo esquema interno local de cada *site*, o esquema conceitual local de cada *site*, o esquema conceitual global e esquemas externos.

O esquema interno local (EIL) refere-se aos aspectos relacionados ao armazenamento físico dos dados do *site*. O esquema conceitual local (ECL) refere-se à estrutura lógica (informações) do banco de dados. O esquema conceitual global (ECG) é formado pela união dos esquemas conceituais locais, permitindo uma visão global dos dados. Finalmente o nível mais externo, os esquemas externos (EE) são as visões definidas aos usuários globais.

Esta arquitetura é denominada ponto-a-ponto (*peer-to-peer*) devido ao fato de que cada *site* possui o SGBD completo, diferente da arquitetura cliente servidor que concentra o gerenciamento dos dados em servidores, enquanto nos clientes ficam as interfaces e aplicações.

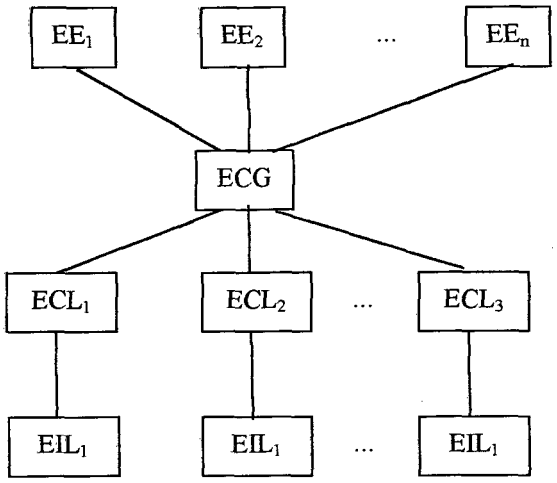

Figura 2.3: Arquitetura data-lógica de SGBD distribuído

Quando o projeto do banco de dados distribuído é realizado a partir de base de dados já existentes o chamamos de *bottom-up*. Deste modo, surge uma nova arquitetura data-lógica de multi-SGBD (figura 2.4). A maior diferença entre esta arquitetura e a data-lógica é a forma do mapeamento entre esquemas. A figura 2.5 ilustra os componentes de um SGBD-D conforme a arquitetura de multi-SGBD.

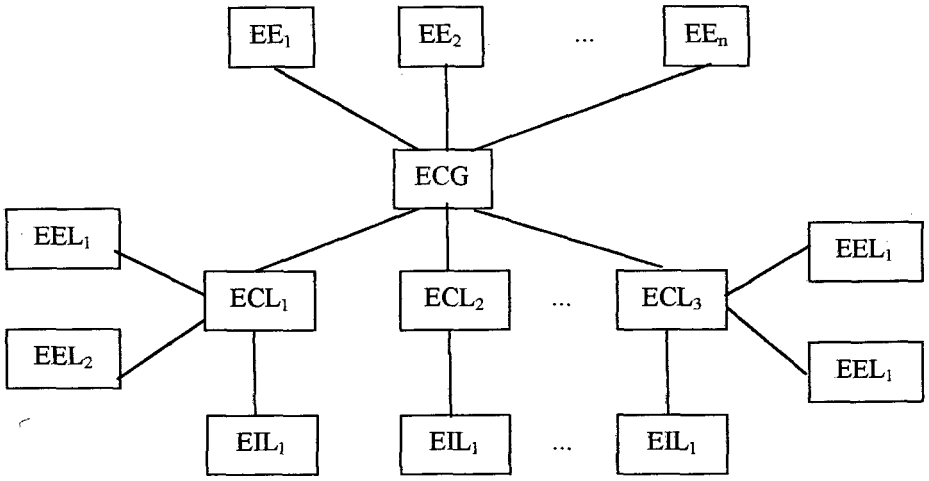

Figura 2.4: Arquitetura data-lógica de multi-SGBD

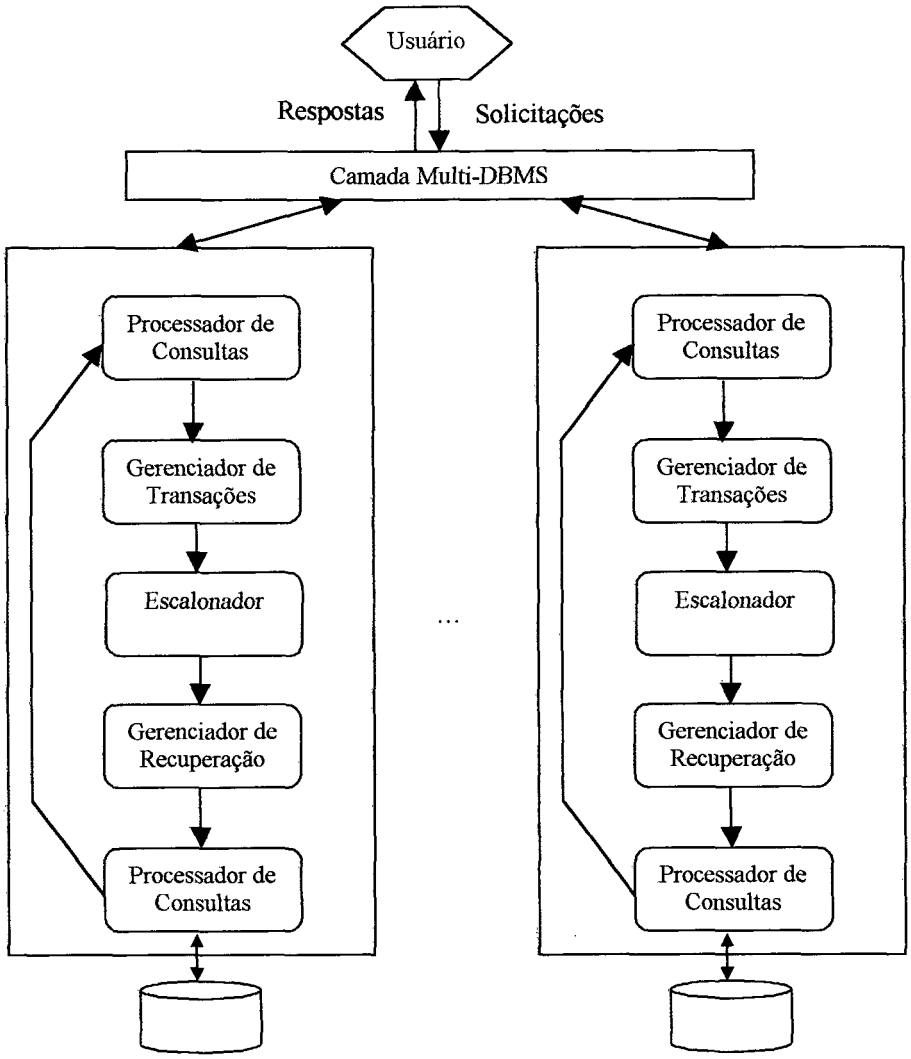

Figura 2.5: Componentes da arquitetura data-lógica de multi-SGBD

2.6 Processamento/otimização de consultas distribuídas

Para sistemas centralizados, o primeiro critério para mensuração do custo de uma estratégia em particular é o número de acessos a disco. Nos sistemas distribuídos, devemos considerar outros problemas, tais como o custo de transmissão de dados na rede.

O processamento de consultas em sistemas de bancos de dados distribuídos envolve várias etapas (figura 2.6). Quando uma consulta (cálculo relacional) é feita sobre as relações distribuídas, primeiramente ela é decomposta. A decomposição de consultas pode ser dividida em quatro passos sucessivos.

Primeiro o cálculo relacional é reescrito em uma forma normalizada adequada. Segundo, é feita uma análise semântica para detecção e rejeição de consultas incorretas. Terceiro, a consulta (ainda representada em cálculo relacional) é simplificada eliminando predicados redundantes. Quarto, o cálculo é reestruturado em uma consulta algébrica (álgebra relacional).

Conforme descrito anteriormente, os dados de um sistema de banco de dados distribuído são fragmentados/replicados e armazenados próximos aos pontos nos quais são mais utilizados. Após a decomposição da consulta (álgebra relacional), inicia-se a segunda etapa: a localização dos dados. Esta camada tem como entrada uma consulta em álgebra relacional, cujo papel é determinar quais fragmentos estão envolvidos na consulta e transformar a consulta distribuída em fragmentos de consultas. A localização dos dados das consultas é feita através de um programa de localização dos dados (*localization program*).

Na terceira etapa é realizada a otimização da consulta global, com o objetivo de encontrar a melhor estratégia para realizar as consultas. As camadas anteriores já ajudaram a otimizar as consultas, por exemplo, eliminando expressões redundantes. Portanto, nesta etapa, otimizar a consulta global consiste em encontrar a melhor ordem para executar as operações das consultas fragmentadas, analisando, inclusive, as operações de comunicação que minimizam os custos. Uma técnica para otimizar a seqüência de operações distribuídas analisa os custos de processamento local.

Até agora, as etapas foram realizadas a nível global, e de responsabilidade do *site* controle. *Site* controle é o *site* que recebeu a consulta global e ficou responsável por ela até que esta seja completada. A última etapa, otimização da consulta local é realizada pelos SGBDs participantes. Cada sub-consulta é então otimizada de acordo com os algoritmos centralizados utilizados pelos *sites* locais.

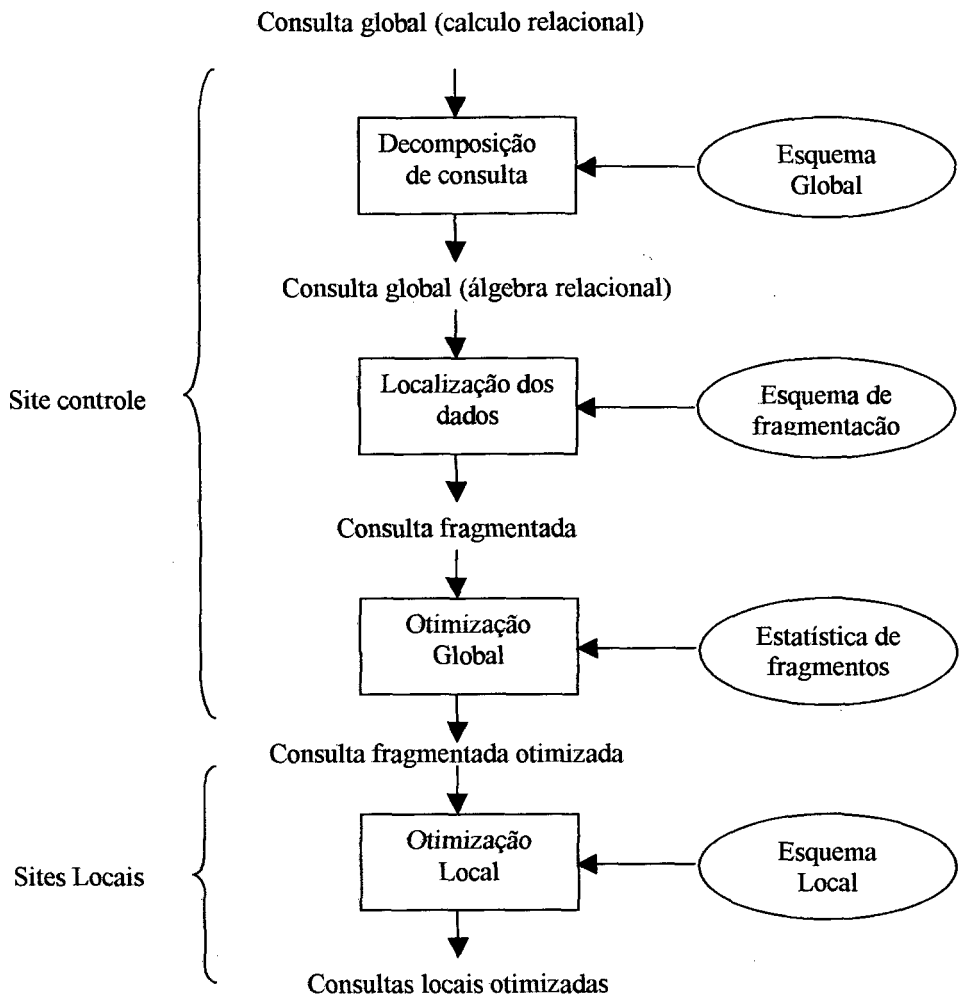

Figura 2.6: Metodologia para processamento de consultas distribuídas

2.7 Gerenciamento e controle da concorrência de transações em sistemas distribuídos

Uma transação é uma coleção de operações que fazem transformações consistentes de estados do sistema preservando a consistência do sistema. Um banco de dados está em um estado consistente se obedece todas as regras de integridade definidas sobre ele.

Mudanças ocorrem devido a modificações, inserções e exclusões que juntas são também chamadas de atualizações. Apesar do objetivo ser garantir que um banco de

dados nunca entre em um estado inconsistente, um banco de dados pode ficar temporariamente inconsistente durante a execução de uma transação. É importante que o banco de dados fique consistente quando a transação terminar (figura 2.7).

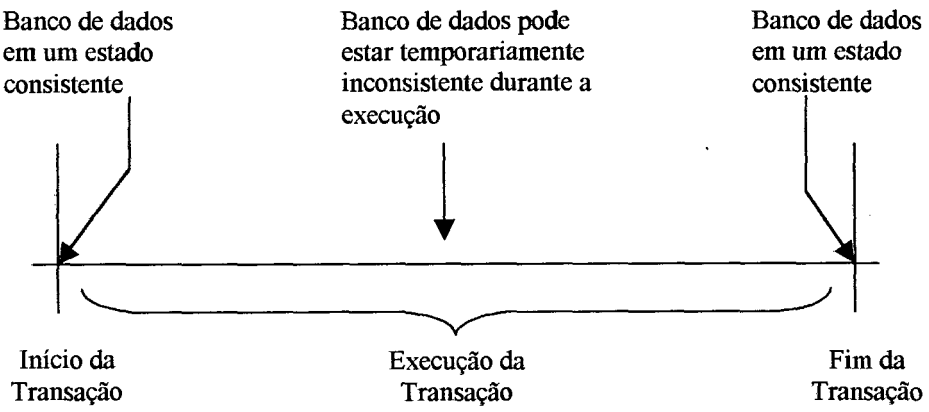

Figura 2.7: Modelo de transação

Várias características dos bancos de dados locais devem ser preservadas; essas características são agrupadas tradicionalmente em quatro propriedades, denominadas propriedades ACID: atomicidade (tudo ou nada), consistência (sem violação das restrições de integridade), isolamento (alterações concorrentes invisíveis e serializáveis) e durabilidade (atualizações confirmadas persistem). Maiores detalhes sobre as propriedades ACID são descritas na seção 7.1.

Transações têm sido classificadas conforme diferentes critérios. Um deles é de acordo com a duração da transação. Elas podem ser classificadas como *on-line* (*short-life*) ou *batch* (*long-life*). Transações on-line são caracterizadas por possuírem um tempo de execução curto e por acessarem poucos dados do banco de dados. Transações batch levam muito mais tempo para serem executadas por acessarem uma grande parte do banco de dados.

O gerenciador de transações (figura 2.5) é responsável por coordenar a execução das operações, enquanto o escalonador é o componente responsável pela implementação de um algoritmo específico para controle de concorrências das transações.

2.8 Controle da concorrência em sistemas distribuídos

O controle de concorrência é o mecanismo que permite diversos programas acessarem um conjunto de recursos compartilhados. O objetivo do controle de concorrência é realizar o compartilhamento de todos estes recursos de forma transparente. Desta forma, cada programa pode se comportar como se todos os recursos estivessem a sua disposição, cabe ao controlador de concorrência intermediar o acesso aos recursos compartilhados e garantir que cada programa possa ser executado de forma independente de todos os outros [Ferreira e Finger, 2000].

O mecanismo de controle de concorrência distribuída deve sincronizar transações concorrentes de maneira que a consistência do banco de dados seja mantida, enquanto o grau máximo de concorrência é obtido. Portanto, o objetivo do controle de concorrência é garantir que as transações sejam executadas atomicamente, isto é, se a transação terminar com sucesso todas as alterações serão permanentes, caso contrário ela não deve ter nenhum efeito e todas as alterações deverão ser desfeitas.

Algoritmos para o controle da concorrência são classificados em pessimista e otimista. Algoritmos pessimistas partem da premissa de que as transações conflitam entre si e estabelecem protocolos para acesso exclusivo aos dados pelas operações, assegurando o isolamento das ações da transação e a correção durante a execução das transações.

Os algoritmos otimistas partem da premissa de que não existem conflitos entre as transações, possibilitando livre acesso aos dados. Ao final da execução, através da avaliação do histórico produzido, verifica-se se houve violação do isolamento. Caso tenha havido violação, a transação é desfeita.

Nos algoritmos baseados em bloqueios uma transação bloqueia um objeto antes de usá-lo. Caso um objeto bloqueado seja requisitado por uma outra transação, esta deve aguardar até o objeto ser desbloqueado. As transações indicam suas intenções de solicitar bloqueio de dados para um gerente de bloqueios.

Outra classe de algoritmos são os de pré-ordenação (*Timestamp Ordering*). Para uma transação é atribuída uma etiqueta de tempo global. O gerente de transação

anexa a etiqueta de tempo a todas ações enviadas pela transação e operações conflitantes são resolvidas pela ordem da etiqueta de tempo.

Existem vários algoritmos para o controle da concorrência. Todos eles têm como objetivo manter as propriedades de isolamento e consistência de transações. Maiores detalhes sobre algoritmos para controle de concorrência são descritos no capítulo 8.

2.9 Principais áreas de estudo em banco de dados distribuído

Nos últimos anos, vem surgindo a cada dia novas aplicações com banco de dados despontando várias áreas para pesquisa. Destacam-se [Özsu e Valduriez, 1999] as seguintes áreas como as mais pesquisadas:

- *Projeto de banco de dados*: projeto de bancos de dados distribuídos incluem os mesmo princípios de projeto de bancos de dados centralizados, mas envolvem outras decisões, como o lugar dos dados e dos programas através dos sites da rede. Para que essa distribuição não seja muito custosa, é necessário que o projetista considere algumas questões na fase de projeto, como: quanto fragmentar, como fragmentar, como testar os resultados, como alocar os fragmentos.
- *Processamento de consultas distribuídas:* esta área lida com o desenvolvimento de algoritmos que analisam as consultas e as convertem para uma série de operações de manipulação de dados. Trata o problema de como decidir a melhor estratégia para executar consultas globais de modo eficiente e efetivo.
- *Gerenciamento de diretórios distribuído:* Um diretório contém informações sobre os itens de dados de um banco de dados. Um diretório pode conter informações sobre todos os dados dos bancos de dados ou apenas informações de cada *site* local. Pode também ser centralizado em um

único site ou distribuído em vários *sites* e ter uma única cópia ou várias cópias.

- *Controle da concorrência distribuída:* esta área é, sem dúvida, a mais estudada. O problema de controle de concorrência em sistemas distribuídos acrescenta muitos desafios em relação aos centralizados. Apesar de já existirem muitas alternativas de soluções, inúmeras pesquisas são realizadas a fim de descobrirem soluções ainda melhores.

- *Gerenciamento de deadlocks*: o problema de *deadlock* surge quando uma transação, para poder continuar seu processamento, espera por uma outra transação que não está disponível. Este problema é similar ao problema de *deadlock* encontrado em sistemas operacionais. São pesquisadas alternativas de prevenção, como evitar, detecção e recuperação.

- *Recuperação de falhas*: quando uma falha ocorre, e um ou mais *sites* ficam inacessíveis, os bancos de dados dos *sites* operacionais devem permanecer consistentes e atualizados. Além disso, quando a falha for recuperada, o sistema deve ser capaz de atualizar os sites que estavam inoperáveis.

- *Bancos de dados Heterogêneos:* quando não existe homogeneidade (quanto aos modelos de dados, linguagem de acesso, etc) entre os vários bancos de dados surge a necessidade de uma tradução para acessar os vários *sites*. Estes sistemas heterogêneos são conhecidos como sistemas de múltiplos bancos de dados (*multidatabase systems*) e impõem novos desafios e problemas a serem solucionados.

2.10 Comentários Finais

A descentralização das corporações e o avanço da tecnologia de rede nos anos oitenta fizeram com que surgissem os bancos de dados descentralizados, ou distribuídos.

Neste capítulo, foram apresentados alguns tópicos básicos referentes a sistemas de bancos de dados distribuídos, a fim de revisar conceitos necessários para o entendimento do restante deste trabalho.

Por fim, descrevem-se as principais áreas de estudo em sistemas de bancos de dados distribuídos. A última área citada, bancos de dados heterogêneos, é o assunto tratado com maiores detalhes neste trabalho.

O capítulo seguinte define sistemas de bancos de dados distribuídos heterogêneos, destacando suas arquiteturas e terminologia.

Capítulo 3

ARQUITETURAS PARA DISTRIBUIÇÃO DE DADOS

A necessidade de troca de informações entre organizações independentes através de meios digitais fez com que surgissem diversas possibilidades para a distribuição de informação.

As possíveis alternativas para a distribuição de dados estão classificadas [Özsu e Valduriez, 1999] e levam em consideração as três dimensões: autonomia, heterogeneidade e distribuição (figura 3.1).

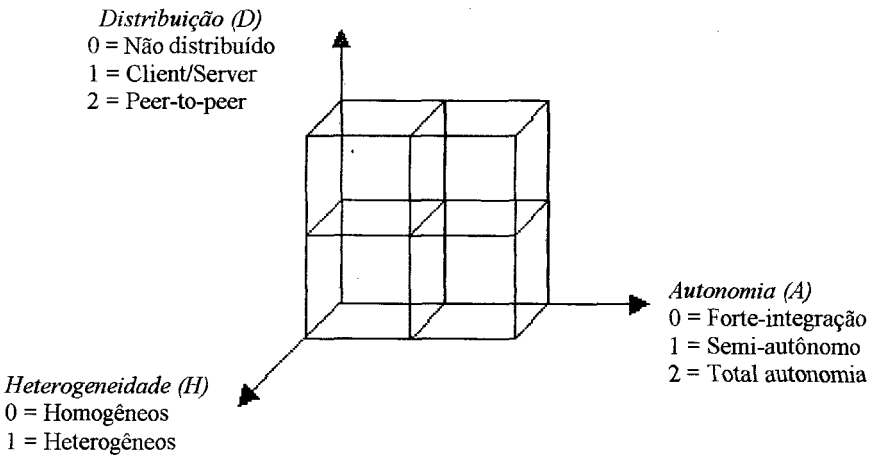

Figura 3.1: Alternativas de implementação de SGBDs

O conceito de autonomia pode ser entendido como a capacidade que cada sistema gerenciador de banco de dados (SGBD) possui de operar independentemente. Autonomia refere-se à distribuição do controle (e não de dados).

Define-se [Gligor e Popescu-Zeletin, 1986] como um sistema autônomo aquele que cujas operações locais (como processamento e otimização de consultas) não são afetadas por outros sistemas de bancos de dados, e a consistência e/ou operação do sistema não é comprometida quando um SGBD é acrescentado ou retirado da confederação.

Autonomia é um fator de grande importância, pois quanto mais independentes forem os sistemas componentes, menor será a necessidade de um controle de transação global. A autonomia pode ser medida através do grau com que um SGBD continua a processar as solicitações das transações locais após sua integração a um sistema com múltiplos bancos de dados (*multidatabase*).

De acordo com a taxonomia proposta por Özsu e Valduriez, existem três tipos de autonomia: forte-integração, semi-autonomia e total autonomia (ou total isolamento).

Em sistemas fortemente integrados uma visão única de toda a base de dados é disponibilizada aos usuários que compartilham as informações, as quais podem estar fisicamente armazenadas em múltiplas bases de dados. Sistemas semi-autônomos operam independentemente, e participam da federação compartilhando seus dados. Nos sistemas totalmente isolados, os componentes individuais são SGBDs chamados *stand-alone*, pois desconhecem a existência de outro SGBDs, bem como a maneira de se comunicar com eles.

A heterogeneidade pode ser vista como conseqüência da autonomia dos SGBDs componentes. Ela é percebida nos diferentes SGBDs utilizados e na semântica dos dados de cada local. A heterogeneidade pode ocorrer de várias formas em sistemas distribuídos: hardware, protocolo de rede, gerenciadores de dados, linguagens de consultas, modelo de dados, protocolo de gerência de transação, etc. [Özsu e Valduriez, 1999]. Os dados podem ter diferentes interpretações dentro de uma organização. Assim, sistemas podem ser homogêneos ou heterogêneos.

A última dimensão citada, refere-se à distribuição física dos dados nos múltiplos *sites*. Existem várias maneiras como os SGBDs podem ser distribuídos, porém podemos resumi-las em duas classes: *client/server* e *peer-to-peer*. Se considerarmos a opção de não distribuição dos dados, identificamos uma terceira arquitetura.

Client/server concentra o gerenciamento dos dados em servidores, enquanto nos clientes ficam as interfaces e aplicações. Existem *sites* clientes e *sites* servidores, e suas funções são diferentes.

Em sistemas *peer-to-peer*, também chamados de *"flully distributed"*, não existe distinção entre máquinas clientes ou servidoras. Cada máquina tem o SGBD completo e pode se comunicar com outras máquinas para executar pesquisas e transações.

Entre nenhuma e total autonomia, distribuição e heterogeneidade existem muitos níveis intermediários. A partir destas três dimensões, Özsu e Valduriez definiram várias arquiteturas para banco de dados. As alternativas possíveis para cada dimensão foram valoradas com os números 0, 1 e 2. Conforme ilustra a figura 3.1, estes números possuem significados diferentes sobre cada uma das dimensões.

Na dimensão da distribuição, um SBD pode ser distribuído (*Client/Server, Peer-to-peer,* etc.) ou não distribuído (centralizado), enquanto que na dimensão da heterogeneidade, ele pode ser homogêneo ou heterogêneo. No entanto, quando se trata da dimensão da autonomia, as nomenclaturas utilizadas: sistemas de bancos de dados federados e sistemas de múltiplos bancos de dados (*multidatabase*) encontram definições diferentes. Essas nomenclaturas são descritas com maiores detalhes nas seções seguintes.

3.1 Sistemas Homogêneos

Sistemas de bancos de dados homogêneos são aqueles que possuem o mesmo software gerenciador de banco de dados em cada *site* da rede, mesmo se os computadores e/ou os sistemas operacionais sejam diferentes (figura 3.3). Esta é uma

situação comum, pois alguns fabricantes fornecem o mesmo SGBD para computadores de fabricantes diferentes. Estes sistemas são semelhantes aos sistemas centralizados, porém, em vez de armazenar os dados em um único *site*, eles podem ser distribuídos em vários *sites* da rede.

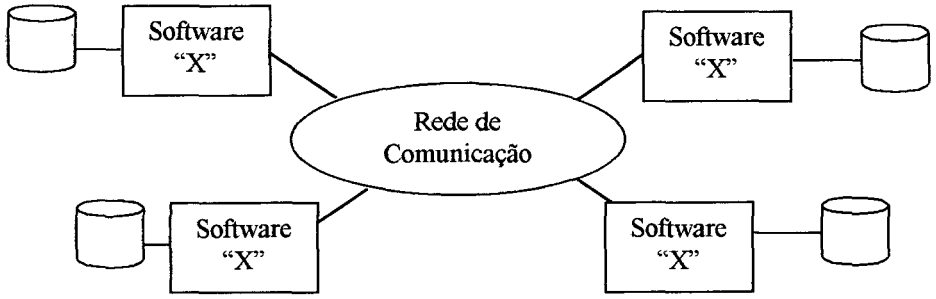

Figura 3.2: Banco de dados Distribuído Homogêneo (mesmo SGBD em cada site)

SGBDDs homogêneos podem ser divididos conforme sua autonomia em três classes: autônomos, não-autônomos e semi-autônomos.

Em um SGBDD homogêneo autônomo existem usuários locais e globais, sendo que cada banco de dados local não conhece a existência de outros bancos de dados e nem como se comunicar com eles. Consequentemente, o usuário local não possui acesso ao resto do sistema. No SGBD não-autônomo os SGBDs locais não operam de forma independente; existe um controle global sobre a execução nos SGBDs locais. Neste tipo só existem usuários globais.

Os sistemas semi-autônomos operam independentes, e participam da federação compartilhando seus dados. Neste tipo de sistema, existem usuários locais e usuários globais, e os usuários locais podem acessar o resto do sistema.

3.2 Sistemas Heterogêneos

Diversas aplicações de banco de dados têm sido desenvolvidas requerendo dados de uma variedade de sistemas de bancos de dados preexistentes, localizados em vários ambientes heterogêneos de hardware e software.

A manipulação de informações localizadas em bancos de dados heterogêneos requer uma camada adicional de software no topo dos sistemas de bancos de dados existentes. Essa camada de software é chamada de Sistema de Gerência de Bancos de Dados Heterogêneos – SGBDH.

Diferentes nomes têm sido utilizados na literatura para designar o sistema que acessa dados localizados em múltiplos e pré-existentes bancos de dados. O esforço para distinção e conceituação de soluções está expresso em diversos trabalhos que sumarizam resultados e apresentam taxonomias. Atualmente, os principais termos utilizados são sistemas de bancos de dados federados e sistemas de múltiplos bancos de dados (*multidatabase*). Essas expressões às vezes são utilizadas como sinônimas, mas correspondem a visões específicas dos problemas e das soluções pesquisadas.

Considera-se um SGBDD heterogêneo [Özsu e Valduriez, 1999] aquele que usa pelo menos dois tipos de SGBDs diferentes (figura 3.4). Portanto, SGBDH fornece transparência não só da distribuição dos dados, mas também dos diferentes sistemas que o usuário acessa.

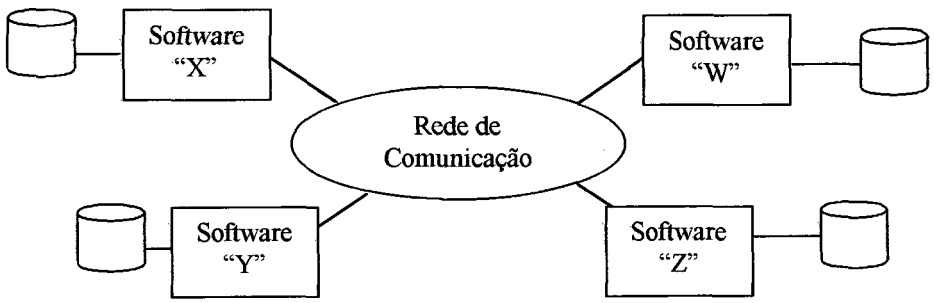

Figura 3.3: Banco de dados Distribuído Heterogêneo (SGBDs diferentes em cada site)

Um SGBDH fornece uma visão integrada que esconde diferenças de estruturas e distribuição do vários bancos de dados. Esta visão integrada é apresentada como uma visão global do banco de dados (esquema conceitual global) e é expressa em algum modelo de dados comum aos SGBDs locais, como o orientado a objetos, entidade-relacionamento ou o modelo relacional.

O SGBDH é responsável pelo mapeamento de dados e operações entre o banco de dados virtual (esquema conceitual global) e o banco de dados local (esquema conceitual local), por resolver diferenças entre modelos, esquemas e sistemas, e por gerenciar as transações distribuídas e o controle de concorrência [Özsu e Valduriez, 1999].

3.3 Banco de Dados Distribuído Heterogêneo

Em sistemas distribuídos heterogêneos, as nomenclaturas mais comumente utilizadas são: sistemas *multidatabase*, sistemas federados e sistemas legados. O termo sistema distribuído heterogêneo é uma generalização destas arquiteturas.

3.3.1 Sistemas *Multidatabase*

Um sistema com múltiplos bancos de dados: *multidatabase* (SMBD) é um tipo especial de sistema de banco de dados distribuído. É formado por uma coleção coerente e integrada de dados que logicamente aparenta ser um único banco de dados mas é implementado fisicamente em vários bancos de dados.

Cada banco de dados participante de um SMBD é autônomo. Os usuários locais dos bancos de dados participantes continuam usando as suas aplicações locais no banco de dados sem nenhuma alteração pela sua participação no SMBD.

Os bancos de dados que participam no SMBD são geralmente heterogêneos e os usuários não precisam saber como ou de onde os dados são acessados. Em um SMBD existem usuários locais e globais (figura 3.4).

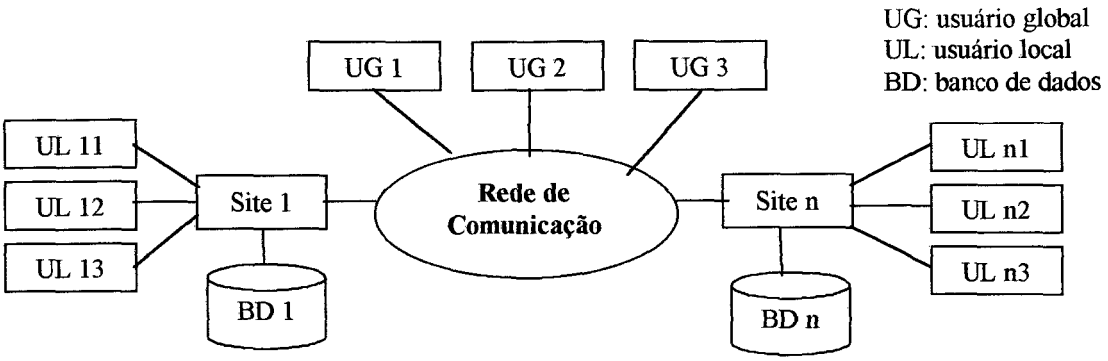

Figura 3.4: Sistema Multidatabase (com usuários locais e globais)

3.3.2 Sistema Federado e não-federado

Consideram-se os sistemas de bancos de dados federados [Sheth e Larson, 1990] como um sub-caso de sistemas *multidatabase*, sendo que os sistemas federados podem ser classificados em fracamente ou fortemente acoplados dependendo do gerenciamento da federação.

Sistemas fracamente acoplados são aqueles que não possuem um esquema global dos dados; já um sistema fortemente acoplado possui um esquema global.

Um sistema de banco de dados heterogêneo com acoplamento forte é composto por um conjunto de SGBDs componentes, heterogêneos, cooperativos mas autônomos, integrados de tal forma na federação que consultas e atualizações podem ser realizadas de forma transparente à localização dos dados e aos caminhos de acesso.

Os sistemas não-federados são *multidatabase* que não possuem usuários locais. O esquema conceitual global é definido através da união de todos os esquemas locais. Desta forma, todos os dados dos bancos de dados locais são compartilhados.

Em um sistema *multidatabase* federado, os bancos de dados locais são semi-autônomos, pois operam independentemente e participam da federação compartilhando parte de seus dados.

O modelo de banco de dados federado é mais flexível pois suporta a autonomia dos bancos de dados participantes da federação. Para organizações descentralizadas, este modelo é ideal porque cada componente do banco de dados controla o acesso a seus dados.

3.4 Sistemas Legados

Sistemas legados [Silva, 1994] (*Legacy Systems*) são aqueles sistemas que estão em uso por muito tempo, que atendem aos requisitos dos usuários e são de difícil substituição ou porque a reimplementação de seu código é inviável financeiramente ou porque eles são imprescindíveis, já que esses sistemas não podem ficar sem execução por muito tempo.

Na maioria das vezes, os sistemas legados foram desenvolvidos em linguagens procedurais, não implementam abstração de dados e não possuem documentação, exceto o código fonte. Tudo isso dificulta a adição de novas funcionalidades e a realização de manutenção no sistema.

Em [Brodie e Stonebraker, 1995] define-se Sistemas Legados como aqueles sistemas que contêm dados valiosos, mas que carecem de poder ou agilidade para satisfazer as necessidades atuais da organização. A necessidade de sobrevivência dos sistemas legados, em sua grande maioria, faz com que várias alternativas tenham sido propostas para resolver, ou pelo menos minimizar esse problema.

3.5 Comentários Finais

Neste capítulo foram apresentadas características das possíveis alternativas para a distribuição de dados conforme a classificação de Özsu e Valduriez, que considera as três dimensões: autonomia, heterogeneidade e distribuição e descrevendo suas características.

Descreveu-se também, o conceito de sistemas homogêneos e heterogêneos, revisando os diferentes termos utilizados para referenciar sistemas heterogêneos.

A evolução rápida, constante e desorganizada da Informática criou nas empresas um ambiente híbrido, onde diversas tecnologias precisam coexistir e se interligar. Várias estratégias têm sido utilizadas para interligar base de dados.

A seguir, o capítulo 4 apresentará as maneiras possíveis para integrar os bancos de dados individuais já existentes e autônomos, destacando dois principais tipos de arquiteturas, aquelas que utilizam ou não um esquema global.

Capítulo 4

ARQUITETURAS PARA INTEGRAÇÃO DE BASES DE DADOS

Normalmente, os dados que empresas e instituições públicas de médio e grande porte desejam compartilhar são dados que estão em bancos de dados heterogêneos já existentes, o que torna a interoperabilidade ainda mais complexa.

Nestes casos, os bancos de dados individuais já existentes são autônomos e é necessário projetar uma forma ideal para integrar tais sistemas. Este projeto de integração executado a partir de base de dados existentes é chamado de *bottom-up*, e diferentes autores apresentam várias alternativas.

A integração de bancos de dados [Özsu e Valduriez, 1999] é o processo no qual informações dos bancos de dados participantes são integrados para formar um único e coeso *multidatabase*. Em outras palavras, é o processo de projetar um esquema conceitual global a partir dos esquemas locais de cada banco de dados participante do *multidatabase*.

A dificuldade na definição do modelo de dados utilizado pelo SGBD heterogêneo [Silva, 1994] é conseqüência da necessidade de se escolher um modelo de dados com poder de expressão suficiente para capturar a semântica dos dados expressa pelos esquemas locais dos SGBDs.

Existem estudos que não utilizam o esquema conceitual global para integrar *multidatabase*. Há discussões se o esquema conceitual global deve existir ou não em sistemas *multidatabase*. Portanto, existem dois modelos que podem ser utilizados para integrar bancos de dados: arquiteturas que especificam um esquema conceitual

global e arquiteturas que não especificam um esquema conceitual global. Ambas são apresentadas nas seções seguintes.

4.1 Integração através de modelos que especificam um esquema conceitual global

Uma alternativa para integração de bancos de dados [Özsu e Valduriez, 1999], [Bell e Grimson, 1992] é através da especificação de um esquema conceitual global a partir dos esquemas conceituais locais. Bell e Grimson classificam os sistemas federados que possuem um esquema global como *fortemente acoplados*. Portanto, um SBDD heterogêneo fortemente acoplado é composto por um conjunto de SGBDs componentes, integrados de forma que a localização dos dados e os caminhos de acesso são transparentes aos usuários.

A construção de um esquema global é uma tarefa difícil e complexa. O esquema global pode ser formado pela união de esquemas locais (figura 4.1). Deste modo, o esquema global será um conjunto formado por esquemas locais.

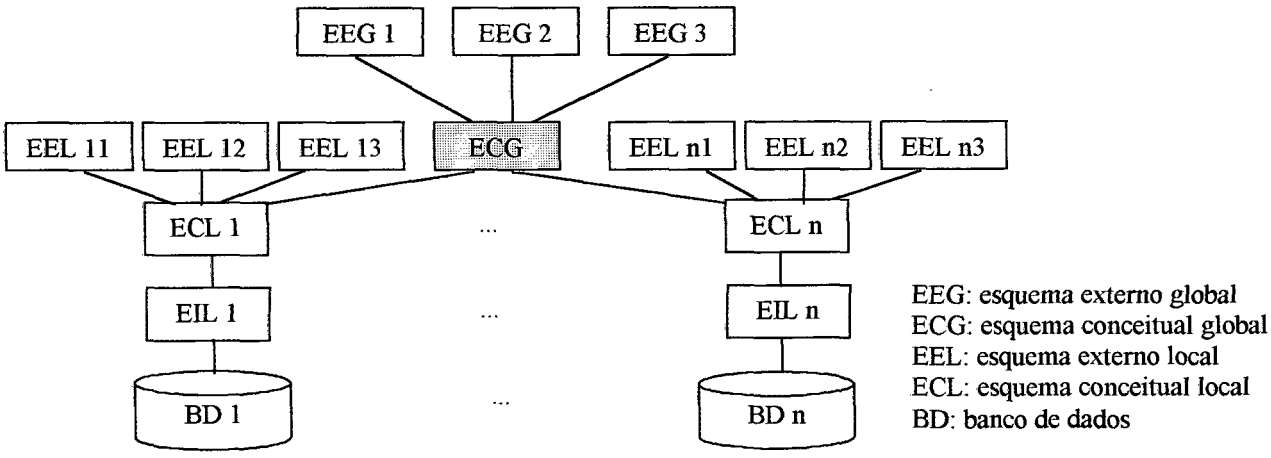

Figura 4.1: Arquitetura com esquema conceitual global

Uma outra alternativa, é a utilização de uma camada extra chamada "*participation schema*" (figura 4.2). Nesta camada, cada banco de dados local define

os dados que deseja compartilhar e o esquema conceitual global é composto pela união de todos os *"participation schema"*.

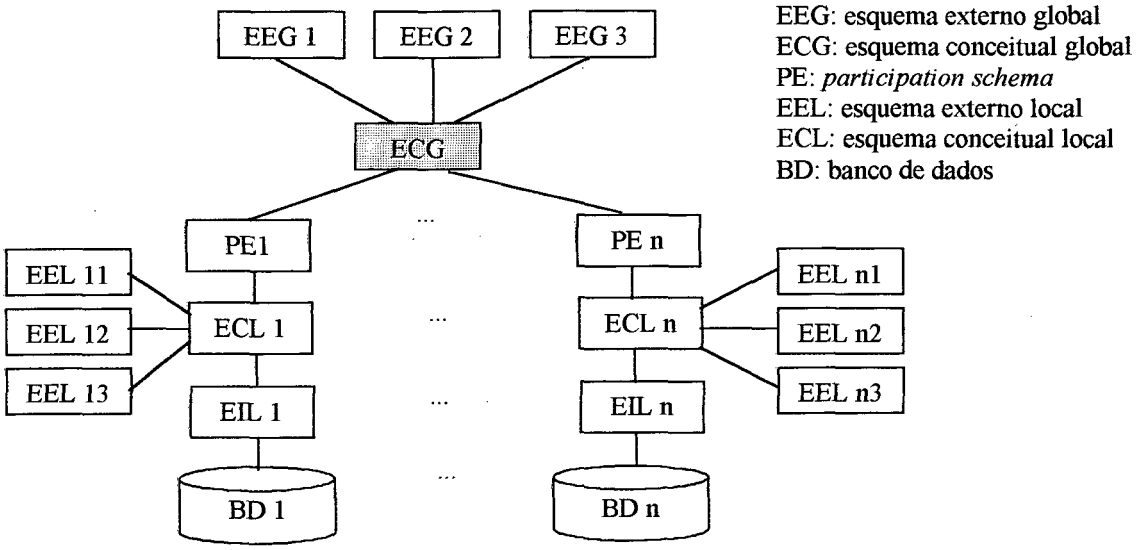

Figura 4.2: Arquitetura com esquema conceitual global e "participation schema"

Deste modo, o esquema conceitual global é um sub-conjunto da união de todos os esquemas conceituais locais, pois é formado apenas por parte dos esquemas conceituais locais. Em ambas alternativas as visões para usuários que requerem acesso global são definidas a partir do esquema conceitual global.

A maior diferença entre o projeto do esquema conceitual global em sistemas distribuídos e este tipo de sistema é que, no primeiro, o mapeamento ocorre do esquema conceitual local para o esquema global. No segundo, o mapeamento é ao contrário, do esquema global para o conceitual. Assim, o projeto de um sistema *multidatabase* é normalmente *bottom-up*, enquanto que nos sistemas distribuídos é *top-down*.

4.1.1 O processo de integração e tradução de esquemas

O processo de integração ocorre em dois passos: tradução e integração de esquemas. A tradução só é necessária se os bancos de dados forem heterogêneos

(cada esquema local foi definido usando modelo de dados diferentes). Nesta primeira etapa, o esquema conceitual de cada banco de dados é traduzido para um esquema intermediário padrão. O esquema intermediário corresponde ao esquema local traduzido para um modelo de dados de uso comum no SGBDH. Deve ser feito um estudo para a escolha de um esquema padrão ideal. O modelo orientado a objetos [Özsu e Valduriez, 1999] é visto, geralmente, como o modelo padrão mais apropriado. Mais detalhes sobre orientação a objeto para interoperabilidade esta descrito no capítulo 9.

Devido à grande comercialização de bancos de dados relacionais nos últimos anos, atualmente há um grande interesse na integração destes SBD*s*. Neste caso, o processo de tradução é desnecessário, pois não existe heterogeneidade entre os modelos de dados.

Na fase de integração de esquemas, realizada em seguida ao processo de tradução, ocorre a integração dos esquemas intermediários gerando um esquema conceitual global.

Existem alguns trabalhos recentes no desenvolvimento de sistemas *multidatabase* federados, em que a fase de integração é realizada em etapas. Primeiramente são integrados os sistemas com modelo de dados similares, e estes são combinados entre si em um próximo estágio. Por exemplo: sistemas relacionais são combinados, sistemas orientado a objetos são combinados, em seguida em uma nova etapa os dois são integrados em um esquema conceitual global.

Integração [Özsu e Valduriez, 1999] é o processo de *identificar* os componentes de um banco de dados que estão relacionados com um outro, *selecionar* a melhor representação para o esquema conceitual global, e, finalmente, *integrar* os componentes de cada esquema intermediário. A figura 4.3 ilustra os processos de tradução e integração.

Várias ferramentas têm sido desenvolvidas para auxiliar o processo de integração. Metodologias de integração podem ser classificadas [Batini et al., 1986] como binária ou n-ária. Binária (figura 4.4) é uma metodologia de integração que envolve a manipulação de dois esquemas por vez. Já na metodologia n-ária (figura 4.5) ocorre a manipulação de mais que dois esquemas por vez.

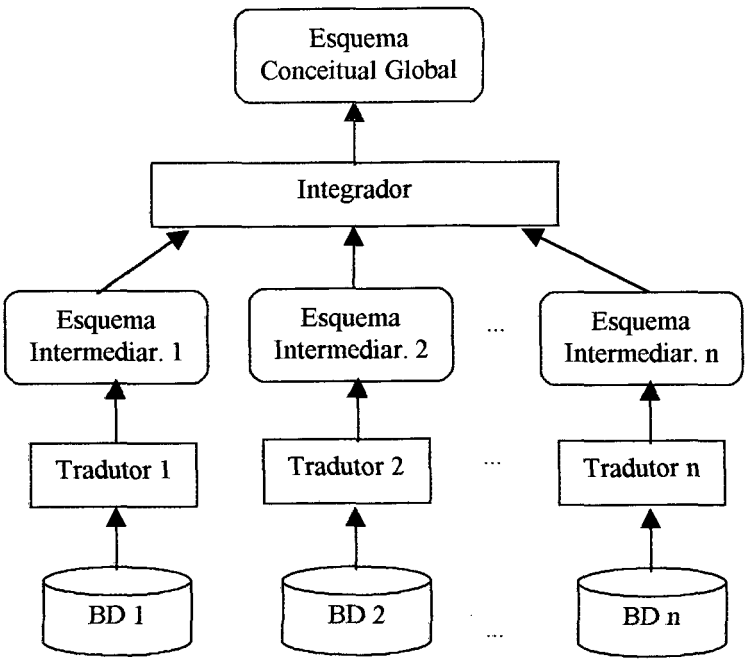

Figura 4.3: Integração de Bancos de dados: tradução e integração

Na metodologia binária, a manipulação dos esquemas pode ocorrer em passos (*ladder*), ilustrado na figura 4.4.a, ou cada par de esquema é integrado, criando um esquema intermediário que será integrado em uma nova etapa, conforme ilustra a figura 4.4.b.

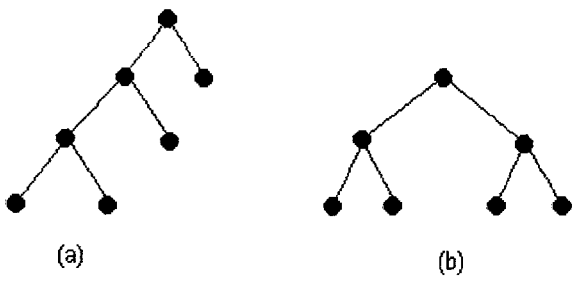

Figura 4.4: Método de integração binária

O método de integração n-ária pode ser implementado de duas formas: integração em um passo (*one pass integration*) e iterativo (*iterative*).

Quando todos os esquemas são integrados de uma vez, produzindo o esquema conceitual global após uma única iteração chamamos de "*one pass integration*". A figura 4.5.a ilustra esta situação. Uma vantagem deste modelo é que durante o processo de integração as informações de todos os bancos de dados estão disponíveis. Porém, tem como desvantagem a complexidade e dificuldade na automação.

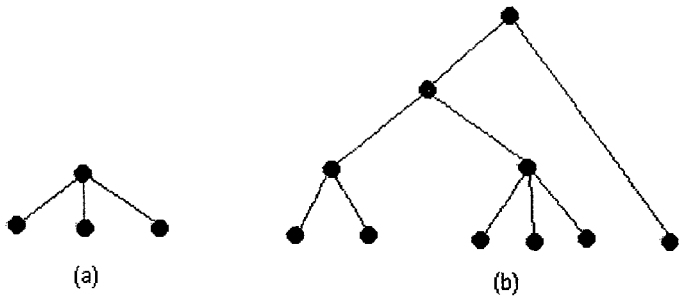

Figura 4.5: Método de integração n-ária

A integração de esquemas pelo método n-ária iterativo ocorre com várias iterações, conforme ilustra a figura 4.5.b. Este modelo oferece mais flexibilidade. O número de esquemas a ser considerados por passo é menor, facilitando sua automação.

Por razões práticas, a maioria dos sistemas utiliza a metodologia binária, porém muitas pesquisas [Elmasri et al., 1987], [Yao et al., 1982] defendem o método de integração em um passo devido à disponibilidade de informações dos bancos de dados a serem integrados.

A integração de esquemas envolvem duas tarefas [Yan et. al., 1997]: homogeneização e integração. Na homogeneização, são tratados os problemas de heterogeneidade semântica e estrutural. Problemas semânticos referem-se ao significado, interpretação e como os dados são usados.

O problema mais importante de heterogeneidade semântica é o de conflito de nomes: *sinônimos* (duas entidades com nomes diferentes, mas com o mesmo

significado) e *homônimos* (duas entidades com o mesmo nome e com significados diferentes). Existem vários métodos alternativos para lidar com conflitos de nomes.

Um dos métodos para solução de problemas de homônimos propostos [Elmasri et al., 1987], é colocar um prefixo no nome do esquema ou do modelo. Para sinônimos, esta solução já não é viável. Na fase de homogeneização ocorre grande intervenção humana, pois ela requer conhecimento semântico sobre todos os esquemas intermediários.

Conflitos estruturais [Batini et al., 1986] podem ocorrer de quatro formas:

- *Conflitos de tipos:* quando um mesmo objeto é um atributo em um esquema e em outro é uma entidade;
- *Conflitos de dependências:* ocorrem quando diferentes tipos de relacionamentos são usados para representar a mesma coisa em esquemas diferentes;
- *Conflito de chaves:* ocorrem quando existem várias chaves candidatas possíveis, e diferentes chaves primárias são selecionadas nos esquemas diferentes;
- *Conflitos de comportamento:* são conflitos implícitos pela modelagem (por exemplo: quando o último item de um banco de dados é apagado, a relação é excluída).

Alguns dos problemas de heterogeneidade semântica e estrutural tratados na homogeneização não são possíveis de serem implementados. Isso faz com que seja essencial a intervenção humana na solução destes.

Após a homogeneização dos esquemas é feita a integração. Os esquemas dos múltiplos bancos de dados (agora esquemas intermediários) são combinados em um único esquema conceitual global e reestruturados da melhor forma possível. Definem-se [Batini et al., 1986] três dimensões para fusão e reestruturação: completeza, minimidade e compreensibilidade.

Uma fusão é completa se todas as informações de todos os esquemas integrados estão em um esquema comum. Uma fusão é não-mínima quando existe redundância de informações no esquema integrado.

Compreensibilidade é a dimensão utilizada para determinar o nível de "entendimento" do esquema final. É difícil quantificar exatamente o que faz algo ser ou não facilmente compreensível, pois este conceito é muito subjetivo. Pode ser necessário fazer um balanço entre minimidade e compreensibilidade, quando a fusão e reestruturação estiverem completas.

4.1.2 Desvantagens do modelo com SCG

Muitas vezes, a heterogeneidade entre os esquemas a serem integrados é muito grande, tornando inviável o investimento no desenvolvimento do esquema global, principalmente quando o número de pesquisas globais são relativamente baixas.

As principais desvantagens deste modelo são:

- Difícil automação.
- Requer conhecimento humano para a solução de conflitos como semânticos, estruturais, etc.
- Não se adapta facilmente a mudanças dos esquemas locais; quando isso ocorre, a integração deve ser refeita.

4.2 Integração através de modelos que não especificam um esquema conceitual global

Os sistemas federados que não possuem um esquema global são classificados [Bell e Grimson, 1992] como fracamente acoplados. Nos últimos anos, tem crescido o interesse neste tipo de sistema, também chamado por alguns autores de "*interoperable database system*".

Foram propostas [Bell e Grimson, 1992] duas alternativas para construir sistemas fracamente acoplados (sem um esquema global). A primeira forma é definir as visões externas aos usuários globais a partir de um ou mais esquemas conceituais locais. Neste modelo, o mapeamento ocorre entre o esquema externo (visões) e o

esquema conceitual local, diferente da arquitetura que usa esquema conceitual global, onde o mapeamento ocorre entre o esquema conceitual global e o local. O acesso a múltiplos bancos de dados é provido por meio de uma poderosa linguagem na qual as aplicações do usuário são escritas (figura 4.6).

Uma outra alternativa para o modelo fracamente acoplado é construir as visões externas definidas aos usuários globais a partir de um ou mais "esquema de exportação" ao invés do esquema conceitual local (figura 4.7).

O esquema de exportação permite que cada banco de dados local defina os dados que deseja compartilhar com outros. Deste modo, tem-se um modelo sem esquema conceitual global e cada banco de dados local define o que deseja compartilhar.

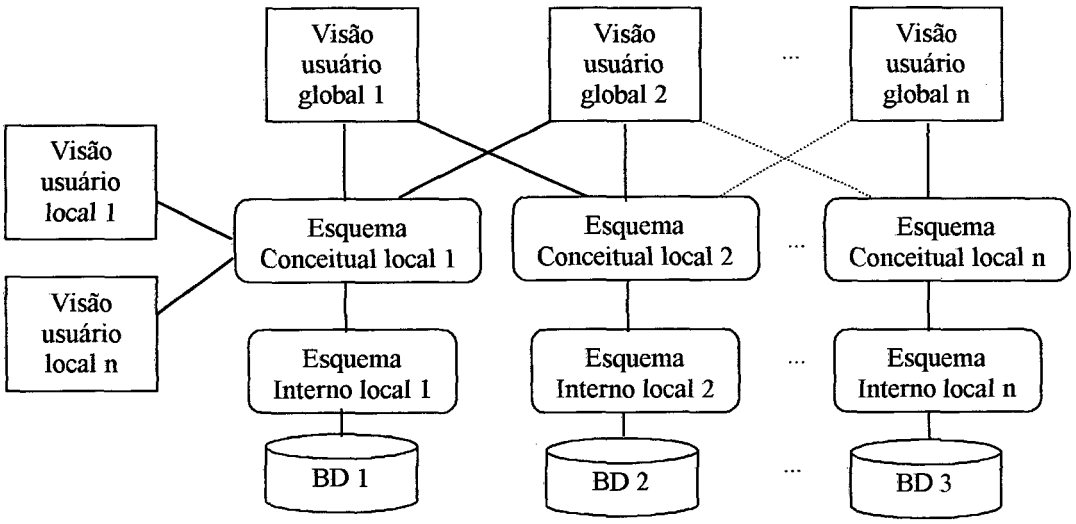

Figura 4.6: Modelo de um multidatabase fracamente acoplado sem export schema

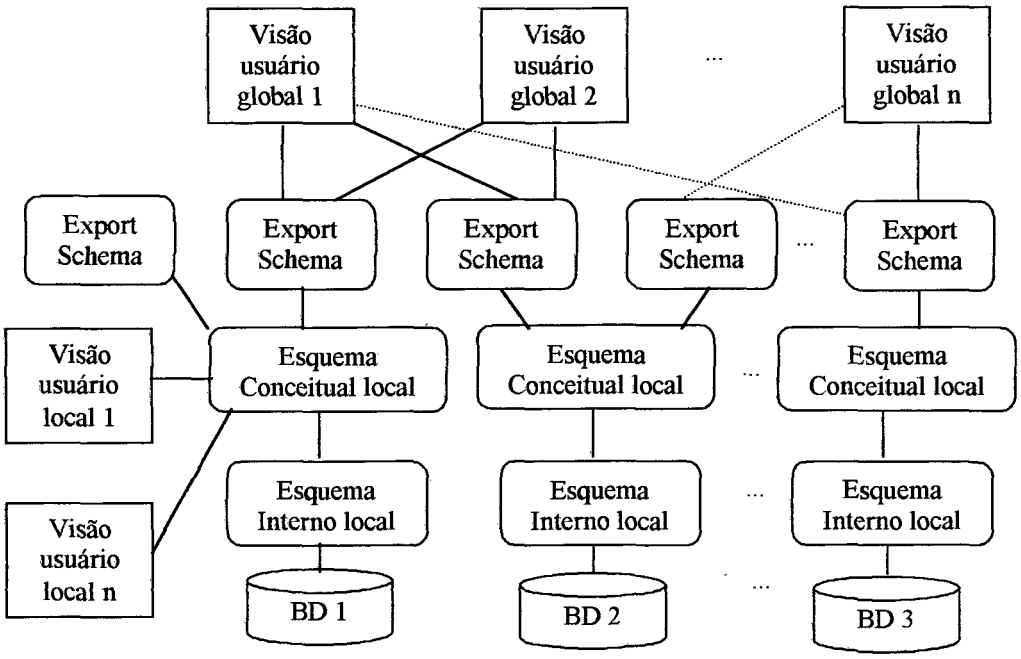

Figura 4.7: Modelo de um multidatabase fracamente acoplado com export schema

Define-se sistemas *multidatabase* [Litwin, 1988] como um gerenciador de várias bases de dados sem um esquema global. Considera a falta do esquema conceitual global uma vantagem dos *multidatabases* sobre os sistemas distribuídos.

4.2.1 Vantagem do modelo sem ECG

Sistemas sem um ECG lidam com mudanças dos esquemas componentes melhor que os fortemente acoplados, pois é mais fácil construir novas visões do que refazer um esquema conceitual global.

4.2.2 Desvantagem do modelo sem ECG

É de responsabilidade do usuário criar e atualizar visões, porém no modelo com ECG é de responsabilidade do gerenciador de sistemas multidatabase.

A tabela 4.1 resume os dois principais modelos de Integração de Base de Dados e suas principais características.

Modelo	Acoplamento Forte	Acoplamento Fraco
Esquema Global	Sim	Não
Vantagens	Alto grau de transparência ao usuário.	Adapta-se facilmente a mudanças.
Desvantagens	- Difícil automação; - Requer conhecimento humano; - Não se adapta facilmente a mudanças.	- O usuário deve conhecer a distribuição para criar visões globais.
Exemplos de SGBDH	Multidatabase, MIND, Jupter e HEROS.	DDTS, Pegasus e Vodak

Tabela 4.1: Principais modelos de Integração

4.3 Linguagens para sistemas multidatabase

Os sistemas multidatabase normalmente utilizam uma linguagem de manipulação global comum aos diferentes *sites* componentes. As consultas globais são escritas nesta linguagem comum e então são traduzidas para os respectivos bancos de dados locais para poderem ser executadas. Várias linguagens para acesso a sistemas heterogêneos vêm sendo propostas. Entre outras, podemos citar GSQL

(*Generalized Structured Query Language*) que é similar ao SQL, MDSL, DDTS, MRDSM, etc.

A tradução da linguagem global para a local é de grande importância para o processamento de consultas em multidatabase. Para facilitar a conversão, informações sobre a sintaxe e semântica destas linguagens devem ser mantidas em um diretório ou em um banco de dados auxiliar. O maior problema na tradução de linguagens é que elas podem não ter a mesma funcionabilidade.

4.4 Comentários Finais

A dificuldade na definição de um modelo integrado fez com que surgissem várias propostas para integrar *multidatabases*. Dentre elas, destacam-se as arquiteturas que utilizam um esquema global e as que não o especificam.

Muitos autores discutem a necessidade de um esquema global, propondo diferentes propostas para integração. Neste capítulo, procedeu-se a apresentação destes modelos, destacando suas principais características, vantagens e desvantagens.

Independente do modelo de integração, consultas globais e locais são requeridas pelos usuários. O capítulo seguinte descreve o processamento de consultas em ambientes heterogêneos sob diferentes aspectos.

Capítulo 5

PROCESSAMENTO DE CONSULTAS

Devido à falta de informação sobre os custos e controle nos bancos de dados locais, o processamento de consultas em sistemas heterogêneos possui complexidades adicionais aos sistemas distribuídos tradicionais a serem tratadas.

Existem várias abordagens para a execução de consultas em um ambiente heterogêneo. Uma das alternativas possíveis é o uso de sistemas multidatabase que suportam um modelo de dados comum e uma linguagem de consulta global acima dos diferentes tipos de sistemas de banco de dados existentes. Estes sistemas de múltiplos bancos de dados utilizam um esquema global que é o resultado da integração dos esquemas exportados dos bancos de dados locais.

Deste modo, um usuário pode fazer uma consulta global em um multidatabase e receber dados originados de vários bancos de dados locais. Graças à transparência provida por um sistema *multidatabase* com esquema global, o usuário não precisa saber onde os dados estão armazenados e nem como a consulta é processada. Entretanto, a homogeneização e integração de esquemas são tarefas extremamente difíceis e nem sempre viáveis. Atualmente, existem muitas pesquisas que buscam soluções para os problemas no processamento de consultas em ambientes heterogêneos sem a integração em um esquema global; uma delas é a utilização da arquitetura de mediadores descrita com mais detalhes na seção 5.2.

Devido à autonomia dos *sites* locais existe uma clara distinção entre uma consulta global e consulta local. Uma consulta local é uma consulta no esquema local de um banco de dados gerenciada pelo SGBD local, sem o controle do

MSGBD (sistema gerenciador multidatabase). Uma consulta global é uma consulta gerenciada por mais de um SGBD local e pelo MSGBD. Quando uma consulta global é submetida, ela é decomposta em um conjunto de sub-consultas locais e seus resultados são agrupados gerando, assim, o resultado da consulta global.

Formalmente, uma consulta global pode ser representada [Morzy e Krolikowski, 1997] por uma quádrupla (R_q, C_{ql}, C_{qg}, A_q), onde R_q é o conjunto das relações envolvidas na consulta a um esquema global, Cq_l é um conjunto de condições locais, C_{qg} é um conjunto de condições globais e A_q é um conjunto de atributos que formam o resultado da consulta.

Conforme a natureza dos sistemas de banco de dados, diferentes abordagens são utilizadas para lidar com o processamento de consultas. O sistema de múltiplos bancos de dados (*multidatabase*) é responsável por traduzir consultas ou atualizações globais de forma apropriada para cada sistema local, realizar o envio para os SGBDs locais que realizam o processamento, agrupar os resultados e gerar o resultado final para o usuário. Além disto, cabe ao sistema coordenar a confirmação ou o cancelamento da transação global que pode ser uma consulta ou atualização.

Considera-se [Özsu e Valduriez, 1999] que o processamento de consultas distribuídas é realizado em quatro passos: decomposição da consulta, localização dos dados, otimização global e otimização local. Uma consulta distribuída [Gardarin e Valduriez, 1989] é uma generalização do processamento de consulta local, que é realizada em três passos: decomposição, otimização e execução. Em sistemas distribuídos (homogêneos), cada banco de dados local possui seu próprio processador de consultas que executa as consultas locais e acima destes, existe uma camada de software que fornece a interoperabilidade entre os *sites* (figura 5.1).

O processamento de consultas em um sistema multidatabase (heterogêneo) é mais complexo. Alguns autores [Bell e Grimson, 1992] destacam que consultas em um sistema *multidatabase* envolvem vários problemas além dos já existentes em sistemas homogêneos. [Sheth e Larson, 1990] atribuem este aumento de complexidade aos seguintes fatores:

- A capacidade de cada componente dos SGBDs locais podem ser diferentes, o que impede um tratamento uniforme sobre as consultas nos múltiplos *sites* e a avaliação de custos necessária para a otimização de consultas.
- Pode haver dificuldades em mover dados entre SGBDs, pois eles podem ter habilidades diferentes em ler os dados "movidos".
- A capacidade de otimização local em cada *site* pode ser diferente.
- A autonomia dos *sites* pode causar problemas, pois um *site* pode não estar disponível no começo, ou pode interromper seu serviço a qualquer momento. Isso requer técnicas de processamento de consultas que são tolerantes a sistemas não disponíveis.

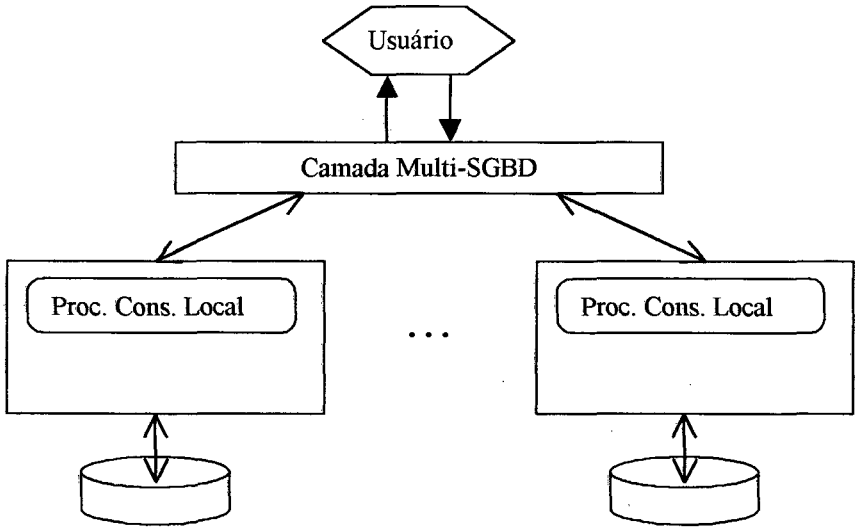

Figura 5.1: Processamento de consultas em sistemas distribuídos

Além desses fatores, [Özsu e Valduriez, 1999] acrescentam que a arquitetura de um sistema gerenciador de múltiplos bancos de dados impõe alguns desafios. Nestes sistemas, há uma camada do MSGBD em cada *site* (figura 5.2). A complexidade aumenta devido ao fato de que a execução de consultas envolve a cooperação entre os vários MSGBDs.

Enquanto em sistemas distribuídos o processamento de consultas lida apenas com a distribuição dos dados pelos múltiplos *sites*, em um ambiente de sistema distribuído heterogêneo os dados estão distribuídos não somente pelos *sites* mas

também sobre os múltiplos bancos de dados, cada um gerenciado por um tipo de SGBD autônomo.

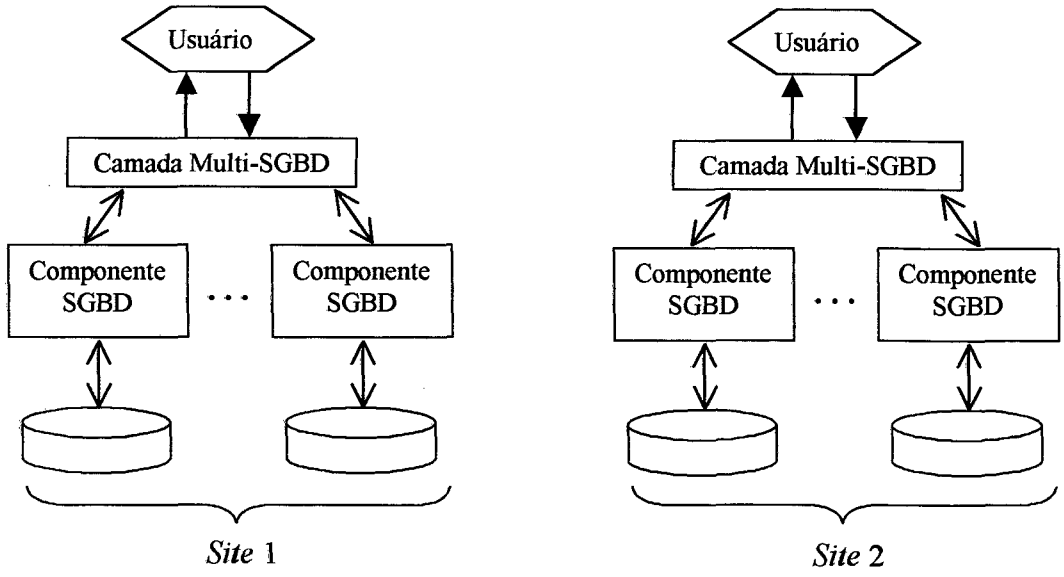

Figura 5.2: Processamento de consultas em sistemas distribuídos heterogêneos (multidatabase)

Portanto, no processamento de consulta distribuída (figura 5.1) existe a cooperação entre duas partes: o processador de consultas local e o processador de consultas global, já em sistemas distribuídos multidatabase (figura 5.2) existem três partes:

- a camada de MSGBD no *site* de controle, *site* que recebe e controla o processamento da consulta global;
- a camada de MSGBD de cada *site*, que participa no processamento da consulta;
- o SGBD componente que, por fim, otimiza e executa a consulta.

Quando um *site* recebe uma consulta global, a linguagem utilizada é uma linguagem para consulta no esquema global. Este *site* que recebe a consulta é chamado de *site* controle e fica responsável pela consulta até que ela seja completada.

5.1 Processamento de Consultas em SGBDs Heterogêneos [Özsu e Valduriez, 1999]

Uma consulta global pode ser executada seguindo algumas etapas (figura 5.3). O sistema gerenciador dos múltiplos bancos de dados decompõe a consulta global em subconsultas, traduz e as envia aos SGBDs locais que executam o processamento local. É de responsabilidade do gerenciador do multidatabase agrupar os resultados parciais e gerar o resultado final para o usuário.

Segundo Özsu e Valduriez, as etapas básicas do processamento de consultas em sistemas multidatabase envolvem os seguintes passos: decomposição da consulta global, fragmentação de cada subconsulta e tradução.

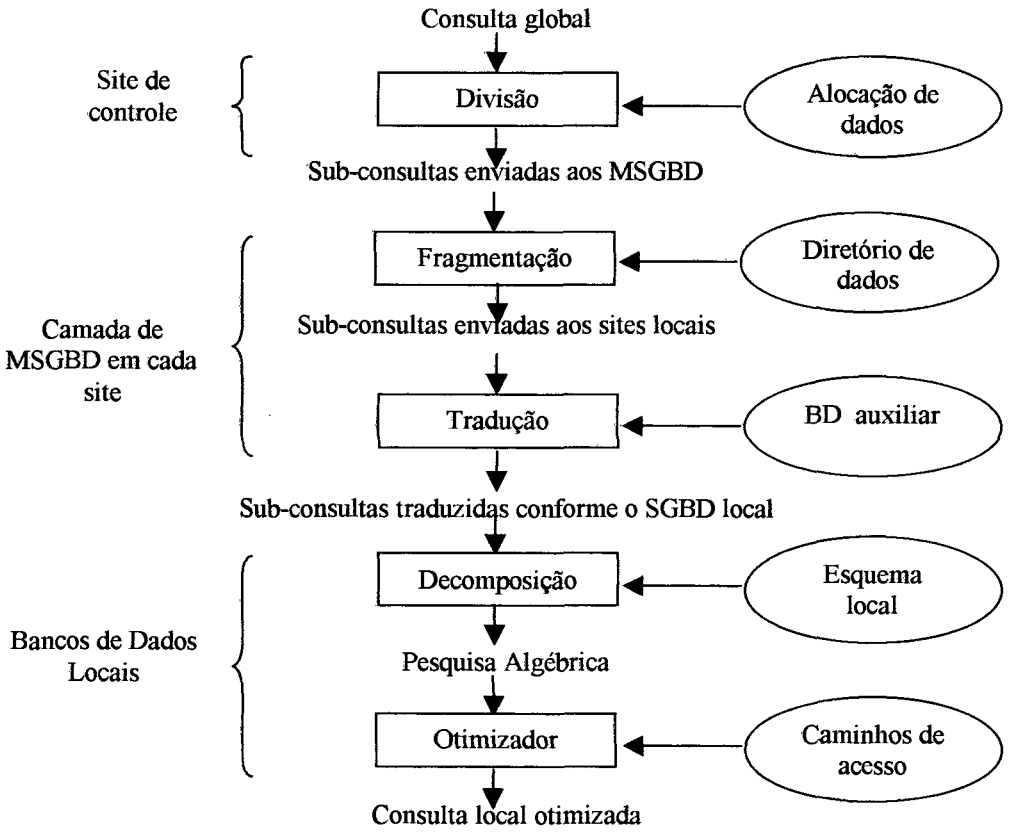

Figura 5.3: Etapas do Processamento de consultas em multidatabase

A primeira etapa no processamento da consulta global é decompô-la em subconsultas baseando-se na distribuição dos dados nos múltiplos *sites*. Nesta etapa, é necessário se preocupar apenas com a localização dos dados pelos *sites*, e a única informação requerida é a localização dos dados que é armazenada em um diretório global.

Após a decomposição, cada subconsulta é então enviada para o *site* onde será processada. Vale ressaltar que, após a decomposição, as sub-consultas ainda estão na linguagem de consulta global.

Em cada site o sistema gerenciador de múltiplos bancos de dados fragmenta novamente cada subconsulta e as envia para cada SGBD componente. Nesta etapa cada subconsulta deve ser traduzida para a linguagem do respectivo SGBD local.

Ao se fazer a tradução da consulta (agora local) que se encontra na linguagem de consulta global para a linguagem de consulta do SGBD local, devem ser mantidas várias informações sobre ambas linguagens (global e local) usada pelos gerenciadores.

Essas informações podem ser mantidas dentro do diretório global (que contém a localização dos dados) ou em um banco de dados auxiliar que permitirá o mapeamento entre o esquema global e os esquemas participantes.

As consultas submetidas aos SGBD são processadas conforme as consultas em sistemas centralizados: decomposição (simplificação), otimização e execução.

5.2 Processamento de Consultas em MSGBD utilizando mediadores

Até bem pouco tempo, os sistemas heterogêneos existentes possuíam relativamente poucos *sites*. Com o surgimento e evolução da internet, formou-se uma grande quantidade de informação digital fazendo com que surgissem sistemas federados cada vez mais heterogêneos e com maior número de *sites*.

Quanto maior o número de *sites* envolvidos em uma consulta, provavelmente será maior a autonomia e heterogeneidade entre os sites e a resposta a uma consulta global requer que todas as fontes de dados envolvidas estejam disponíveis. Além

disso, a manutenção do sistema fica mais complexa para o administrador de banco de dados, pois ao adicionar novos *sites* é necessário alterar esquemas, atualizar dicionários, colher novas informações de custo, etc.

Para que os múltiplos bancos de dados locais possam ser acessados de maneira uniforme, existem pesquisas que propõem uma arquitetura com o uso de mediadores. O mediador recebe a consulta global, transforma-a em subconsultas e as distribui para seus respectivos *sites*. Quando as respostas parciais retornam, o mediador as combina gerando a resposta final para a aplicação.

Os mediadores são desenvolvidos independentes e podem ser combinados fornecendo uma arquitetura para lidar com a complexidade introduzida pelo grande número de bancos de dados componentes. O uso de mediadores permite [Lima e Melo, 1999] a modularização ao invés da centralização. Para tratar a heterogeneidade dos *sites*, *wrappers* (tradutores) traduzem as subconsultas do mediador para a linguagem local de cada banco de dados componentes e vice-versa.

5.2.1 Arquitetura de Mediadores para Interoperabilidade entre Bancos de Dados Heterogêneos

O objetivo da arquitetura com mediadores é prover um modo de acesso ao diversos bancos de dados locais participantes do *multidatabase*, não utilizando o modelo com esquema global.

Alguns autores alegam que em algumas situações práticas, a utilização de esquema global ou uma linguagem *multidatabase* para acessos globais não são soluções apropriadas para atender as necessidades dos usuários. Isto ocorre devido ao fato de que usuários locais podem ter dificuldades em acessar a federação quando não estão familiarizados com o esquema global ou com a linguagem *multidatabase*.

A principal característica do modelo de mediadores é que as regras de transformação são automatizadas, o que permite ao usuário usar a linguagem de seu banco de dados local para fazer consultas globais sem ter que conhecer outros esquemas ou linguagens dos diferentes BDs.

Existem pesquisas realizadas para transformação de consultas entre modelos objeto e relacional [Markowitz e Shoshani, 1993] [Meng et al., 1993]. Estas pesquisas apresentam modelos que fazem transformações bidirecionais entre consultas objeto e relacionais. Em um destes modelos [Markowitz e Shoshani, 1993] foi estabelecido um SGBD relacional como interface (*front-end*) a um SGBD orientado a objetos. Apresentou-se então um modelo de transformação de consultas SQL para OQL e vice-versa.

Na arquitetura de mediadores as diferentes linguagens são transformadas em representações intermediárias e são processadas pelo mediador de consultas. No modelo proposto por Huang e outros, foram utilizadas duas linguagens de consultas: ODMG-OQL e SQL para investigar a arquitetura de mediadores proposta. SQL é a linguagem padrão para bancos de dados relacionais, já para banco de dados orientado a objetos não existe uma linguagem padrão. A ODMG (Object Database Management Group) propôs a ODMG-OQL como padrão.

Ao usar uma linguagem local (nativa) para acessar bancos de dados heterogêneos são requeridos pelo menos dois procedimentos: tradução de esquemas e transformação da linguagem de consulta. A figura 5.4 ilustra uma arquitetura de mediador.

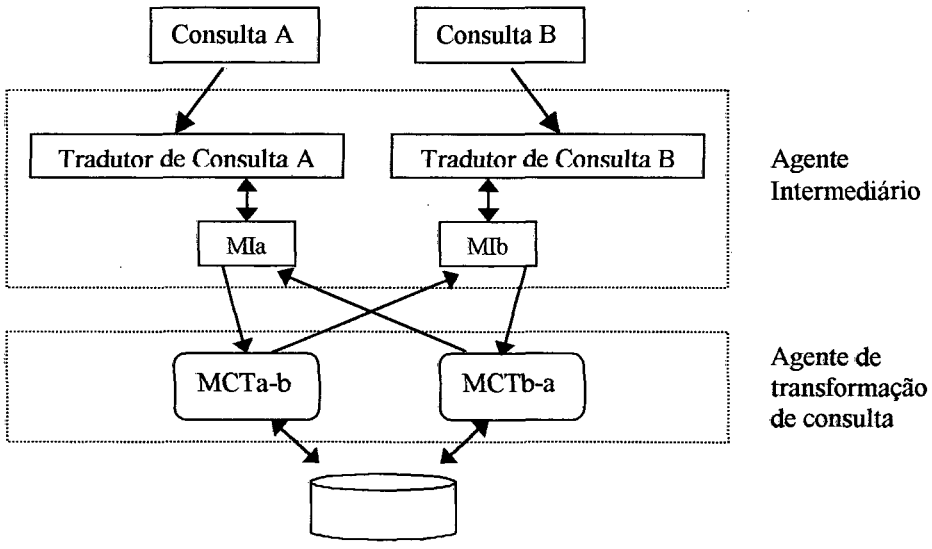

Figura 5.4: Arquitetura de mediador de consultas

O agente intermediário (AI) age como um tradutor de consultas, transformando uma linguagem de consulta nativa de um banco de dados local para um módulo intermediário (MI). Portando cada MI do AI representa uma aplicação de banco de dados local traduzido para uma linguagem intermediária.

O agente de transformação de consulta (ATC) contém os módulos de consultas traduzidas (MCT). Ele é responsável por transformar um módulo intermediário em outro. Para cada par de MI traduzidos terá um MCT que possui um repositório com os dados necessários para o mapeamento entre as linguagens de consultas.

Uma restrição deste modelo é que existe uma certa dificuldade em manter consistência deste repositório, principalmente devido ao fato de que os esquemas normalmente sofrem mudanças. Uma ferramenta adicional pode ser necessária para manter o repositório consistente.

5.3 Comentários Finais

Neste capítulo foram descritas características, complexidades e modelos de processamento de consultas. Foram apresentados o processamento de consultas em sistemas que utilizam um esquema global e a arquitetura de mediadores (sem esquema global). Estes são os modelos mais utilizados por sistemas gerenciadores de bancos de dados heterogêneos.

Ambos fornecem transparência e facilidade de acesso a dados globais aos usuários, porém, devido às dificuldades encontradas na homogeneização e integração de esquemas em um esquema global (capítulo 3), a arquitetura de mediadores tem sido apontada como uma solução mais adequada para sistemas *multidatabase*.

Processamento de consultas em sistemas centralizados e distribuídos (homogêneos) é um assunto bem definido. Entretanto, em sistemas distribuídos heterogêneos ainda há muitos desafios a serem vencidos. Um sistema gerenciador multidatabase, deve não somente processar uma consulta, mas também escolher o melhor plano de consulta.

Para isso, informações locais serão necessárias para a otimização de uma consulta global. Devido a autonomia e heterogeneidade dos bancos de dados locais, tais informações são de difícil acesso pelo MSGBD, surgindo assim um novo problema que vem sendo muito pesquisado: a otimização de consultas em sistemas distribuídos heterogêneos.

O capítulo seguinte trata de questões relacionadas a otimização de consultas, apresentando características, problemas e modelos propostos.

Capítulo 6

OTIMIZAÇÃO DE CONSULTAS

Como em um sistema de banco de dados distribuído tradicional, a otimização de consultas é um importante problema que tem um grande impacto na performance de um sistema multidatabase [Evredilek et al., 1997]. Como processar uma consulta global eficiente é tarefa do otimizador de consulta multidatabase (OCM).

A autonomia e heterogeneidade dos bancos de dados locais de um sistema multidatabase são características que dificultam a otimização de consultas. Isto em função de que informações locais são necessárias para a otimização de uma consulta global e estatísticas de custos de consultas locais não estão disponíveis para o OCM. Além disso, o MSGBD não possui controle direto sobre as sub-consultas executadas nos bancos de dados locais, e portanto, o OCM não tem como prognosticar o tempo de execução nestes. Assim, as soluções aplicadas para otimização de consultas globais em sistemas distribuídos tradicionais não são apropriadas para sistemas multidatabase.

Atualmente existem vários estudos sobre modelos de otimização e processamento de consultas globais em MSGBD. Independente do método utilizado, o objetivo é fazer uma análise do custo das consultas locais, escolher a melhor opção e executá-la. Existem várias técnicas novas propostas nos últimos anos para estimar custos locais e até mesmo modelos de otimização que não utilizam custos locais.

A otimização de consultas em MSGBD [Özsu e Valduriez, 1999] é similar a sistemas distribuídos em alguns aspectos, porém diferente em outros. Considera-se

que assim como em sistemas distribuídos homogêneos, a otimização de consultas em MSGBD pode ser ou baseada em heurística ou baseada em custos.

Para a otimização de consultas baseada em heurística, Özsu e Valduriez relatam duas alternativas que podem ser aplicadas na decomposição da consulta em sub-consultas.

A primeira alternativa baseada em heurística é decompor a consulta global em consultas menores possível, sendo que cada uma delas vai ser executada pelo componente do DBMS. Várias sub-consultas podem ser submetidas a um dado componente SGBD. O processador de consultas global coleta os resultados parciais. Este modelo tem como vantagem o fato de que a decomposição é relativamente simples. A desvantagem é que o processador/otimizador de consultas global trabalha mais, e há mais mensagens transmitidas para executar a consulta.

A segunda alternativa baseada em heurística é decompor a consulta global em sub-consultas maiores possíveis, sendo que cada uma será executada pelo componente do SGBD. Cada componente executa apenas uma sub-consulta. Os resultados, em menos mensagens, são coletados pelo processador de consultas global. Neste caso, o processador/otimizador trabalha menos, já que o processamento entre os *sites* é minimizado.

Na otimização de consultas baseada em custo a diferença está principalmente em termos de funções de custo. A falta (ou dificuldade em obter) informações sobre as operações dos SGBDs componentes geram problemas na definição de custos da consulta global. Definição de custos globais e dos SGBDs componentes é um problema bastante estudado e já existem várias possibilidades de solução.

É interessante determinar o custo de execução de uma consulta das camadas mais baixas da árvore de consulta, que são executadas pelos SGBDs componentes. Três modelos alternativos existem para determinar o custo da execução de consultas nos SGBDs componentes:

a) Tratar o componente SGBD como uma caixa preta, executar alguns testes de consultas neles e a partir disto determinar a informação do custo necessária.

b) Usar conhecimentos adquiridos anteriormente sobre o componente SGBD e subjetivamente determinar o informação do custo.

c) Monitorar o tempo de execução do componente SGBD e dinamicamente colher a informação do custo.

Entre estes modelos, o primeiro tem atraído mais atenção. O custo de funções globais consiste em três componentes: custo de inicialização, custo de recuperação e custo do processamento de tuplas. São processadas consultas exemplo para que os custos sejam medidos.

Resume-se a otimização de consulta global [Morzy e Krolikowski, 1997] em dois passos. No primeiro passo a consulta global é decomposta em um conjunto de sub-consultas. Em um segundo passo os resultados das consultas parciais são unidos através de operações de junção formando o resultado da consulta global. O processo de execução de uma consulta global é especificado por um plano de execução de consulta (PEC). Assim, o problema da otimização da consulta global consiste em selecionar o melhor PEC.

Mais formalmente, Morzy e Krolikowski relatam que o problema da otimização de consulta global pode ser formulado como: dado uma consulta global Q, X_Q consiste em todas as possibilidades de planos de consulta P para Q e uma função de custo $C(P)$ que associa um custo de execução para cada P. Podemos, então, formular o problema de uma consulta global como encontrar no espaço X_Q o plano ou os planos de consultas que possuem menor custo: $min_{P \in XQ} C(P)$.

Portanto, distingue-se claramente dois níveis de otimização: *otimização global*: selecionando o plano de execução global ótimo; *otimização local*: selecionando o melhor plano de execução local. Ambos os níveis estão estritamente interrelacionados. Existem vários modelos de processamento e otimização de consultas em sistemas *multidatabase* os quais, possivelmente se diferem nas possíveis maneiras de como ambos os níveis cooperam entre si [Morzy e Krolikowski, 1997].

Os modelos que propõem planos de consultas, devem ter habilidade em encontrar os custos das consultas nos SGBDs autônomos. Existem vários algoritmos

propostos [Salza et al., 1998] para estimar custos locais. Eles geralmente se baseiam nas duas técnicas: *método de calibração e método de exemplificação.*

A primeira técnica, de calibração, foi proposta por Du e outros, em 1992. A idéia é desenvolver um modelo de custo genérico para os SGBDs locais. Para isso é criado um banco de dados sinteticamente e executado um conjunto de consultas sobre este banco de dados para deduzir os coeficientes das fórmulas de custos dos bancos de dados locais. Os valores dos coeficientes refletem fatores como velocidade de hardware, sistema operacional e configurações dos SGBDs locais.

A Segunda técnica para estimar custos de consultas locais é o método de exemplificação, proposto por Zhu e Larson, em 1994. Esta técnica tenta usar consultas exemplos para encontrar parâmetros para fórmulas de custos locais. A idéia para este método consiste em agrupar todas as consultas possíveis para cada SGBD local em classes e executar uma consulta exemplo para cada classe.

O modelo baseado em estatísticas de custo para consultas globais se torna inapropriado porque é incapaz de se adaptar a mudanças de execução, atrasos inesperados ou *sites* indisponíveis [Morzy e Krolikowski, 1997].

6.1 Aspectos, problemas e desafios propostos para o processamento/otimização de consultas

Processamento e otimização de consultas em sistemas *multidatabases* têm sido estudadas por mais de 15 anos. Vários trabalhos já estão concluídos, porém, ainda existem vários desafios a vencer. Em um futuro próximo, surgirão sistemas federados com 1000 *sites* ou mais. Para a integração destes bancos de dados, ainda existem vários problemas a serem solucionados [Morzy e Krolikowski, 1997] .

É importante destacar que um otimizador de consulta multidatabase (OCM) não pode simplesmente definir o melhor plano para uma consulta global e executá-lo. Isso porque: primeiro, alguns *sites* envolvidos neste plano podem estar indisponíveis; segundo, mesmo o melhor plano pode se comportar mal com a ocorrência de atrasos inesperados na transferência de dados devido a problemas nos *sites* e/ou na rede.

Outro problema é como decidir que objeto entre dois bancos de dados diferentes refere-se ao mesmo objeto no mundo real, principalmente se pensarmos na integração de 1000 bancos de dados.

Consultas em *multidatabase* são normalmente pesquisadas a partir de diferentes aspectos [Lee e Chen, 1997]:

Nível de Operação: a otimização é analisada de acordo com a forma na qual operações de junção e união são executadas [Chen, 1990].

Nível de Álgebra: similar aos sistemas centralizados, a consulta é expressa em uma expressão algébrica e com a utilização de regras esta expressão pode ser representada por um conjunto de expressões equivalentes. A otimização é feita escolhendo uma das expressões que oferecerem menor custo de processamento. Nesta área se incluem as pesquisas em álgebra multirelacional [Grant et. al., 1993] para sistemas fracamente acoplados (sem esquema global) e álgebra hiperrelacional [Lee e Wu, 1996] para sistema fortemente acoplados (com esquema global).

Nível de estratégia de execução: o objetivo deste nível de otimização é estimar o tempo de processamento entendendo o custo de processamento dos sistemas locais participantes. Baseia-se na estimativa de modelos de custos dos bancos de dados participantes para que o sistema multidatabase seja capaz de determinar o melhor plano de consulta [Yan, 1993], [Zhu e Larson, 1994].

Nível semântico: reduz ao custo de execução de consulta através da eliminação/redução de operações ou reduzindo o espaço de pesquisa pela utilização de conhecimento semântico dos bancos de dados. Meng e Yu, são dos poucos autores que se encaixam nesta categoria [Meng e Yu,1995].

6.2 Otimização através de modelos de processamento de consultas híbrido [Getta e Sedighi, 1999]

No plano de processamento de consultas seqüencial uma subconsulta q_i é submetida para um banco de dados local se e somente se os resultados das subconsultas $q_1,...,q_i$-1 já estiverem disponíveis ao site controle. Este método é

proveitoso quando os resultados parciais $r_1,...r_i$-1 podem ser usados para reduzir o tempo de processamento das subconsultas restantes $q_i,...q_i$+k.

O plano de processamento de consultas paralelo submete simultaneamente todas as subconsultas para os bancos de dados locais. É útil quando a complexidade computacional de todas as subconsultas é mais ou menos a mesma e a maioria das subconsultas não pode ser simplificada pelo resultado de outras subconsultas.

A maioria das soluções propostas para a otimização de consultas em sistemas heterogêneos baseia-se no plano de consulta paralelo. Uma solução para a otimização de consultas globais proposta [Getta e Sedighi, 1999] é baseada na conversão do plano de consulta paralelo em planos híbridos (seqüenciais e paralelos). Getta e Sedighi destacam que uma importante vantagem deste modelo é que a otimização provê melhor performance nos estágios de pré e pós-processamento da consulta global.

Um plano para o estágio de pré-processamento determina a ordem a qual as subconsultas são submetidas para os bancos de dados locais. Considerando t o tempo de processamento de um determinado plano de consulta S, o método propõe a transformação de S em um plano de consulta híbrido S' de modo que $t_{S1} < t_{S2}$. Um plano para o estágio de pós-processamento determina a ordem na qual os resultados parciais são integrados para formar o resultado final da consulta global. A integração dos resultados parciais é determinada por uma expressão de integração.

Caso um dos bancos de dados locais por algum motivo demore mais tempo que esperado para retornar o resultado da consulta parcial, os resultados no plano de consulta global otimizado será prejudicado. Para resolver este problema foi proposto uma otimização dinâmica no estágio de pós-processamento.

6.3 Otimização de consultas considerando conflitos entre esquemas [Lee e Chen, 1997]

Diferente das técnicas de otimização de consultas citadas nas seções anteriores, [Lee e Chen, 1997] propuseram um novo estudo sobre os efeitos dos vários tipos de

conflitos de esquema nos custos do processamento de consultas em sistemas *multidatabase*. No modelo proposto, os diferentes tipos de conflitos de esquemas são quantificados e cada um recebe um valor (peso).

Considerando a existência de um esquema global, quando uma consulta global é submetida os atributos e relações a serem acessados são conhecidos, sendo possível analisar os conflitos existentes entre os esquemas participantes da consulta global. Para cada tipo de conflito é definido um peso, sendo possível determinar o melhor *site* para se processar a consulta. Desta forma, surge um novo aspecto estudado na otimização de consultas em sistema multidatabase: o nível de esquema. O nível de esquema (*schema level*) estuda o efeito das diferenças entre esquemas no custo da execução de consultas.

6.3.1 Equivalência semântica entre relações

Em sistemas de banco de dados relacionais, relações são definidas pelo usuário para modelar o mundo real. O relacionamento entre uma relação e seu significado no mundo real refere-se a semântica.

Duas relações são consideradas semanticamente equivalentes [Lee e Chen, 1997] se ambas possuem o mesmo significado no mundo real. Porém, mesmo que os atributos de duas relações sejam os mesmos, isso não significa que elas sejam equivalentes (figura 6.1).

Portanto relações semanticamente equivalentes não precisam necessariamente ter exatamente o mesmo conjunto de atributos. Por exemplo, na figura 6.2, as duas relações *funcionário* e *func* possuem atributos diferentes porem são consideradas semanticamente equivalentes.

Figura 6.1: Relações semanticamente não-equivalentes

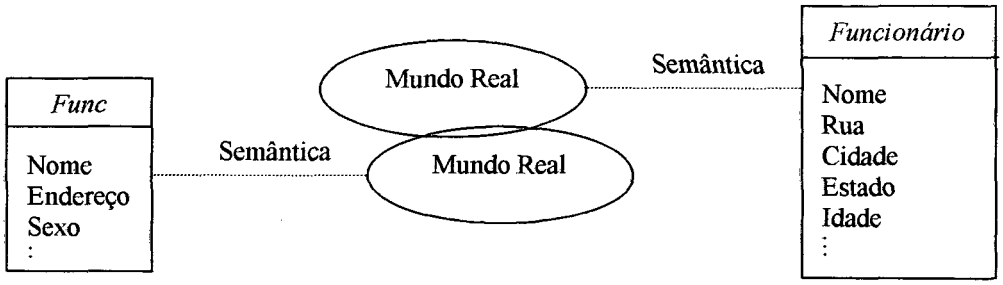

Figura 6.2: Relações semanticamente equivalentes

6.3.2 Conflitos entre Bancos de Dados

Existem vários tipos de conflitos entre bancos de dados. Estes foram classificados [Lee e Chen, 1997] em seis tipos:

- **Conflitos de valor-para-valor** (*Value-to-value conflicts*): este tipo de conflito ocorre quando bancos de dados utilizam representações diferentes para o mesmo valor de dado. Por exemplo, dólar (Americano) versus Yen (Japonês).
- **Conflitos de valor-para-atributo** (*Value-to-attribute conflicts*): ocorre quando a mesma informação é expressa como um valor em um banco de dados e em um outro banco de dados é representado como um atributo. Por exemplo, considerando *sexo* como um atributo em um determinado banco de dados (BD), *feminino* e *masculino* são os dados que serão armazenados neste atributo. Em um outro BD, *feminino* e *masculino* podem existir como atributos. Deste modo, ocorrerá um conflito de valor-para atributo.
- **Conflitos de valor-para-tabela** (*Value-to-table conflicts*): ocorre quando um valor de um atributo em um BD é expresso como uma tabela em outro banco de dados. Por exemplo, considerando *feminino* como um valor para o atributo *sexo* em um determinado BD, e em um outro BD, *feminino* representa uma tabela. Neste caso, ocorrerá um conflito de valor-para-tabela.

- **Conflitos de atributo-para-atributo** (*Attribute-to-atribute conflicts*): estes conflitos ocorrem quando são utilizadas definições diferentes para atributos semanticamente equivalentes. Por exemplo, *endereço* é um atributo na tabela *estudante* de um BD. Na tabela *estudante* de um outro BD, *endereço* é representado por outros três atributos *rua*, *cidade* e *estado*.
- **Conflitos de atributo-para-tabela** (*Attribute-to-table conflicts*): ocorre quando um atributo de um banco de dados é representado como uma tabela em outro banco de dados. Por exemplo, *endereço* é um atributo na tabela *estudante* de um BD, e em outro BD *endereço* é uma tabela.

Na técnica de otimização de consultas proposta por [Lee e Chen, 1997], cada tipo de conflito recebe um peso que será utilizado em uma fase posterior para determinar qual (ou quais) banco de dados será utilizado para executar a consulta global com custo mínimo. Baseadas na complexidade das operações, as regras para atribuir pesos são:

1. Se um atributo 'a' do esquema global e seu equivalente no banco de dados local são representados exatamente da mesma maneira o peso será 0 (zero).
2. Se um atributo 'a' do esquema global possui equivalente em um BD e as operações para tradução for de baixa complexidade então o peso será maior que 0 (zero). Quanto menor a complexidade maior será o peso.
3. Se um atributo 'a' do esquema global possui equivalente em um BD e as operações para tradução for de alta complexidade então o peso será menor que 0 (zero). Quanto menor a complexidade maior será o peso.
4. Se um atributo 'a' do esquema global não possuir equivalente em um BD, ou se uma operação em 'a' não pode ser convertida em nenhuma

operação sobre um equivalente de 'a' no BD, o peso é denominado como 'X'.

Neste modelo de otimização, é de responsabilidade dos projetistas do *multidatabase* dar valores para os pesos. O valor exato de um peso depende de alguns fatores como estratégias de processamento de consultas locais ou mesmo a configuração do hardware de um *site* componente.

Quando uma consulta global é submetida, é computado o peso total das operações necessárias para realizar a consulta para cada um dos planos possíveis. O plano que obtiver o maior peso total, será o que possui menor custo de execução, portanto, o melhor.

6.4 Algoritmos para otimização de consultas globais [Lee et al., 1999]

O foco principal das pesquisas em otimização de consultas em sistemas multidatabase sempre foi em como estimar o custo local de cada SGBD participante da consulta, já que os SGBDs participantes são como caixas-pretas para o MSGBD.

Lee e outros afirmam que muitos fatores afetam o custo do processamento de uma consulta em um SGBD local, o que faz com que os métodos de otimização baseados em custos locais sejam irreais, levando-os a tomar decisões incorretas. Propuseram uma estratégia de otimização considerando um nível mais baixo sem a necessidade de conhecer os custos de processamento dos bancos de dados participantes.

Nesta estratégia, considera-se que para executar uma operação (ex. junção) entre duas relações de BDs locais diferentes, primeiro deve ser feito a conversão dos dados. Para isso, utilizam-se tabelas e softwares especiais para mapeamento entre dados heterogêneos. Deste modo, identificam-se duas estratégias básicas para execução de consultas: (1) os SGBDs locais executam as operações e o MSGBD converte os dados; (2) o MSGBD executa as operações e também as convertem.

Vamos considerar um exemplo para ilustrar as duas estratégias. Considere que G é uma consulta global que efetuará a união de relações A, B e C de três SGBDs diferentes $SGBD_A$, $SGBD_B$ e $SGBD_C$. G é será representada por: $G = A\ x_1\ B\ x_2\ C$.

6.4.1 Estratégia 1: operações executadas pelos SGBDs

Nesta estratégia (figura 6.3), primeiramente o $SGBD_A$ envia a relação A para o MSGBD que irá converter os dados de R em um formato compreensível para $SGBD_B$. Em seguida o MSGBD envia A para $SGBD_B$ que executa a operação $A\ x_1\ B$.

O $SGBD_B$ envia o resultado (AB) para o MSGBD que faz a conversão de AB para um formato compreensível para $SGBD_C$. Em seguida, o MSGBD envia AB para $SGBD_C$ que executa a operação $AB\ x_2\ C$ e envia o resultado final ABC para o SGBD. Finalmente, o MSGBD converte e retorna ABC para o usuário que solicitou a consulta global.

A desvantagem desse modelo é que ele exigirá muita comunicação entre cada SGBDs participante e o MSGBD. O problema se agravará se considerarmos aplicações acessadas através da *World Wide Web* (WWW).

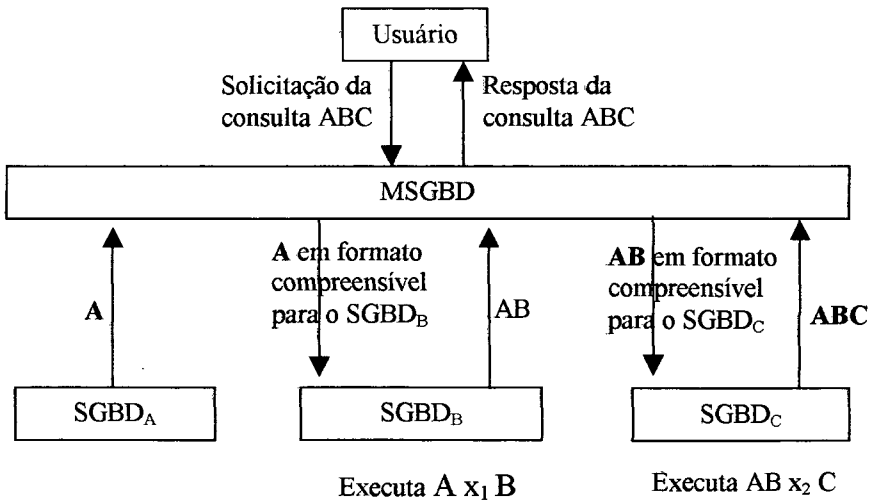

Figura 6.3: Estratégia 1- operações executadas pelos SGBDs

6.4.2 Estratégia 2: Operações executadas pelo MSGBD

A mesma consulta descrita na seção anterior será executada da seguinte maneira. $SGBD_A$, $SGBD_B$ e $SGBD_C$ enviam respectivamente A, B e C para MSGBD simultaneamente. O MSGBD faz a conversão das três tabelas, executa as operações de junção x_1 e x_2 e envia o resultado para o usuário (figura .6.4).

Este modelo não requer muita comunicação entre os SGBDs participantes e o MSGBD. Por isso, considera-se esta segunda estratégia mais apropriada apesar de requerer um grande esforço de processamento do MSGBD. Assim, técnicas para reduzir custos se tornam cruciais para a boa performance de um MSGBD.

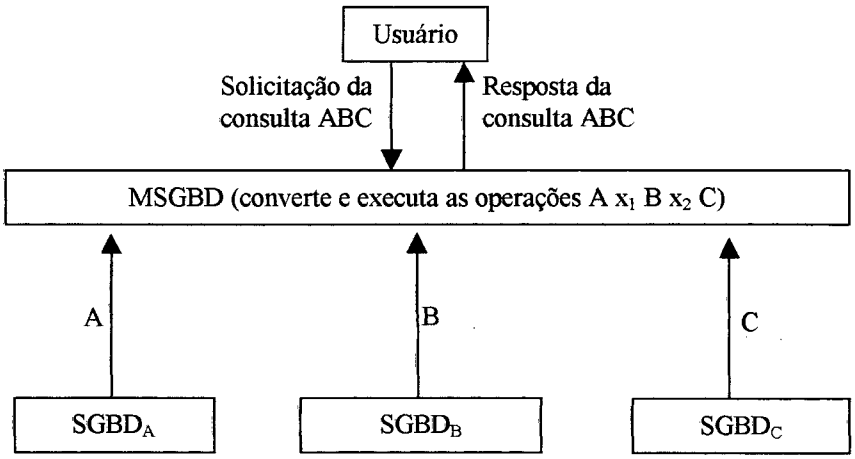

Figura 6.4: Estratégia 2- operações executadas pelos MSGBD

Foram propostos em [Lee et al., 1999] três modelos para redução do custo de processamento de consultas globais em sistemas *multidatabase*. Os dois primeiros métodos são baseados em estratégias tradicionais utilizadas para otimização de consultas: *harsh jpoin* e *nested-loop join*.

O terceiro método proposto é baseado no conceito de *sort-merge join* e possui dois passos principais: separando (*sorting*) as relações e combinando-as (*merging*) para obter o resultado final.

A etapa que combina os resultados parciais é executada pelo MSGBD que lida com discrepâncias entre dados. O objetivo deste modelo é encontrar o número máximo de pares de relações que podem ser unidas no MSGBD.

Este algoritmo destaca-se dos outros métodos propostos por possuir uma estratégia de otimização que não requer o conhecimento de custos locais. Uma outra vantagem é que a carga de trabalho no MSGBD é minimizada.

Uma pesquisa comparativa entre o método *sort-merge* e outros dois métodos de otimização de consultas para ambiente multidatabase foi realizada em [Lee et al., 1999]. A tabela a seguir resume os resultados obtidos. Analisando os resultados obtidos, percebe-se que o método *sort-merge join* é o melhor algoritmo para ambientes multidatabase.

Algoritmos	Custo de processamento nos sites (SGBDs locais)	Custo de processamento no MSGBD
Hash join	Alto	Médio
Sort-merge join	Médio	Baixo
Nested-loop join	Baixo	Muito Alto

Tabela 6.1: Comparação entre métodos de otimização de consultas

Para a seções seguintes vamos considerar que uma consulta global é representada por um grafo $G\ (V,E)$, onde V são as sub-consultas (nós) e E são as uniões (arcos). A figura a seguir mostra um exemplo de união de sub-consultas no MSGBD. O número de nós (relações) e arcos (uniões) são representados por |V| e |E| respectivamente. No exemplo da figura 6.3, |V|= 6 e |E|=5.

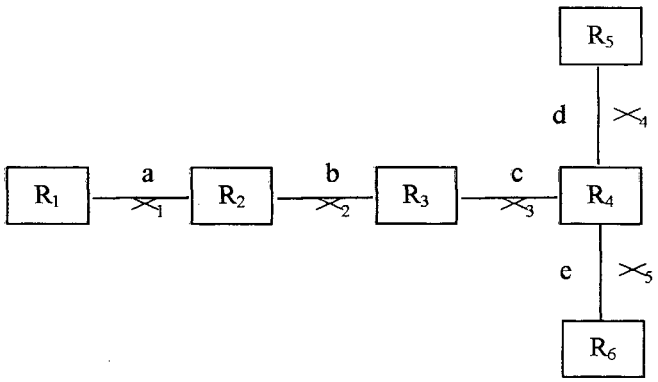

Figura 6.4: Grafo representativo da união de sub-consultas no MSGBD

6.4.3 Modelo *First-Come-First-Serve* (FCFS)

Neste modelo os resultados das sub-consultas que forem recebidas primeiro pelo MSGBD serão as primeiras a serem unidas, ou seja, a união ocorrerá por ordem de chegada. Não existe nenhum conhecimento prévio sobre a ordem que as consultas parciais serão unidas.

Por exemplo, considere que as duas operações de junção R_1xR_2 e R_3xR_4 estão envolvidas em uma consulta. Se R1 e R2 forem recebidas pelo MSGBD antes que R_3 e R_4, ele irá executar R_1xR_2 antes que R_3xR_4.

Esta é considerada um boa estratégia porque normalmente não se tem controle sobre o tráfico na rede. Uma seqüência de uniões pré-definidas poderá prejudicar a performance da consulta global caso uma das sub-consultas a ser unida seja bloqueada pelo tráfico de uma rede.

A seguir, descrevemos mais formalmente o modelo FCFS. A complexidade do modelo FCFS é de ordem |E|, pois |E| (número de arcos) do grafo de consulta G(V,E) representa o número de uniões que deverão ser realizadas.

Algoritmo FCFS :

Entrada: G (V, E);
Saída: uma seqüência de uniões Q;

Repita até |V| = 1;

Início

Escolha $R_i x R_j$, de modo que R_i R_j pertençam a E e Ri Rj já chegaram no MSGBD;

Adicionar (Add) Ri x Rj em Q;

Atualizar grafo G;

Fim.

6.4.4 Modelo Greedy Scheduling (GRS)

O modelo GRS propõe um algoritmo iterativo que seleciona em cada iteração a menor quantidade de dados possível a serem unidos.

Existem critérios que podem ser utilizados para selecionar a próxima sub-consulta a ser unida. Um deles é selecionar a união que resulte (*output*) na menor quantidade de dados. Um outro critério é selecionar a união cuja entrada de dados (*input*) seja mínima.

A seguir descrevemos este algoritmo. A complexidade do modelo GRS é dada por $|E|^2$,pois |E| (número de arcos) é o número iterações e para cada iteração é necessário verificar todos os arcos |E| .

Algoritmo GRS:

Entrada: G (V, E);

Saída: uma seqüência de uniões Q;

Repita até |V| = 1;

Início

Escolha $R_i x R_j$, de modo que R_i R_j pertençam a E e resultem na menor quantidade de dados;

Adicionar (Add) Ri x Rj em Q;

Atualizar grafo G;

Fim.

6.4.5 Modelo Maximum Merge Scheduling (MMS)

Neste modelo o papel do MSGBD é determinar o número máximo de pares de relações que podem ser unidas e instruir aos SGBDs participantes a ordenarem seus resultados. Considerando o exemplo da figura 6.3 o número máximo de pares seria 2, conforme ilustra os pontilhados na figura 6.4.

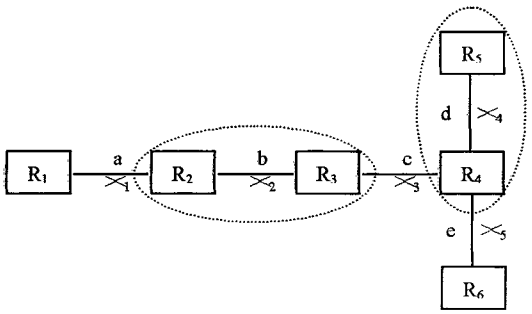

Figura 6.5: Número máximo de pares encontrados pelo método MMS

Podem-se encontrar outras opções de pares de relações. Após marcados os pares, o MSGBD pede aos SGBDs participantes para ordenarem seus resultados por um dado atributo. Por exemplo, o MSGBD pede ao SGBD de R3 ordenar seus resultados pelo atributo *a* (*order by a*) e fará o mesmo para o SGBD de R2. Deste modo, haverá duas uniões simplificadas e parte da tarefa de ordenação ficam para os SGBDs participantes.

Ao comparar os três métodos das seções anteriores [Lee et. al, 1999] propuseram um modelo que gera consultas de maneira randômica. Cada consulta neste modelo é um grafo onde os nós são relações e as arestas são junções. Operações locais são processadas localmente pelos SGBDs participantes. Conseqüentemente, cada nó de um grafo de consulta representa uma relação resultante dos *sites* participantes. Há duas tarefas chaves para gerar esses grafos. Uma é modelar as junções sobre par de relações e outra é modelar a junção de atributos.

A seguir descrevemos mais detalhes sobre os passos destas duas tarefas chaves do modelo proposto.

Passos para modelar pares de relações: considerando N_r o número de relações envolvidas na consulta e p o número de nós que R_j foi conectado.

1. Criar um nó como nó inicial.
2. Denotar o nó inicial como R_j.
3. R_j pode conectar de 1 a 4 novos nós, de acordo com as probabilidades: 1 nó (1/2), 2 nós (1/4), 3 nós (1/8) e 4 nós (1/8), onde o valor entre parentes é a probabilidade. O número de nós de cada consulta não poderá ser maior que N_r.
4. Se o número de nós da consulta é igual a N_r, vá para o passo 5. Se for menor, selecione um novo nó deste grafo como inicial e retorne ao passo 2.
5. Para cada par de nós adjacentes neste grafo, gerar uma aresta para conectá-los.

Passos para modelar a junção de atributos:

1. Ligue atributos com todas as arestas. Deixe todas arestas desmarcadas.
2. Randomicamente escolha uma aresta como a inicial: E_i.
3. Para a união de atributos unidos por uma determinada aresta que não está marcada, chamaremos de E_j, que é adjacente a E_i. Se E_j possuir atributos em comum com E_i, então marque E_j. Após acabarem todas as arestas não marcadas adjacentes a E_i, marque E_i.
4. Se todas as arestas do grafo estiverem marcadas o protocolo pára. Caso contrário, arbitrariamente selecione uma aresta que não esteja marcada, considere-a como E_i e vá para o passo 3.

Lee e outros afirmam que o método proposto possui algumas vantagens sobre os métodos FCFS, GRS e MMS pelos seguintes fatores. Não é necessário conhecer modelos de custo dos bancos de dados participantes, o método proposto reduz a

sobrecarga do MSGBD por distribuir parte das tarefas para os *sites* participantes, além de ser muito mais fácil de implementar.

6.5 Comentários finais

Neste capítulo discutimos aspectos, problemas e modelos propostos para a otimização de consultas em *multidatabases*. Apresentamos vários modelos de otimização segundo vários autores. Nota-se que não existe um consenso do melhor método. Existem muitas abordagens para tratar a otimização de consultas.

Entretanto, percebe-se que os modelos de otimização de consultas globais tradicionais que utilizam informações dos bancos de dados componentes estão sendo substituídos, pois planos de consultas que estimam custos locais pode resultar em planos irreais.

Tendo em vista a existência de operações locais e globais que são executadas simultaneamente, surge o problema do controle de transações. A seguir, o capítulo 7 detalha o gerenciamento de transações.

Capítulo 7

GERENCIAMENTO DE TRANSAÇÕES

Uma transação é vista como uma unidade atômica de computação consistente e confiável, composta por uma seqüência de operações atômicas sobre dados. Como resultado da sua execução, a transação pode causar uma mudança de estado do banco de dados.

Um SGBDH gerencia um banco de dados virtual resultante da integração dos diferentes bancos de dados. Este banco de dados é chamado de banco de dados global, em oposição aos bancos de dados locais, gerenciados pelos SGBDs locais.

As operações executadas sobre o banco de dados global compõem transações globais. As operações globais são transformadas em operações locais e submetidas aos SGBDs que integram o sistema de banco de dados heterogêneo.

Em outras palavras, existem dois tipos de transações em um ambiente multidatabase: transações locais e transações globais. Transações locais são controladas pelos SGBD locais, enquanto transações globais são controladas pelo MSGBD. Cada transação global é um conjunto de subtransações submetidas aos sistemas locais.

A mudança de estado do banco de dados por uma transação é conseqüência da aceitação dos resultados da sua execução (*commit*). No caso da execução ser cancelada (*aborted*), os resultados não provocam mudanças no estado do banco de dados (figura 7.1).

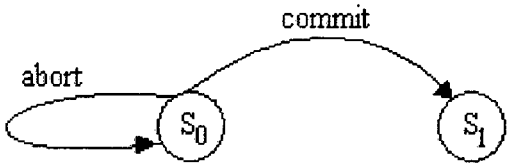

Figura 7.1: Uma transação pode resultar em transições de estado do banco de dados

Uma transação sempre termina, mesmo quando ocorrem falhas. Se uma transação completa seu trabalho com sucesso, esta transação é aceita (*committed*). Caso a transação não possa ser concluída com sucesso – seja porque a própria transação detectou uma condição que impede a conclusão ou por motivos externos, como o envolvimento em bloqueio mútuo com outra transação – a transação é cancelada (*aborted*).

O estado do banco de dados deve ser consistente, isto é, todos os objetos nele armazenados devem obedecer às regras de integridade explicitamente definidas para o banco. Como conseqüência, observamos que as transações são funções que devem mapear um estado consistente para um outro estado consistente.

As transações devem ser executadas concorrentemente, de forma transparente e confiável para a aplicação [Özsu e Valduriez, 1999]. A transparência refere-se à capacidade da manutenção do banco de dados em um estado consistente, mesmo na presença de várias transações executadas simultaneamente. A confiabilidade refere-se à capacidade do sistema de banco de dados em resistir aos diversos tipos de falhas que podem ocorrer no ambiente durante a execução de uma transação.

Considerando que existam vários bancos de dados localizados nos *sites* $S_1,S_2,S_3,...,S_n$, sendo n maior que um, uma transação T_i é uma seqüência de operações de leitura(r_i) e escrita(w_i) concluídas com um *commit* (c_i) ou um *abort* (a_i).

Em um *site* s_i, existe um *escalonador* S_i que armazena a seqüência das transações locais e globais. Duas transações T_i e T_j estão em conflito direto em um *escalonador* S_k se e somente se S_k contém operações $o_i(x)$ seguida por $o_j(x)$, onde $o_i(x)$ é uma escrita em um item de dados de x e T_i não é abortada ou concluída antes T_j ter começado.

Duas transações T_i e T_j estão em conflito indireto em um *escalonador* S_k se e somente se houver uma seqüência de transações T_1, T_2,...,T_n tal que T_i está em conflito direto com T_1 e T_1 está em conflito com T_2..., e T_n está em conflito direto com T_j. As transações T_i e T_j estão em conflito no escalonador S_k se e somente se elas estiverem em conflito direto ou indireto.

A arquitetura de MSGBD envolve um número de SGBDs, cada um com seu gerenciador de transação local (GTL) e uma camada com o MSGBD no topo (figura 7.2). O gerenciador de transação da camada MSGBD é chamado de gerenciador de transações globais (GTG). Uma transação global é dividida em um conjunto de subtransações globais, e cada uma será executada no site correspondente (figura 7.2).

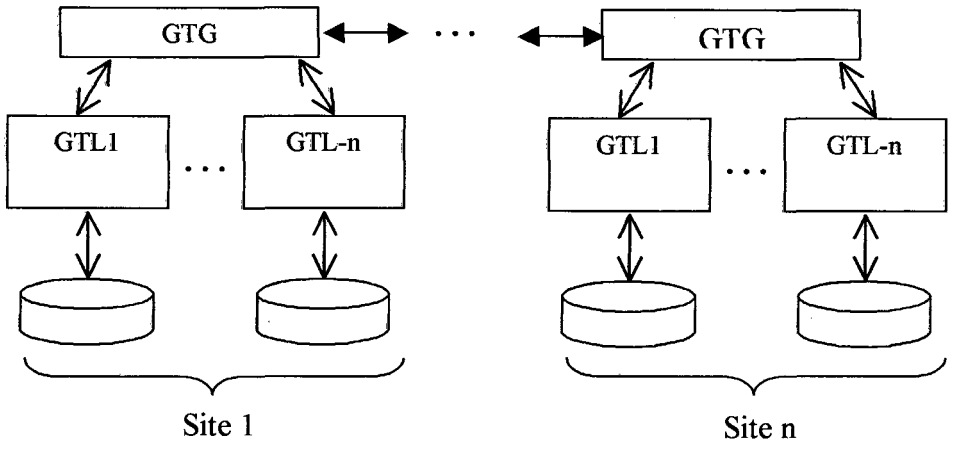

Figura 7.2: Gerenciamento de transações em sistemas multidatabase

7.1 Características das transações

Um dos principais objetivos de um MSGBD é manter os SGBDs locais o mais isolados e intactos possíveis. Várias características importantes dos bancos de dados locais devem ser preservadas; essas características são agrupadas tradicionalmente em quatro propriedades, denominadas propriedades ACID:

- *Atomicidade*: refere-se aos efeitos das transações sobre os estados do banco de dados como unidades de operação únicas e indivisíveis. Isso significa que todos ou nenhum dos seus resultados são considerados. Se uma transação é interrompida por uma falha, seus resultados parciais devem ser desfeitos.
- *Consistência*: é expressa de duas maneiras: a consistência da transação em si mesma e a consistência do banco de dados em face da execução concorrente de transações;
- *Isolamento*: se várias transações são executadas concorrentemente, os resultados devem ser o mesmo como se elas fossem executads uma após outra em alguma ordem. Isto requer que os resultados parciais de uma transação não sejam observáveis por outras transações;
- *Durabilidade*: significa que os resultados da execução de uma transação, uma vez aceitos, o sistema deve garantir que os resultados de suas operações nunca sejam perdidos, a despeito de falhas subseqüentes.

7.2 Modelos de transação

Existem vários modelos de gerenciamento de transações para aplicações específicas. O objetivo de modelos de transação é definir padrões de processamento para um determinado tipo de aplicação. Em [Silva, 1994] são descritos vários modelos para diferentes aplicações. A seguir serão descritos alguns modelos relacionados com aplicações heterogêneas.

A cada dia surgem novas áreas de aplicação com diferentes características que demandam novos modelos de transação. Nota-se que a base destes novos modelos é o da transação plana. Portanto, o modelo de transações planas direta ou indiretamente é a base de todos os outros modelos.

7.2.1 Transações Planas

A maioria dos SGBDs comerciais gerenciam suas transações conforme o modelo de transação plana. Este modelo apresenta uma única camada de controle pela aplicação. As operações são executadas entre delimitadores de início e fim e são aceitas ou rejeitadas em blocos. Deste modo, não é possível aceitar ou cancelar apenas parte das operações de uma transação. Ou ela é efetivada por completo ou totalmente cancelada (figura 7.3).

Transações distribuídas [Silva, 1994] podem ser vistas como transações planas que são executadas em um ambiente distribuído. A estrutura resultante pode ser uma hierarquia de operações que refletem a distribuição dos dados pelos diferentes locais.

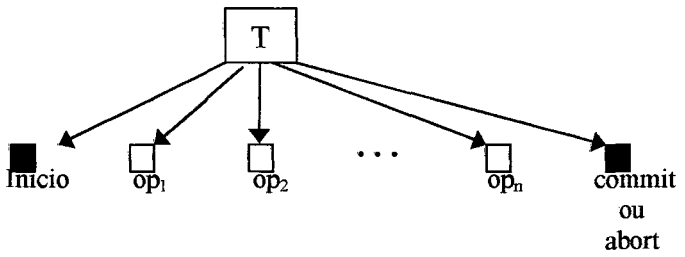

Figura 7.3: Modelo de Transações Planas

7.2.2 Transações Aninhadas

Podem ser vistas como árvores de transações planas, com várias camadas de controle pela aplicação. Uma transação (chamada raiz ou pai) é dividida em subtransações (filhas) que, por sua vez, podem ser novamente divididas em outras subtransações (figura 7.4). Apenas as transações do nível-folha contêm operações sobre dados, enquanto as transações dos nós intermediários fornecem uma espécie de banco de dados de controle para seus descendentes [Silva, 1994].

As sub-transações são classificadas como: vital, não vital ou de contingência. Se uma transação não vital for cancelada, a transação (raiz) ignora o cancelamento e prossegue a execução. Caso uma transação vital é cancelada, a transação raiz pode ser cancelada ou iniciar a execução de uma sub-transação de contingência. Se a meta

da transação raiz não for atingida em um espaço definido de tempo, esta pode ser cancelada.

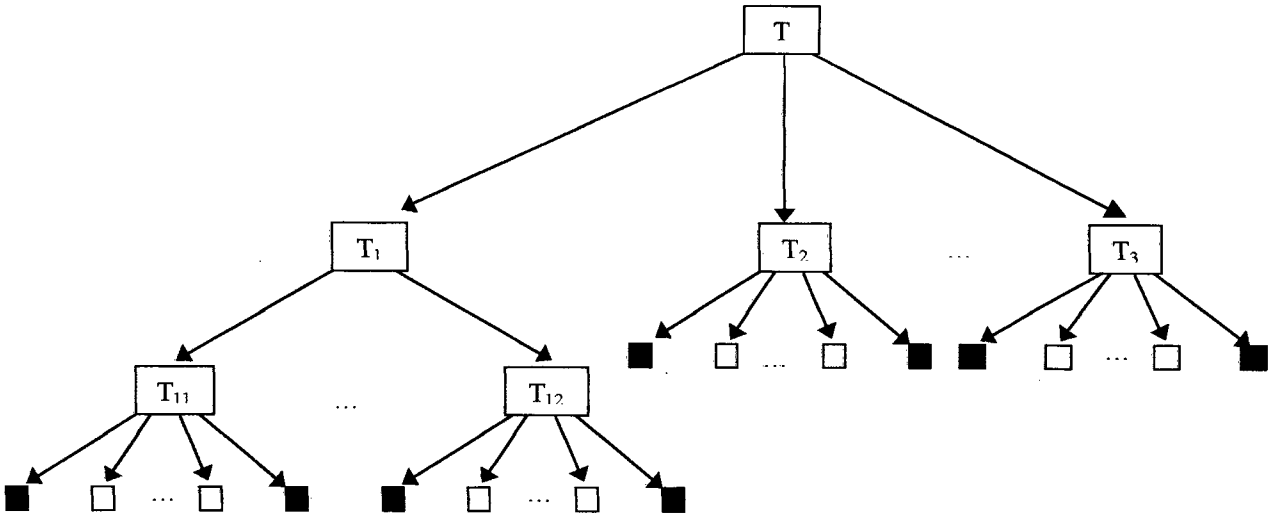

Figura 7.4: Modelo de Transações Aninhadas

7.2.3 Sagas

Este modelo foi proposto como solução para o problema das transações de longa duração. Uma Saga [Silva, 1994] é uma transação de longa duração, composta por um conjunto de subtransações relativamente independentes. Associada a cada sub-transação existe uma sub-transação compensatória, que faz com que o banco de dados não necessariamente retorne ao estado anterior, mas levado a um estado consistente.

O conceito de Sagas [Buchmann et al., 1991] é derivado do conceito de transações aninhadas, caracterizado como aninhamento aberto com dois níveis básicos: transação raiz e subtransações, possibilitando a visão externa do resultado da execução das subtransações à medida em que são aceitos.

O sucesso, ou cancelamento de uma Saga é dado pela última transação componente executada: se ela for aceita, toda a Saga é aceita; caso contrario, se ela for cancelada, todas as transações aceitas anteriormente têm seus resultados desfeitos através de transações compensatórias e a Saga é cancelada (figura 7.5).

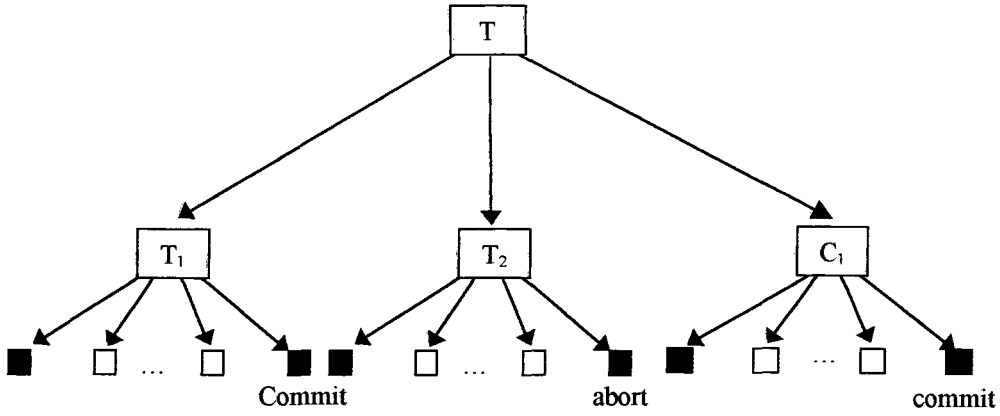

Figura 7.5: Modelo de Transações Sagas

7.2.4 Transações DOM (Distributed Object Management)

Este modelo é indicado para aplicações que integram vários sistemas componentes e possivelmente heterogêneos. Uma transação DOM é uma hierarquia de subtransações que incorpora conceitos de transações aninhadas e de sagas.

O modelo DOM permite que uma transação siga o modelo de aninhamento fechado (as sub-transações não se relacionam), aberto (existe relacionamento entre sub-transações) ou uma combinação de ambos modelos.

É possível a execução concorrente ou serial no modelo DOM, através da especificação das dependências entre as subtransações componentes. Uma teoria da correção dos resultados da execução das transações ainda não foi totalmente desenvolvida.

7.2.5 O modelo Flex de Transações

O modelo Flex [Elmagarmid, 1992] é um modelo adequado ao ambiente de bancos de dados múltiplos e heterogêneos. Foi desenvolvido de modo que uma transação é vista como um conjunto de tarefas e a cada tarefa é associado um conjunto de sub-transações funcionalmente equivalentes, bastando que uma delas seja executada com sucesso para que a tarefa seja concluída.

Neste modelo, utilizam-se transações compensatórias que são associadas pelo programador da transação. Em caso de cancelamento da transação, os efeitos semânticos das sub-transações já aceitas são desfeitos. Cabe ao programador da transação especificar em cada transação os conjuntos de execuções de sub-transações aceitáveis, isto é, que resultem na aceitação global. Como critério de correção são observadas as propriedades ACID, tanto a nível global como para cada sub-transação.

7.2.6 O modelo de Transação S

O modelo de transação S [Veijalainem et al., 1992] foi desenvolvido para atender um sistema bancário com múltiplos bancos de dados heterogêneos com acoplamento fraco (sem um esquema global), onde a autonomia dos bancos de dados correspondentes deviam ser preservados.

Neste modelo, a heterogeneidade dos sistemas componentes é tratada com a criação de uma linguagem homogênea para definição das transações globais. Uma transação global será composta por uma transação raiz-S, transações superiores-S e sub-transações-S. Caso um grupo de sub-transações-S falhar, a transação superior pode requisitar a execução de outro subgrupo de sub-transações-S.

Após ser aceita, uma sub-transação local pode tornar seus resultados visíveis para outra sub-transação-S. No caso de cancelamento, os resultados são desfeitos por sub-transações compensatórias. Uma restrição deste modelo é que caso uma transação-S global falhar, poderá haver necessidade de intervenção humana para restaurar a consistência dos bancos de dados envolvidos.

7.2.7 Modelo de transação do HEROS

HEROS (*Heterogeneous Object System*) é um SGBDH orientado a objetos com acoplamento forte, ou seja com esquema conceitual global. Em uma primeira versão

o sistema de gerência de transações do HEROS foi desenvolvido por Silva em 1994 e foi inspirado no modelo de Sagas.

O modelo de transações HEROS [Silva, 1994] é com aninhamento aberto e permite a modularização de transações, possibilitando que resultados parciais sejam observáveis por outras transações e evitando que em transações de longa duração o acesso aos dados seja bloqueado por longos períodos, favorecendo o uso compartilhado dos dados.

O componente do HEROS que implementa a gerência de transações é o "Sistema de Gerência Global de Transações" (SGG), e por um conjunto de componentes locais, um para cada respectivo local participante da federação do HEROS, os "Subsistemas de Gerência Local de Transações" (SGL).

Os SGL são os responsáveis pela interação com os SGBD's locais. Em cada local, eles representam o sistema global, controlando a submissão de comandos para execução pelo SGBD respectivo e representando o SGBD no processo de aceitação das sub-transações globais. O controle da execução decorrente de transações no HEROS é feito para assegurar a produção de histórias globalmente serializáveis. Para que a história de uma transação global seja serializável, a história das suas sub-transações locais devem ser localmente serializáveis em uma ordem consistente com a ordem global [Castro, 1998].

A estrutura de uma transação proposta em [Silva, 1994] era formada por uma hierarquia de dois níveis: (1) transação raiz, que representa o nó de mais alto nível e (2) transações componentes, os demais nós (figura 7.6). As transações componentes podiam ser dos seguintes tipos:

- Compensável: tipo de transação que pode ter sua execução aceita e posteriormente desfeita através da execução de uma transação compensatória;
- Compensatória: este tipo de transação é executada para desfazer os efeitos de uma transação compensável.
- Pivô: tipo de transação que não é compensável e não é compensatória, isto é, uma vez aceita, não pode ter seus resultados desfeitos através da execução de uma transação compensatória.

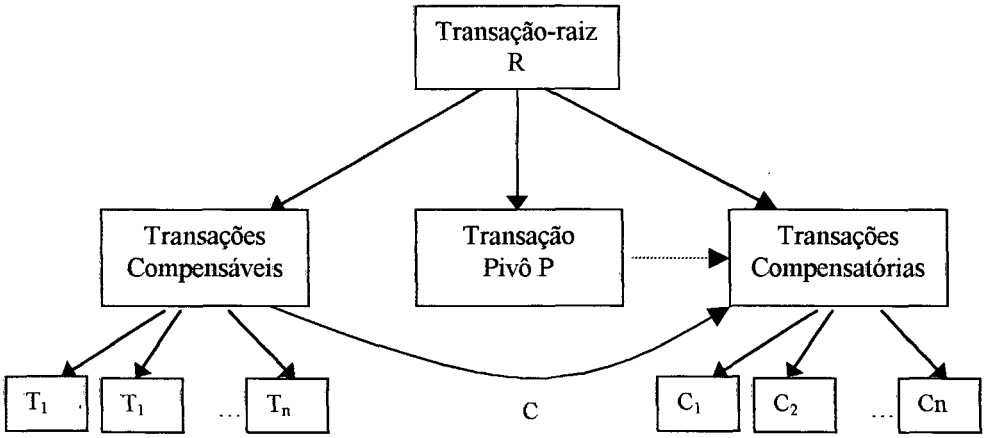

Figura 7.6: Modelo de Transação HEROS [Silva, 1994]

Neste modelo, uma transação só poderá ter uma única transação pivô e seu resultado aceito somente após terem sido executadas todas as transações compensáveis. A aceitação da execução da transação pivô implica na aceitação integral da transação. É possível se ter um número qualquer de transações compensáveis e suas correspondentes transações compensatórias.

Em [Uchoa et. al., 1996] realizou-se uma análise das vantagens de reestruturar o modelo de gerenciamento de transações propostos em [Silva, 1994] para uma arquitetura composta por classes de objetos, usando uma implementação CORBA.

A partir daí, outras pesquisas [Oliveira, 1997], [Castro, 1998], [Uchoa, 1999] foram realizadas, cada uma delas exigindo uma nova extensão da arquitetura HEROS.

7.3 Problemas na gerência de transações em ambiente heterogêneo

O gerenciamento de transações em SGBDs tradicionais é um problema bem definido e com diversas soluções. No entanto, estas soluções têm apresentado restrições no ambiente distribuído heterogêneo e têm sido o foco de muitas pesquisas.

Uma das primeiras pesquisas a questionar essas restrições foi realizada por [Gligor e Popescu-Zeletin, 1986] com uma série de questionamentos que gerariam inúmeros trabalhos com propostas de soluções.

Entre todos os problemas de interoperabilidade, Özsu e Valduriez destacam o gerenciamento de transações como o mais estudado. O desafio é permitir atualizações globais simultâneas sem violar a autonomia do *site*.

Autonomia na execução implica que o gerenciamento de transações globais são executadas independentes da execução das transações locais ocorridas nos SGBDs componentes. Ou seja, o gerenciador de transação de um SGBD não é modificado para acomodar atualizações globais. O GTG não pode detectar os conflitos nos *sites* locais, já que ele desconhece as transações locais.

Nos SGBDs tradicionais, o controle da execução concorrente de transações e a manutenção da consistência dos bancos de dados em caso de falhas é um problema bem definido e com diversas soluções. Na literatura não existem descrições detalhadas da execução da gerência de transações globais por SGBDH.

É freqüente a apresentação de conceitos a serem aplicados para a realização de funções específicas – como a tradução de comandos globais submetidos pelas aplicações em comandos locais submetidos aos SGBDs componentes ou o controle de acesso a dados baseado na comutatividade das operações (o que fazer) – mas não são encontradas descrições detalhadas de procedimentos que possibilitariam sua implementação (como fazer) [Özsu e Valduriez, 1999].

No entanto, a maior parte das soluções disponíveis baseiam-se na existência do projeto de um sistema único e distribuído, homogêneo (*top-down*). Essas soluções não podem ser diretamente aplicadas aos SGBDH que apresentam, adicionalmente às dificuldades causadas pela distribuição e pela réplica de dados, as dificuldades resultantes da autonomia e da heterogeneidade dos sistemas de banco de dados participantes.

7.4 Teoria da Seriabilidade

Serialização é um método no qual é verificado se o resultado final de uma transação é equivalente ao de uma execução serial das operações. Este é um dos critérios de correção da execução de transações mais utilizado. Garantir a seriabilidade é uma tarefa complexa.

Em cada *site* é feito um escalonamento (*squedule*) da seqüência das operações de transações locais e globais. Um escalonador, também chamado de história, é definido sobre um conjunto de transações T = {T1, T2, ...,Tn} e especifica a ordem de execução das operações destas transações.

O escalonador é um elemento fundamental da arquitetura de um SGBD. Ele é responsável por evitar a ocorrência de escalonamentos indesejáveis e de garantir as propriedades básicas dos escalonamentos.

Um escalonamento preserva a ordem interna de cada transação e ordena mais algumas operações entre as transações, incluindo todas as operações conflitantes. Um escalonamento completo deve ordenar todas as operações conflitantes existentes. Na prática, muitos escalonamentos possíveis são indesejáveis, e o controle da concorrência tem como função garantir que só os escalonamentos desejáveis sejam gerados pelo sistema [Ferreira e Finger, 2000].

Considerando as três transações abaixo, um *escalonador* completo é ilustrado na figura 7.7.

T_1: Read (x)	T_2: Read (x)	**T3**: Read (x)
Write (x)	Write (y)	Read (y)
Commit	Read (z)	Read (z)
	Commit	Commit

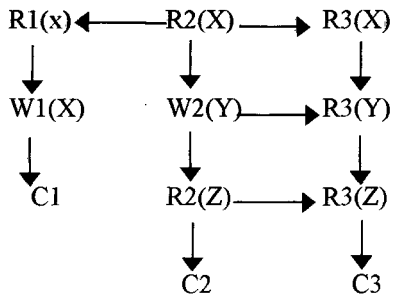

Figura 7.7: Escalonador Completo

A existência de conflitos entre transações indicam que a ordem de execução de tais operações é importante. Se em um *escalonador* S as operações de diferentes transações não são intercaladas, o *escalonador* é considerado serial.

Em outras palavras, um escalonamento [Ferreira e Finger, 2000] é dito serial se para cada par de transações T_i e T_j, todas as operações de T_i precedem todas as operações de T_j, ou vice-versa. A execução serial de um conjunto de transações mantém a consistência do banco de dados.

7.5 Correção, confiabilidade e recuperabilidade de transações

A avaliação da correção da execução concorrente de transações é feita com um critério que indica se o resultado final é ou não aceitável. O critério tradicionalmente utilizado é o da serialização do escalonamento das ações das transações.

A noção de correção baseada na execução serial é natural e decorre da observação de que, se uma transação *T* é executada sozinha por um SGBD desde que *i) T* seja consistente em si mesma, e *ii)* o estado inicial do banco de dados seja consistente, *T* levará o banco de dados a um novo estado, também consistente.

Sempre que duas ou mais transações executadas concorrentemente efetuarem operações sobre um mesmo objeto de dado, há necessidade de se verificar a semântica das operações. Caso conflitem, deve-se estabelecer um critério que

preserve as propriedades ACID. O critério normalmente utilizado para verificar a existência de conflito entre operações de diferentes transações sobre um mesmo objeto de dados é simples: se pelo menos uma das operações for uma operação de escrita (*write*), então ocorre conflito entre as operações. A definição da ordem de execução das operações conflitantes depende do protocolo de sincronismo que estiver sendo observado pelo sistema de gerência de transações.

Escalonamentos são também chamados de histórias por alguns autores [Özsu e Valduriez, 1999]. Em sistemas de banco de dados distribuídos existem várias histórias, uma para cada local (história local) e uma história global, obtida pela união de todas as histórias locais. No caso de sistemas homogêneos, a obtenção de uma história global serializável requer que cada história local seja serializável e que a ordem de serialização seja a mesma em todos os locais onde as transações conflitantes forem executadas.

A confiabilidade na execução das transações está associada às propriedades de atomicidade e durabilidade. Um sistema confiável, mesmo na ocorrência de falhas, deve atender aos requisitos do usuário sem romper as regras de consistência definidas para o banco de dados. Em um ambiente de banco de dados podem ocorrer, basicamente, falhas de transação, de sistema operacional ou do próprio SGBD, de meio de armazenamento e de comunicação.

Se uma falha acontece em meio à execução de uma transação, o SGBD deve ser capaz de executar as ações necessárias para recuperar um estado consistente do banco de dados. O mecanismo usual consiste na guarda em memória permanente (*logging*) de informações para a recuperação.

No caso de sistemas distribuídos, a propriedade atomicidade requer que uma transação tenha seus resultados aceitos ou rejeitados em todos os locais onde suas partes forem executadas. O consenso é atingido através de um mecanismo básico de consultas entre os gerentes de transações de cada localidade.

7.6 Comentários Finais

Os problemas de recuperação e concorrência estão estritamente ligados à noção de processamento de transações. A gerência de transações em um SGBDH tem por objetivo a obtenção de dados consistentes e a preservação da consistência global na presença de atualizações globais.

Nos SGBD distribuídos tradicionais (homogêneos), o controle da execução concorrente de transações e a manutenção da consistência em caso de falhas são problemas bem definidos e com diversas soluções. No entanto, essas soluções têm apresentado restrições no ambiente de SGBDH.

Neste capítulo, definiu-se o processamento de consultas em ambientes heterogêneos, destacando-se as principais características e problemas no gerenciamento das transações. Descreveu-se também, alguns modelos de gerenciamento de transações relacionados com aplicações heterogêneas.

O controle da concorrência entre transações locais e globais será detalhado no capítulo seguinte.

Capítulo 8

CONTROLE DA CONCORRÊNCIA DE TRANSAÇÕES

Quando vários usuários executam transações que acessam objetos que estão armazenados em diferentes *sites* de um sistema distribuído, surge o problema da concorrência entre as transações.

O controle da concorrência de transações tem sido o foco de pesquisas nos últimos 20 anos. Vários problemas e soluções têm sido formalizadas e implementadas em várias aplicações no mundo real [Bhargava, 1999].

Existem várias propostas de algoritmos para garantir a consistência da execução concorrente de transações em ambientes multidatabase. Todos eles têm como objetivo principal manter as propriedades de consistência e isolamento das transações, além de satisfazer um grande número de condições. Devido a autonomia dos SGBDs componentes, existem dificuldades em manter estas propriedades.

O controle da concorrência das transações pode ser feito de vários modos. Portanto, existem várias maneiras de se classificar os modelos propostos para controle da concorrência. Os critérios utilizados para classificação podem ser relacionado ao modo de distribuição dos bancos de dados, topologia da rede, etc. Agrupam-se [Özsu e Valduriez, 1999] os métodos de controle de concorrência em duas classes: métodos otimistas e métodos pessimistas (figura 8.1).

Algoritmos pessimistas partem da premissa de que as transações conflitam entre si e estabelecem protocolos para acesso exclusivo aos dados pelas operações, assegurando o isolamento das ações da transação e a correção durante a execução das

transações. O grupo pessimista, possui três tipos de algoritmos: *locking*, *ordering* e *hybrid*.

Os algoritmos otimistas partem da premissa de que não existem conflitos entre as transações, possibilitando livre acesso aos dados. Ao final da execução através da avaliação do histórico produzido, verifica-se se houve violação do isolamento. Caso tenha havido violação, a transação é desfeita. Este grupo, similarmente ao pessimista, possui dois tipos de algoritmos: *locking* e *ordering*.

Algoritmos otimistas lidam com a fase de validação apenas antes da fase de escrita (figura 8.3), enquanto os algoritmos pessimistas não permitem que transações acessem os dados antes de validar a existência de conflitos nas transações (figura 8.2).

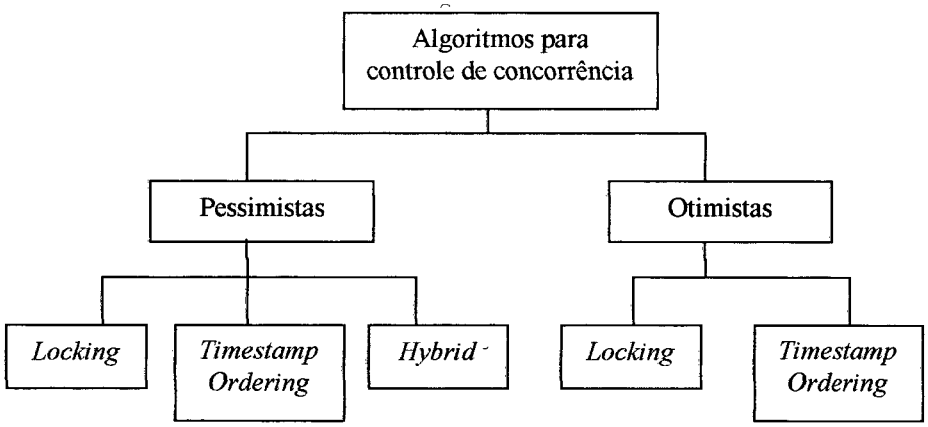

Figura 8.1: Classificação dos algoritmos para controle de concorrência

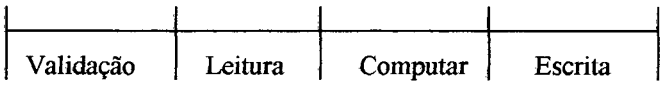

Figura 8.2: Fases de execução de um modelo pessimista

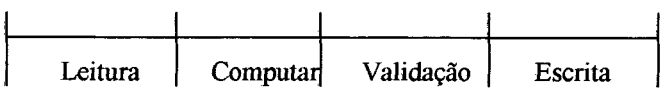

Figura 8.3: Fases de execução de um modelo otimista

Nos modelos de controle de concorrência *locking*, a idéia principal é garantir que um dado que é compartilhado por operações conflitantes seja acessado por uma operação por vez. Para isso, é associada uma "fechadura" (lock) aos dados.

Uma "fechadura" não poderá ser acessada por uma operação se ela já estiver fechada por uma outra. O SGBDD gerencia as "fechaduras" e cuida de toda transação que possuir operações de leitura ou escrita.

Timestamp ordering é um outro método utilizado para controle da concorrência de transações que reinicia uma transação caso ela esteja envolvida em conflitos. Um *timestamp* é um rótulo que serve para identificar cada transação como única e permitir a ordenação.

Em alguns algoritmos os métodos *locking* e *timestamp ordering* são utlizados. Estes são chamados de algoritmos híbridos *(hybrid)*. Este tipo de modelo não é implementado em SGBD distribuídos.

Apesar de ser possível projetar modelos otimistas baseado em "fechaduras" (*locking*), as propostas otimistas são na maioria baseadas no modelo *timestamp ordering*.

8.1 Problemas no controle da concorrência

Algoritmos de controle de concorrência têm o papel de sincronizar as transações concorrentes ordenando as operações que geram conflitos. Duas transações estão em conflito se cada uma delas possui uma operação que acesse o mesmo item de dado e pelo menos uma destas operações é uma escrita (*write*).

Em sistemas *multidatabase* não é fácil para o gerenciador de transações globais (GTG) determinar a ocorrência de conflitos. Um outro problema é o caso de *deadloks* globais. As seções seguintes detalham estes dois problemas.

8.1.1 Conflito Indireto

A principal dificuldade é a existência de conflitos indiretos, isto é, conflitos que ocorrem durante a execução de transações locais nos bancos de dados componentes. Estes conflitos indiretos não podem ser detectados pelo GTG.

Deste modo, duas transações efetuadas pelo GTG podem não apresentar conflitos entre si, porém, a existência de transações locais podem gerar conflitos indiretos. A seguir é descrito um exemplo que ilustra a ocorrência de conflitos indiretos entre transações.

Consideremos que a_1 e a_2 são itens de dados armazenados em um mesmo *site x* de um *multidatabase*. TG_1 e TG_2 são transações globais, e TL é uma transação local sendo executada no *site x* e sua existência é desconhecida pelo GTG:

GT_1: Read (a_1)	**GT_2**: Read (a_2)	**TL**: Read (a_1)
Update a_1	Update a_2	Read (a_2)
Write (a_1)	Write (a2)	$a_2 \leftarrow a_1$
Commit	Commit	Write (a_2)
		Commit

Se olharmos somente para as transações globais TG_1 e TG_2 (ponto de vista do GTG), não existem conflitos. Porém a existência da transação local TL provocou um conflito indireto entre TG_1 e TG_2, que não é reconhecido pelo GTG.

8.1.2 Deadlock Global

Um *deadlock* ocorre quando uma transação, para poder continuar seu processamento, espera por uma outra transação que não está disponível. Informalmente, uma situação de *deadlock* é um conjunto de pedidos que nunca serão concedidos pelo mecanismo de controle da concorrência [Özsu e Valduriez, 1999].

Um método utilizado para detecção de *deadlock* em sistemas distribuídos é analisar o grafo *wait-for* (GWF). Um GWF é um grafo onde os nós representam as transações, e um arco $T_2 \rightarrow T_1$ significa que a transação T_2 esta esperando por T_1.

Na figura 8.4, T_1 e T_2 são iniciadas nos *sites* A e B respectivamente. T_{A2} requer um item de dado xi que esta em T_{A1}, e o ciclo de *deadlock* é completado por T_{B1} que requer um item de dado xj que esta em T_{B2}. Simbolicamente, este *deadlock* poderia ser representado por:

$$G'' = T_{A1} \rightarrow T_{B1} \rightarrow T_{B2} \rightarrow T_{A2} \rightarrow T_{A1}$$

Através do GWF, é mais fácil localizar *deadlock*. Se o grafo tiver ciclos significa a existência de *deadlock*. A formação deste grafo em sistemas distribuídos é mais complexa, pois duas transações que participem do *deadlock* podem estar em *sites* diferentes como o exemplo da figura 8.4. Chama-se esta situação de *deadlock global*. Assim, em sistemas distribuídos em geral é necessário formar um grafo *wait-for* global (GWFG), que é a união de todos os GWF locais. A detecção de *deadlocks* globais não é uma tarefa simples, visto que sua ocorrência pode ter sido acarretada por transações locais, que são desconhecidas a nível global.

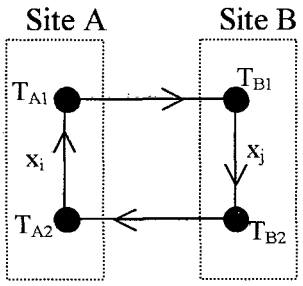

Figura 8.4: Deadlock Global

Algoritmos de detecção de *deadlocks* distribuídos delegam parte da responsabilidade para os *sites* locais. Portanto, existe uma detecção hierárquica, com detectores locais e cada *site* que comunicam seus GWF com os outros sites. Entre os vários algoritmos de detecção de deadlock, o mais conhecido e referenciado foi implementado em System R* [Obermarck, 1982].

8.2 Condições para um controle eficaz da concorrência

Foram definidas algumas condições para especificar quando uma transação global pode fazer atualizações seguras em um sistema multidatabase [Glicor e Popescu-Zeletin, 1986].

Özsu e Valduriez afirmam que essas condições ajudam a determinar a funcionalidade mínima requerida a um gerenciador de transações. Em um SGBD heterogêneo o controle da concorrência deve ser efetuado em dois níveis: global e local.

A primeira condição para prover controle da concorrência global é ter um gerenciador de concorrência local em cada *site* responsável pela execução correta das transações locais. A serialização é o critério de correção mais utilizado.

A segunda condição requer que cada GTL mantenha a ordem de execução determinada pelo GTG. Portanto, o GTG é responsável por coordenar a submissão das subtransações globais para os GTL e coordenar sua execução.

Se o critério para correção utilizado for o de serialização, o GTG é responsável pela serialização das transações globais. Além disso, o GTG é responsável por lidar com problemas de *deadlocks globais* que podem vir a ocorrer entre as transações globais.

Algumas soluções podem ser classificadas em duas categorias: aquelas que consideram cada SGBDs componente como uma caixa preta; e aquelas que assumem algum conhecimento sobre as operações dos SGBDs componentes [Özsu e Valduriez, 1999].

Uma solução que se encaixa na primeira categoria citada, foi proposta em [Georgakopoulos, 1990]. Este modelo resolve o problema de conflitos indiretos convertendo-os em conflitos diretos.

Existem outras soluções para garantir a execução correta de transações. Porém a maioria destas não está bem esclarecida para ambientes heterogêneos.

8.3 Gerenciamento de transação considerando restrições de integridade de itens de dados [Lee e Park, 1998]

Em [Lee e Park, 1998], propõe-se um modelo de gerenciamento de transações considerando restrições de integridade em sistema multidatabase. Essas restrições são aplicadas tanto a nível global quanto local.

8.3.1 Regras de integridade global e local

Sendo um ambiente *multidatabase* formado por um número de SGBDs integrados existem regras para garantir consistência global e local. Regras de integridade são requeridas tanto nos SGBDs locais como no MSGBD. As restrições globais podem ser definidas de acordo com o processo de integração. Cada SGBDs preexistente também define suas regras de integridade sobre seus dados locais.

Os dados de um esquema global (com restrições de integridade global) devem ser gerenciados pelo GTG. Se estes itens de dados puderem ser modificados por uma transação local, ocorrerão problemas na manutenção da integridade global, tais como: (1) uma transação local pode causar inconsistência global, e (2) o GTG não pode saber exatamente quando o valor do dado é localmente atualizado.

Um banco de dados global pode se tornar inconsistente se os dados forem modificados sem preservar suas restrições. Por exemplo, uma atualização local pode produzir inconsistência global. Se o GTG verifica as regras de integridade global periodicamente, uma inconsistência temporária deverá ser tolerada.

8.3.2 Transação global livre

Em sistemas multidatabase é de grande importância verificar onde e como as regras de integridade dos dados globais serão mantidas sem violar a autonomia local.

Para manter integridade global, o GTG deve examinar se o valor de um item de dado é consistente de acordo com as regras de integridade global, pois os SGBDs locais não possuem acesso às regras globais.

Lee e Park dividiram os itens dados de um MSGBD em dois grupos:

- Itens de dados globais: é o conjunto de itens de dados que são definidos nas regras de integridade globais.
- Itens de dados locais: é o conjunto de itens de dados que são definidos nas regras de integridade locais.

Considerando-se D_i o conjunto de dados, LD_i os itens de dados locais e GD_i os itens de dados globais, $LD_i \cap GD_i = 0$ e $D_i = LD_i \cup GD_i$ [Mehrotra et. al., 1991].

O banco de dados global poderá ficar inconsistente se uma transação local atualizar dados globais sem ter o conhecimento das restrições de integridades globais. Para evitar inconsistência global, o GTG poderá monitorar constantemente o banco de dados global, porém isto poderia requerer um alto custo de execução do MSGBD.

No modelo de transação global livre proposto em [Lee e Park, 1998], uma transação local é proibida de atualizar dados globais. Esta restrição mantém a integridade global e reduz custos de verificação de integridade pelo MSGBD. É permitida à transação local somente ler dados locais e globais. Já a transação global é livre para ler e atualizar qualquer item de dado (local ou global). A tabela 6.1 mostra as operações possíveis ou não no modelo de transação global livre para MSGBD.

Transações		Dados	
		Dados Locais	Dados Globais
Transações Locais	Operação: Leitura	Sim	Sim
	Operação: Escrita	Sim	Não
Transações Globais	Operação: Leitura	Sim	Sim
	Operação: Escrita	Sim	Sim

Tabela 8.1: Modelo de transação global livre para MSGBD.

Um *escalonador* que possui apenas transações globais, pode não ser serializável, pois poderão existir conflitos de dados causados por transações locais (conflitos indiretos).

Para garantir a serialidade de um conjunto de transações globais, deve ser feito um *escalonador* que utilize um método de controle de concorrência de transações.

8.3.3 Conflitos diretos entre transações locais e globais

A tabela 8.2 mostra todos os conflitos diretos, conforme os tipos de operações: leitura ou escrita. A letra "o" significa que não pode existir conflito direto entre as duas operações correspondentes. A letra "x" representa a possibilidade de existência de conflito direto entre as operações.

Alguns conflitos diretos entre transações locais e globais podem gerar conflitos indiretos entre transações globais. Portanto, se evitarmos conflitos diretos estaremos evitando, também, os conflitos indiretos.

			Transação Global			
			Operação de Leitura		Operação de Escrita	
			Dado Global	Dado Local	Dado Global	Dado Local
Transação Local	Operação de Leitura	Dado Global	o	o	x	o
		Dado Local	o	o	o	x
	Operação de Escrita	Dado Local	o	x	o	x

Tabela 8.2: Conflito direto entre transação global e local
O= não ocorre conflito e x= possibilidade de conflito

8.3.4 Método otimista para controle de concorrência global

Para garantir a seriabilidade de transações globais, utiliza-se o controle da concorrência global otimista. A idéia básica deste método é simples: quando todas as sub-transações de uma transação global G forem completadas, se houver garantia de que G não está em conflito direto ou indireto com outras transações globais, e G é serializável, então G pode ser encerrada com um *commit*. É papel do GTG fazer uma verificação destes itens antes que a transação global seja aceita.

8.3.4.1 Gerenciamento de conflito indireto

Um gerenciador de transação global otimista valida se os conflitos diretos e indiretos de uma transação global são solucionáveis após o acesso a todos os *sites*. Analisando a tabela 8.3 percebemos que uma operação de leitura em um dado global por uma transação global não gera conflito direto com nenhuma transação local. Portanto, podemos saber que um conflito indireto existe sem nenhuma informação das transações locais.

		Transação Global Gi		
		Leitura		Escrita
Transação Global Gj		Dado Global	Dado Local	
Leitura	Dado Global	o	o	o
	Dado Local	o	x	x
Escrita		o	x	x

Tabela 8.3: Conflito indireto entre transações globais
O= não ocorre conflito indireto e x= possibilidade de conflito indireto

Como operações de leitura em dados globais não geram nenhum tipo de conflito entre transações globais, neste caso, fica garantido a seriabilidade global.

Baseando-se na tabela 8.3, define-se [Lee e Park, 1998] o conceito de uma *transação global estável*. Considerando que $Site_{Gi}(LL)$ é o conjunto de *sites* que uma transação global G_i lê dados locais, e $Site_{Gi}(E)$ é o conjunto de *sites* que uma

transação global G_i escreve, uma transação global G_i é considerada uma *transação global estável*, se e somente se, para toda outra transação G_j, que compartilha o *site* com G_i:

$$Site_{Gi}(LL) \cap \{ Site_{Gj}(LL) \cup Site_{Gj}(E) \} = \varnothing \text{ e}$$
$$Site_{Gi}(E) \cap \{ Site_{Gj}(LL) \cup Site_{Gj}(E) \} = \varnothing$$

A partir desta definição, Lee e Park escreveram o seguinte teorema: *"Uma transação global G_i estável não pode ter nenhum conflito indireto com outra transação global"* [Lee e Park, 1998].

8.3.4.2 Gerenciamento de conflito direto

Para serializar operações com conflitos diretos, o GTG mantém um grafo chamado de Grafo de Serialização Global (GSG) composto por um conjunto de nós V e um conjunto de arestas E que representa a ordem serial da transação global.

O conjunto V (nós) contém as transações globais que já foram concluídas (*commited*) ou não. Uma aresta E (G_i, G_j) refere-se ao fato de que uma operação $p_{ix}(a)$ de uma transação global G_i precede a operação $q_{jx}(a)$ da transação global G_j no *site* S_x e a operação p está em conflito direto com a operação q.

Quando o escalonador otimista recebe uma operação $p_{ix}(a)$ de uma transação global G_i do GTG, ele cria um nó para G_i no GSG, se ainda não existir. Em seguida, ele adiciona uma aresta de G_j para G_i para toda operação $q_{jx}(a)$ que conflita diretamente com $p_{ix}(a)$. Após todas as operações $q_{jx}(a)$ terem sido completadas com sucesso, a operação $p_{ix}(a)$ é enviada para o *site* correspondente S_x.

Para obrigar que operações com conflito direto mantenham a mesma ordem serial em seus *sites* locais correspondentes, o GTG deve atrasar $p_{ix}(a)$ até ele reconhecer todas as operações com conflito direto. Se todas sub-transações de uma transação global G_i estão prontas para serem concluídas (commited), o GTG verifica se o GSG contém ciclos. Se o GSG tiver ciclo, o escalonador resultante pode ser um

escalonador não-seriável. Conseqüentemente, G_i é abortada e deve ser reiniciada. Caso contrário, G_i não possui conflito direto e a transação G_i é aceita.

Resumidamente, de acordo com o modelo para controle de concorrência otimista proposto em [Lee e Park, 1998] descrito nos itens 8.3.4.1 e 8.3.4.2, uma transação global pode ser confirmada (*commited*) se o GSG não tiver ciclos, e a transação global for considerada como uma transação global estável. A figura 8.5 ilustra uma transação global G_i estável.

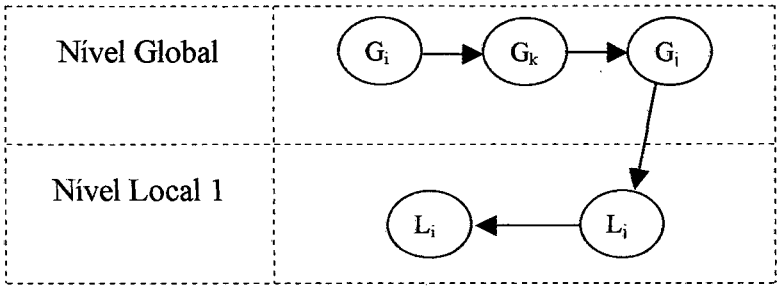

Figura 8.5: Transação Global estável (Gi)

8.4 Método *Tickets* para controle de concorrência global

O método de ticket (MT) foi proposto por [Georgakopoulos et al.,1994]. É um algoritmo amplamente utilizado para o gerenciamento de transações em sistemas multidatabase. Este método utiliza um *ticket* para determinar a ordem serial das sub-transações nos bancos de dados locais.

Um ticket é considerado um marcador de tempo (*timestamp*) cujo valor é armazenado como um item de dado em cada base de local. Os *tickets* são processados por uma operação chamada *take-a-ticket* que lê o valor de um ticket, incrementa-o e escreve. Cada sub-transação de uma transação global é obrigada a executar a operação *take-a-ticket* quando acessar o *site*. Apenas as sub-transações de transações globais têm que pegar o ticket, transações de origem local não são afetadas. O valor do *ticket* é lido pela sub-transação e mais tarde é informado para o MSGBD como um *timestamp* lógico de cada *site*. A validação é executada usando o Grafo de Serialização Global (GSG).

Uma aresta (i,j) do GSG significa que uma operação da sub-transação global Gi é precedida por uma sub-transação global Gj em pelo menos um site. Em outras palavras, se o valor do ticket de Gi é menor que Gj, é inserida uma aresta de Gi para Gj no GSG. O MSGBD valida a transação global verificando a existência de ciclos no grafo. Se houver ciclos a transação não passa no teste de validação.

Se duas sub-transações em um mesmo *site* acessarem dados diferentes, elas serão serializáveis devido ao valor do ticket.

Conforme cresce o número de transações globais, o número de transações globais que são reiniciadas (devido a conflitos) aumenta.

8.5 Avaliação comparativa entre métodos para controle de concorrência

Para avaliar a performance, em [Lee e Park, 1998] compara-se o controle de concorrência otimista proposto nos itens 8.3.4.1 e 8.3.4.2 com o método de *tickets*. Os principais parâmetros utilizados para avaliação e valores são descritos na tabela 8.4. Analisou-se a quantidade de transações globais reiniciadas nos dois referidos modelos.

Parâmetro	Valores
Número máximo de transações globais	[100, 900]
Número de *sites* locais	[10,70]
Tamanho do banco de dados local	30 tabelas
Tempo de operação	30 ms
Operações de leitura	[0,100] %
Operações de leitura em dados globais	[0,100] %

Tabela 8.4: Parâmetros utilizados para avaliação de modelos

Os resultados obtidos são mostrados na figura 8.6. O eixo x mostra o número de transações globais e o eixo y o número de transações globais reiniciadas.

Nota-se que o número de transações globais reiniciadas aumenta conforme cresce o número de transações globais enviadas.

Na figura 8.7, é visível que o número de transações globais reiniciadas diminui quando o número de *sites* locais aumentam.

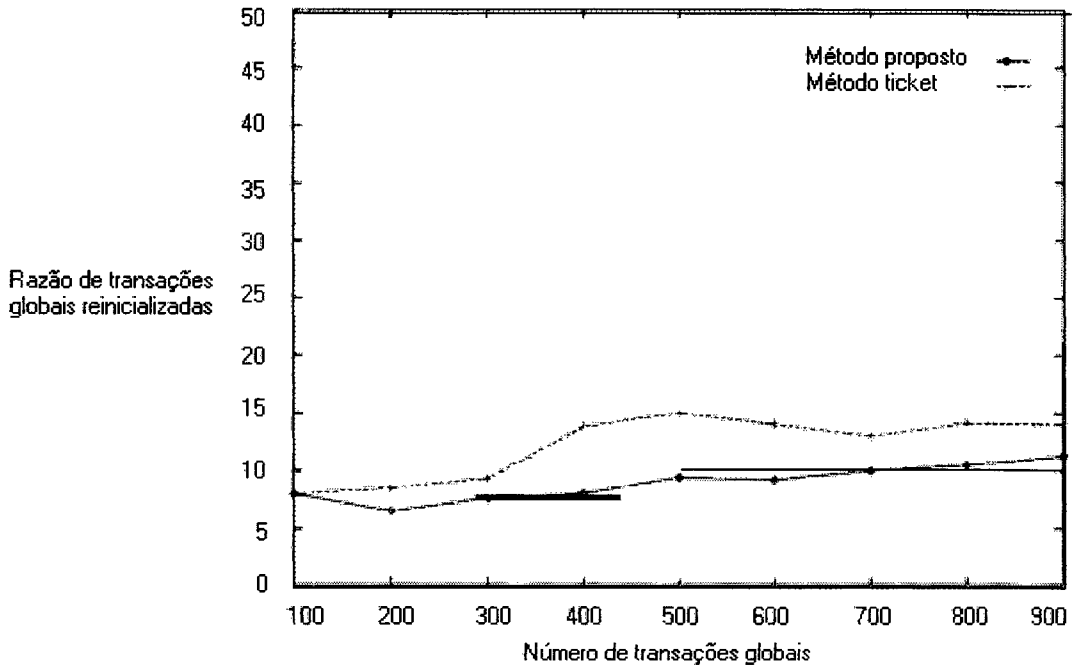

Figura 8.6: Transações globais reinicializadas versus número de transações globais

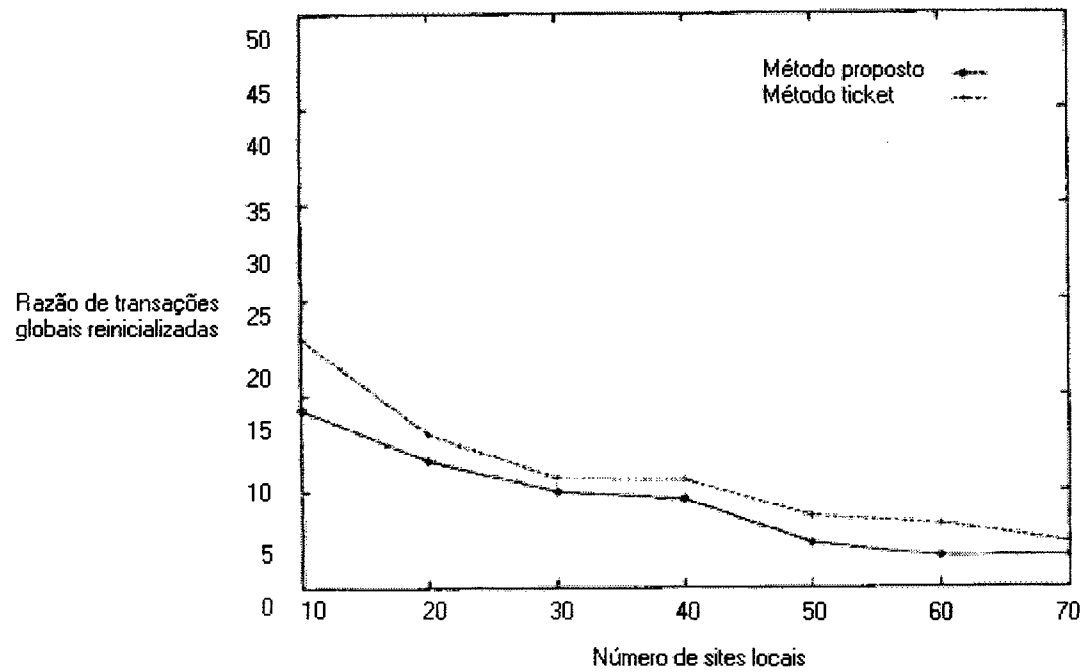

Figura 8.7: Transações globais reinicializadas versus número de sites locais

Os resultados obtidos nesta pesquisa indicam que o método proposto [Lee e Park, 1998] possui um menor índice de reiniciação de transações globais, aumentando a performance e garantindo as regras de integridade locais e globais. Portanto, uma transação global tem mais chances de ser aceita (*commited*) neste método.

8.6 Comentários finais

Neste capítulo foram apresentados aspectos relacionados ao controle da concorrência em sistemas de banco de dados distribuídos heterogêneos. Conforme exposto, existem várias propostas para a execução concorrente de transações nestes ambientes, podendo ser agrupadas em métodos otimistas e pessimistas.

Apresentou-se, entre outros, o gerenciamento de transação considerando restrições de integridade de itens de dados, proposto por Lee e Park, e por fim uma avaliação comparativa entre métodos para controle de concorrência.

O capítulo seguinte, refere-se a interoperabilidade entre sistemas de banco de dados heterogêneos.

Capítulo 9

INTEROPERABILIDADE

Para criação de sistemas de informação integrados e distribuídos, a primeira fase consiste em prover interconectividade entre os sistemas, isto é, capacidade de trocar mensagens [Nicol et al., 1993]. Entretanto a interconectividade garante somente a comunicação, não a cooperação.

A segunda fase [Nicol et al., 1993] consiste em prover interoperabilidade entre sistemas, isto é, a capacidade dos recursos de computação interagirem para executarem tarefas conjuntamente. O grau de interoperabilidade pode variar desde uma simples capacidade de programação até formas mais avançadas que requeiram compreensão mútua, não somente a nível de sistemas ou programas, mas também a nível de semântica dos dados, como é o caso de Sistemas de Banco de Dados Heterogêneos.

Para prover a interoperabilidade entre bancos de dados heterogêneos existem problemas complexos. A questão da interoperabilidade de informação envolve não apenas problemas relacionados com o meio físico, responsável por realizar a troca de informação entre os sistemas, mas principalmente problemas relacionados com a informação propriamente dita.

Estes problemas existem devido a diferenças de semântica, localização dos dados e plataformas (hardware, sistemas operacionais ou protocolos de rede) entre os bancos de dados componentes. Isso significa que, para poderem interagir, os sistemas precisam atender a um padrão mínimo de funcionalidade e estrutura da informação.

Mediante esses fatos, é necessário um modelo capaz de prover interoperabilidade entre sistemas de banco de dados heterogêneos. Um modelo que muito tem contribuído para interoperabilidade de sistemas heterogêneos em geral é o orientado a objeto. A seguir, descrevemos a utilização da orientação a objetos na interoperabilidade entre bancos de dados distribuídos e heterogêneos.

Neste capítulo, introduziremos sucintamente alguns conceitos sobre o modelo orientado a objetos dada a sua forte importância para prover interoperabilidade entre sistemas heterogêneos.

9.1 Orientação a objetos e interoperabilidade

A orientação a objetos está presente em praticamente todas as áreas da computação: nos métodos de análise e projeto, nos bancos de dados (bancos de dados orientados a objetos), no desenvolvimento para Internet e até mesmo no *kernel* de alguns sistemas operacionais.

Para lidar com interoperabilidade entre bancos de dados, a orientação a objetos é uma idéia nova e talvez a mais pesquisada atualmente. Conforme descrito no capítulo 3, o esquema conceitual de cada banco de dados local é traduzido para um esquema intermediário padrão. Este esquema padrão corresponde a cada esquema local traduzido para um modelo de dados de uso comum no SGBDH. O modelo orientado a objetos é visto como o modelo padrão mais apropriado.

O uso de padrões para a orientação a objetos é uma estratégia utilizada para coordenar e integrar os recursos de processamento de informações, interligados por uma rede de comunicação. Na arquitetura dos sistemas de banco de dados heterogêneos, estes padrões podem ser utilizados para a integração dos componentes heterogêneos da federação, aumentando sua flexibilidade para adaptar-se a novos ambientes e sua capacidade de expansão [Uchôa et al., 1996].

Para prover interoperabilidade, fatores como escala, heterogeneidade, gerenciamento de configuração e monitoramento de redes são críticos. A tecnologia de objetos facilita o gerenciamento desta complexidade [Soley e Kent, 1995], já que:

- fornece um meio comum onde estruturas de dados complexas, bem como especificações de comportamento, podem ser representadas e,
- acomoda, naturalmente, heterogeneidade e autonomia: heterogeneidade porque mensagens enviadas para os componentes distribuídos dependem apenas das interfaces dos componentes, indiferentes a suas características internas; e autonomia porque componentes podem mudar independentemente e transparentemente, desde que mantenham suas interfaces inalteradas.

Existem duas maneiras nas quais a orientação a objetos pode facilitar a interoperabilidade em bancos de dados: na integração de esquemas e no gerenciamento de objetos distribuídos. A seguir é feita uma revisão sobre os termos relacionados ao modelo orientado a objetos para melhor entendimento dos aspectos a serem considerados nas seções seguintes.

9.2 Sistemas de gerenciamento de banco de dados orientado a objetos

O desenvolvimento dos Sistemas de Gerenciamento de Banco de Dados Orientado a Objetos (SGBDOO) teve origem na combinação de idéias dos modelos de dados tradicionais e de linguagens de programação orientada a objetos. Nestes sistemas a noção de objeto é usada no nível lógico e possui características não encontradas nas linguagens de programação tradicionais, como operadores de manipulação de estruturas, gerenciamento de armazenamento, tratamento de integridade, etc.

A modelagem de dados orientada a objetos tem um papel importante nos SGBDs por ser um modelo mais adequado para o tratamento de objetos complexos (textos, gráficos, imagens) e dinâmicos (programas, simulações). A principal característica de SGBDOO é modelar estruturas complexas armazenando não somente a estrutura de dados, mas também seu comportamento. SGBDOO combinam conceitos a objeto com capacidade de bancos de dados e, portanto, têm o

potencial de fornecer poderosos repositórios para aplicações avançadas de bancos de dados. Os Sistemas de Gerenciamento de Banco de Dados Orientados a Objetos foram uma das grandes idéias do início dos anos 80. Contudo, até hoje está em desenvolvimento e ainda não é um modelo definido, por isso não são tão utilizados pelas empresas. No entanto, o surgimento cada vez maior de bancos de dados não convencionais para aplicações específicas que tratam dados mais complexos aumenta o valor e o interesse para a tecnologia orientada a objeto.

9.2.1 Objetos

Em um modelo orientado a objetos, toda entidade é considerada um objeto. Um objeto poderia ser uma federação, um componente, um banco de dados, um esquema, ou mesmo um SGBDs.

9.2.2 Classe

Uma coleção de objetos que possuem propriedades similares é agrupado em classes. Uma hierarquia de classes pode ser formada. Um sub-conjunto de objetos de uma classe pode ter alguma propriedade adicional em comum. Então, esses objetos formam uma nova classe que é uma subclasse da classe original. As sub-classes herdam as propriedades e métodos das super-classes.

9.2.3 Métodos

As propriedades de uma classe são especificadas por instâncias variáveis. Objetos possuem métodos, que são basicamente *procedures* associados a eles. O estado de um objeto apenas pode ser acessado através dos métodos (*procedures*) que são invocados por meio de mensagens.

9.2.4 Encapsulamento (Empacotamento)

Encapsulamento, ou empacotamento, é uma propriedade do modelo de objetos que permite a tradução de esquemas de um modelo qualquer para um orientado a

objeto, ocultando as diferenças de interfaces e implementações dos SGBDs componentes. Nos capítulos 3 e 4 se referiu a esta propriedade como esquema intermediário padrão. Um outro termo que tem sido proposto é *wrapper*. Os *wrappers* além dos esquemas, encapsulam as rotinas que traduzem a linguagem de consulta do usuário global para as linguagens dos SGBDs componentes. A partir daí, tudo se comporta como uma *caixa preta* (objeto), que recebe mensagens e parâmetros e devolve resultados.

A figura 9.1 ilustra o encapsulamento de dois SGBDs componentes. Um BD utiliza a abordagem em redes e o outro a relacional. A consulta global é feita em uma linguagem global, no caso do exemplo, foi utilizado SQL. Em cada site a seleção é feita de acordo com a linguagem do SGBD local. Deste modo, é dispensável que o usuário conheça as linguagens dos bancos de dados componentes.

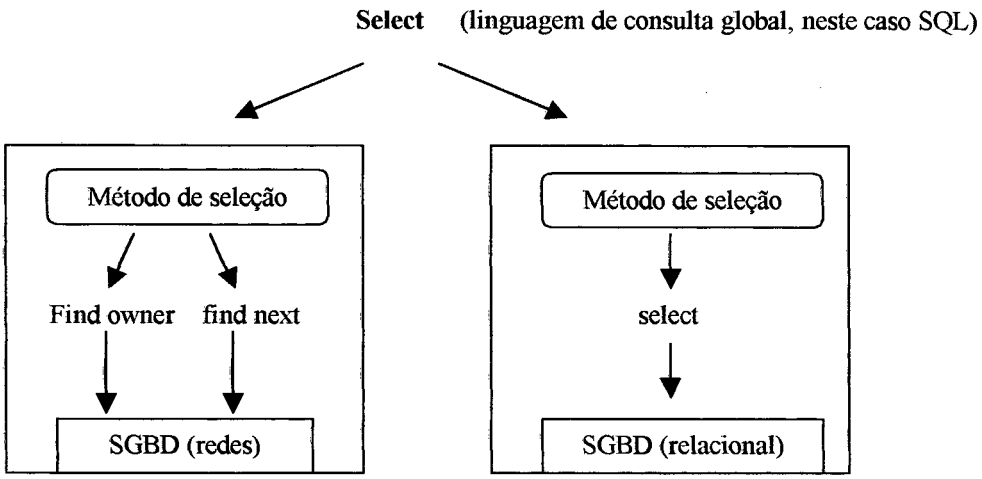

Figura 9.1: Encapsulamento de SGBDs heterogêneos

9.2.5 Especialização/Generalização

Uma outra propriedade do modelo orientado a objetos que é útil para gerenciar a interoperabilidade de banco de dados é a especialização/generalização. Esta propriedade permite a criação de tipos que abstraem a similaridade das entidades armazenadas em bancos de dados diferentes [Özsu e Valduriez, 1999].

Para exemplificar, considere duas entidades de nome *Empregado* definidas de forma diferente em dois bancos de dados componentes. No primeiro banco de dados a entidade Empregado possui os atributos *emp_num* e *salário*, e no segundo os atributos *emp_num* e *endereço*.

No esquema integrado (orientado a objetos), estas duas entidades poderiam ser modeladas como um subtipo de uma entidade mais geral Empregado que abstrai o que é comum entre estas entidades (figura 9.2).

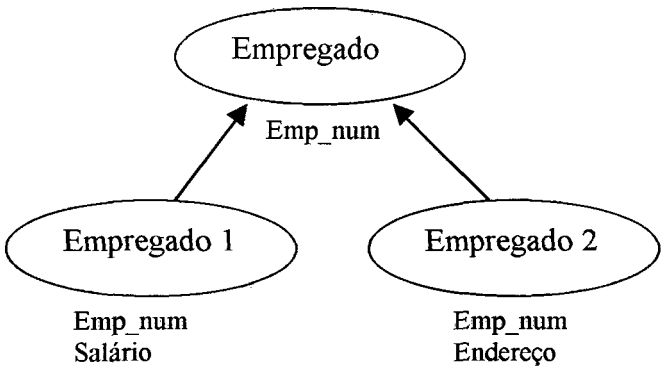

Figura 9.2: Abstração de entidades heterogêneas

Além da orientação a objetos, há várias plataformas para computação distribuída cujo propósito principal é facilitar o desenvolvimento de sistemas abertos que permitem aplicações diferentes se comunicarem entre si.

Existem vários projetos que testam estas plataformas, porém ainda é desconhecida publicação de resultados baseados em uma experimentação científica destes sistemas.

9.3 Esquema padrão – modelo orientado a objetos

A maioria dos sistemas de bancos de dados participantes da federação especificam tipo de modelagem de dados conforme a aplicação. Usuários de um sistema heterogêneo devem aprender as várias representações de dados, se desejarem

acessar sistemas remotos. Obviamente que essa solução não é nem um pouco satisfatória.

Considerando que em um *site* 1 temos uma modelagem relacional e em um *site* 2 o modelo de redes. Existem dois modos de interligar estes *sites*. O primeiro método propõe que o MSGBD automatize a transformação entre modelos através de tradutores, sem a necessidade de integração de esquemas. A desvantagem deste modelo é que quanto maior a heterogeneidade maior será o número de tradutores. Um segundo método é utilizar um esquema de representação intermediário (descrito com mais detalhes no capítulo 3) e integrar estes esquemas intermediários em um esquema global padrão.

Apesar da modelagem relacional dominar o mercado, o esquema padrão orientado a objetos proporciona várias vantagens. A figura 9.3 ilustra a utilização do modelo orientado a objetos como padrão.

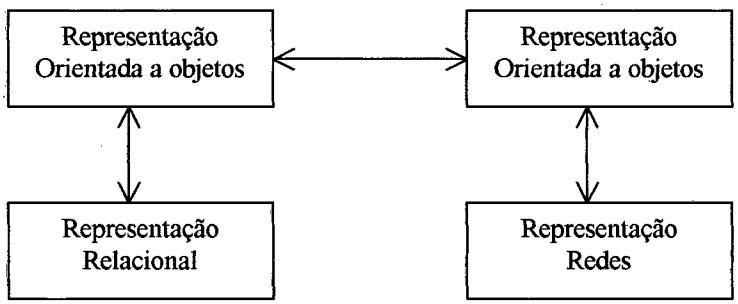

Figura 9.3: Esquema Global orientado a objetos

9.4 Gerenciamento de objetos distribuídos

Nesta seção é mostrado como um modelo orientado a objetos poderia ser usado para representar o sistema de banco de dados heterogêneo como um todo. O modelo de execução de um ambiente distribuído heterogêneo é essencialmente um sistema de gerenciamento de objetos distribuídos (SGOD). Esses sistemas estão se tornando cada vez mais populares pela interoperabilidade entre ferramentas e recursos. O SGBD, bem como o banco de dados é encapsulado como um objeto.

Padrões como CORBA (*common object request broker archicture*) estão sendo propostos para gerenciamento de objetos distribuídos. Maiores detalhes sobre CORBA é descrito nas seções seguintes.

A figura 9.4 representa parcialmente um ambiente. As classes são nomeadas como *federation* cujas instâncias (ocorrências) são as várias federações e *node* que contém os nós como instâncias.

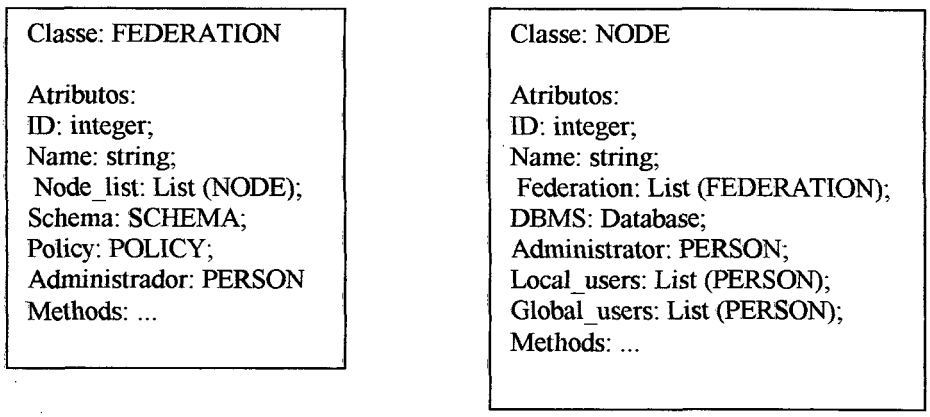

Figura 9.4: Exemplo (parcial) de um ambiente

Cada federação tem as seguintes variáveis de instâncias (atributos): a identificação da federação, o nome, a lista de nós que formam a federação, o esquema, a política administrativa da federação e o administrador ou grupo de administradores.

Para cada *node* (nó) definem-se as seguintes variáveis de instâncias (atributos): a identificação do nó, o nome, o conjunto de federações às quais o nó pertence, o sistema gerenciador do banco de dados (que inclui a política administrativa local, o esquema local, o SGBD e a base de dados), o administrador, os usuários locais e os usuários globais. Administradores e usuários são instâncias da classe PERSON.

Além das variáveis de instâncias mostradas na figura, cada nó terá o sistema de banco de dados como uma instância de variável que será um objeto e representa os sistemas de banco de dados locais.

Para explicar o gerenciamento, utilizar-se-á um exemplo ilustrado na figura 9.5. Supõe-se que o nó N1 quer se unir a federação F1. O nó N1 envia uma mensagem

para a classe FEDERATION com federação F1 como parâmetro da mensagem (msg 1). Quando a federação F1 recebe a mensagem, o método correspondente é executado. A política da federação deve ser examinada para ver se N1 pode se juntar a federação (msg 2), tarefa realizada pelo administrador U1 que mantém a política da federação.

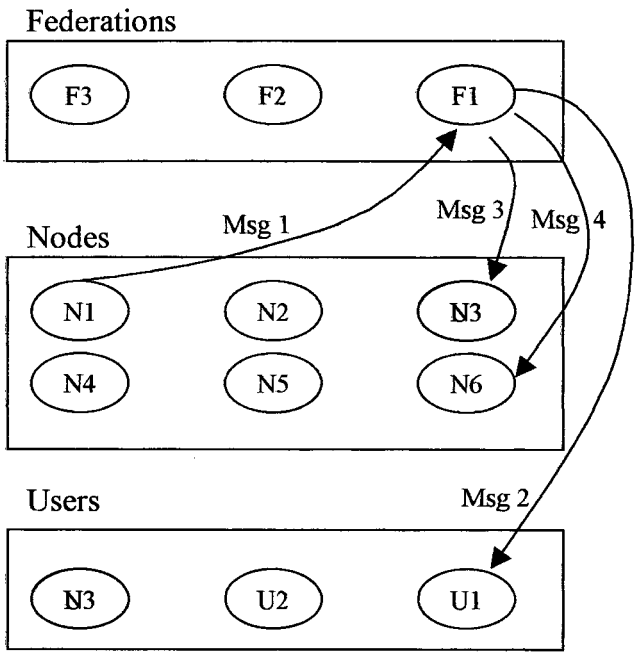

Figura 9.5: Um cenário

Ao mesmo tempo, a federação F1 poderá enviar mensagens também para os nós que já fazem parte da federação para verificar se estes nós desejam que N1 se una à federação (msg3 e msg4). Se todas as restrições forem satisfeitas, F1 envia uma mensagem para o nó N1 aceitando seu pedido de união e o inclui como parte da lista de nós de F1. Os valores das variáveis de instância node_list de F1 e federation_list de N1 são atualizados.

Quanto a operações executadas sobre sistemas distribuídos heterogêneos, uma visão geral é ilustrada na figura 9.6. Primeiro, um objeto de interface (usuário local) recebe um pedido de um outro objeto de interface (outro usuário local ou remoto) denominado objeto cliente. Se o pedido é originário de um usuário local, o objeto de

interface o envia diretamente para o objeto SGBD local. Caso o pedido seja de um usuário remoto, um objeto mediador adequado ao tipo de pedido (select, update, etc) é então solicitado.

No caso de um pedido de consulta, o objeto mediador de consulta irá aceitar o pedido, executar algumas operações como análise gramatical (*parsing*) e enviar o pedido para o objeto processador de consultas.

O objeto processador de consultas irá interagir com o objeto gerenciador de esquemas, o objeto gerenciador de segurança e com o objeto gerenciador de integridade, executar algumas transformações de esquemas, realizar verificações de segurança e integridade, otimizar a consulta e subseqüentemente enviar a consulta (ou sub-consultas) para o objeto SGBD local. O objeto SGBD local irá então executar a consulta.

Finalmente, a resposta é então enviada para o usuário através do objeto processador de consulta e objeto interface.

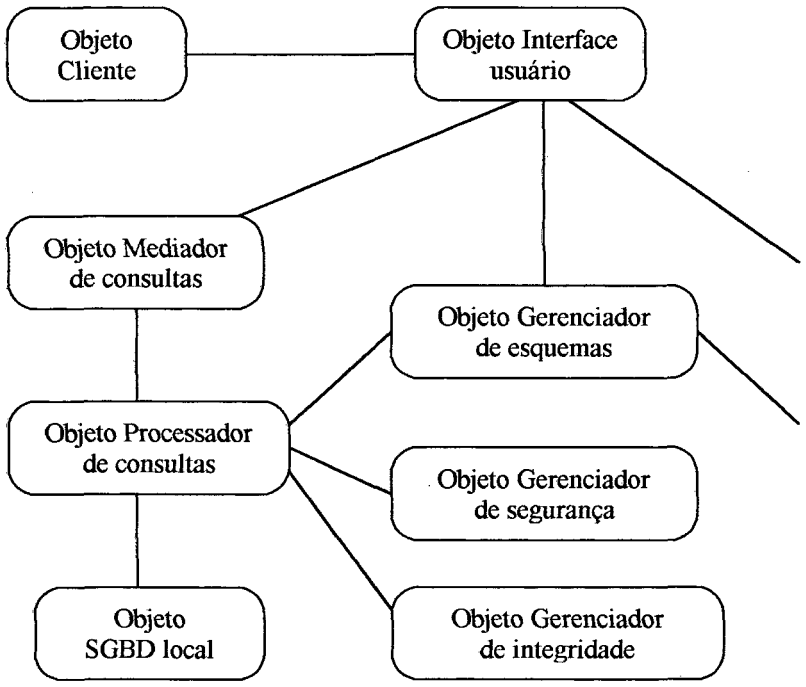

Figura 9.6: Um ambiente de gerenciamento de objetos distribuídos

9.5 Padrão do OMG (Object Management Group)

A diversidade em modelos de objetos é uma das razões que levaram várias empresas especializadas em sistemas orientados a objetos a cooperar, no sentido de definir um mínimo denominador comum, a ser seguido pelos ambientes orientados a objetos disponíveis comercialmente.

OMG – Object Management Group, é uma associação comercial internacional, com fins não-lucrativos e envolve atualmente mais de 800 empresas. Tem como objetivo básico desenvolver um padrão para interoperação de sistemas heterogêneos distribuídos (principalmente orientados a objetos), enfocando o recebimento e emissão de mensagens [Helal e Elmasri, 1995]. A missão do OMG consiste em [Martin, 1994]: maximizar a portabilidade, isto é, a capacidade de reaproveitamento e a interoperabilidade de software; prover uma arquitetura referencial com termos e definições sobre os quais se baseiam todas as especificações.

A OMG cria padrões industriais para os sistemas baseados em objetos disponíveis comercialmente, focalizando o acesso remoto a objetos, via redes, no encapsulamento dos aplicativos existentes e nas interfaces dos bancos de dados baseados em objetos.

O grande diferencial da abordagem dos padrões do OMG é que tais padrões são e serão baseados, tanto quanto possível, em produtos existentes disponíveis comercialmente. Ao contrário da Opens Software Foundation (OSF) da DEC (*Distributed Computing Environment*), OMG não produz softwares, mas somente especificações [Vinoski, 1993]. A OMG firmou importantes alianças para assegurar-se de que seus padrões sejam universais.

9.6 OMA - Object Management Archicture

O *Object Management Archicture* (OMA) é um ambiente definido pela OMG, que define uma arquitetura de gerenciamento de objetos. O objetivo desta arquitetura é o de capacitar interoperabilidade entre softwares de fornecedores diferentes.

A arquitetura do OMA (figura 9.7) consiste em quatro partes principais: Aplicativos-Objetos (*Application Objects*), Recursos comuns (*Common Facilities*), Serviços de Objeto (*Common Object Services* - COSS) e Intermediador de Solicitações de Objetos (*Object Request Broker* – ORB).

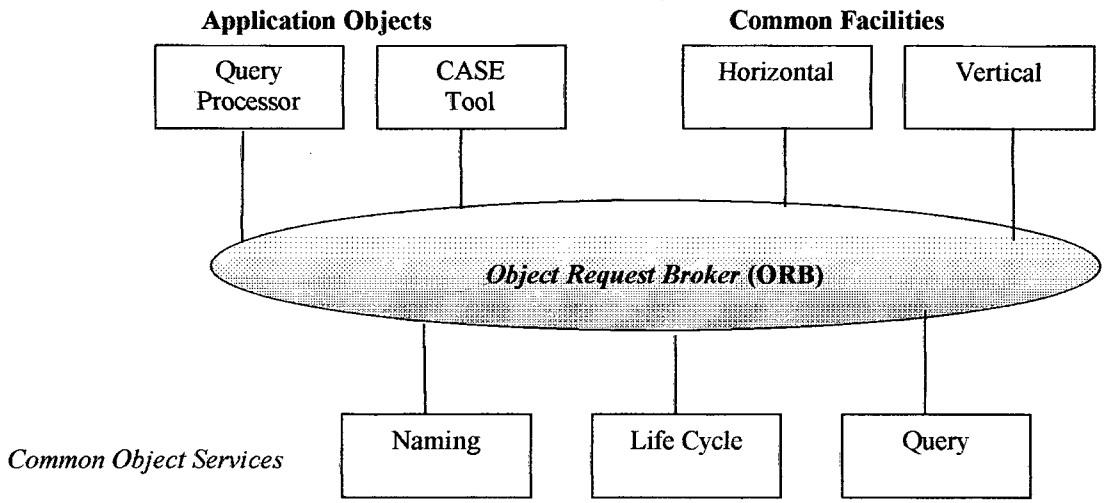

Figura 9.7: Arquitetura de Gerenciamento de Objetos (OMA) do OMG

Aplicativos-Objetos (*Application Objects*) são aplicativos para uso final que podem ser construídos por diversos fabricantes ou pelos departamentos de informática das empresas.

Recursos Comuns (*Common Facilities*) são objetos e classes que provêem capacidades de propósito geral úteis a vários aplicativos [OMG, 1993]. É uma biblioteca compartilhada de funções comumente utilizadas. Existem duas classes de recursos comuns: horizontal e vertical. A primeira consiste nos recursos que são usados por todas (ou muitas) aplicações objetos. Exemplo destes recursos inclui interface com o usuário, sistemas gerenciadores e gerenciamento de tarefas. Aplicações verticais, por outro lado, são componentes especializados para aplicações de domínio selecionado como saúde, finanças, comércio eletrônico e telecomunicações [Özsu e Valduriez, 1999].

Serviços de Objeto (*Common Object Services*) são coleções de serviços que provêem as funções básicas para criar, armazenar, manter, estabelecer subtipos e

gerenciar objetos. Eles incluem serviços de gerenciamento do ciclo de vida, gerenciador de transação, controle de concorrência, dentre outros [Uchôa et al., 1996].

Intermediador de Solicitações de Objetos (*Object Request Broker* – ORB) é o núcleo, o coração do OMA. Ele fornece transparência de localização, ativação e comunicação de objetos, permitindo que os objetos se comuniquem independentemente de plataformas e técnicas específicas. Um objeto cliente faz uma solicitação de maneira padrão e o ORB é responsável por providenciar um objeto que implementa a solicitação requerida, passar os parâmetros ao objeto invocado e retorna os resultados ao objeto cliente solicitador.

9.7 CORBA (*Common Object Request Broker Archicture*) e Interoperarabilidade de Bancos de Dados

Como uma forma de tratar e simplificar a computação em sistemas distribuídos heterogêneos, o OMG (*Object Management Group*) propôs o padrão CORBA (*Common Object Request Broker Architecture*).

CORBA vem sendo utilizado em inúmeros projetos para integrar sistemas de banco de dados. É uma especificação e uma arquitetura que possibilita que objetos em sistemas heterogêneos possam ser acessados de forma independente de sua implementação.

No CORBA, clientes solicitam a execução de serviços e servidores executam tais serviços, tudo em termo de tarefas denominadas operações que são executadas por entidades denominadas de objetos. As aplicações interagem umas com as outras sem saber onde as outras aplicações estão na rede ou como elas executam suas tarefas.

Usando CORBA é possível encapsular aplicações e suas respectivas operações como conjuntos de objetos distribuídos, de tal forma que se possa plugar e desplugar capacidades de clientes e servidores à medida que estes necessitem ser adicionados ou substituídos em um sistema distribuído. CORBA trata heterogeneidade a nível de

plataforma e a nível de banco de dados, oferecendo infra-estrutura para implementar SGBDH, deixando, no entanto, a nível da programação do SGBDH o tratamento da interoperabilidade semântica.

9.8 Comentários Finais

Um modelo muito utilizado para prover interoperabilidade de sistemas heterogêneos em geral é o orientado a objeto

Neste capítulo mostrou-se como um modelo orientado a objetos pode ser utilizado para representar o sistema de banco de dados heterogêneo como um todo. Para isso a OMG definiu uma arquitetura de gerenciamento de objetos com objetivo de capacitar interoperabilidade entre softwares de fornecedores diferentes.

Existe uma grande lista de projetos de sistemas gerenciadores desenvolvidos pela indústria e universidades. No capítulo seguinte é feita uma apresentação de alguns dos projetos existentes, detalhando alguns deles.

Capítulo 10

SISTEMAS GERENCIADORES DE BANCOS DE DADOS HETEROGÊNEOS

Vários sistemas gerenciadores de bancos de dados heterogêneos vêm sendo desenvolvidos nos últimos anos. Existem vários exemplos de sistemas multidatabase comerciais como Sybase, Oracle, Ingres/Star e UniSQL/M e também de projetos que estão sendo desenvolvidos [Elmagarmid et al. 1999]. Alguns dos sistemas existentes suportam totalmente a funcionabilidade de um *multidatabase*, outros, apenas funções específicas.

Sysbase's Open Client and Open Server [Elmagarmid et al. 1999] suporta bancos de dados heterogêneos. Não trabalha com esquema global e sim com duas linguagens para acesso a multidatabase: Transac-SQL e Visual Query Language (VQL). Através destas linguagens o usuário pode definir visões sobre o multidatabase, fazer consultas globais, etc. Oracle também possui uma linguagem que suporta consultas entre sistemas *multidatabase* chamada SQL*PLUS.

Ingres/Star [Elmagarmid et al. 1999] é um software para acesso transparente a sistemas distribuídos Ingres. Permite a definição de visões externas globais sobre o sistema de banco de dados. Diferente do Sybase, este sistema não suporta dependências entre banco de dados. Não é possível que usuários executem consultas globais diretamente aos bancos de dados. Um consulta global é formulada sobre um multidatabase externo cujos elementos são as tabelas referenciadas na consulta.

UniSQL/M [Elmagarmid et al. 1999] é um outro sistema multidatabase comercial cuja linguagem de consulta SQL/M é uma extensão da ANSI SQL, porém

com modelagem orientada a objetos. Este sistema usa classes virtuais globais para integrar entidades heterogêneas de diferentes esquemas.

Ainda que com muitas restrições e dificuldades para prover interoperabilidade entre sistemas heterogêneos, existe uma grande lista de sistemas comerciais existentes. Também existem muitos projetos sendo desenvolvidos. Nas seções seguintes serão apresentados alguns desses projetos com suas principais características.

10.1 *Multidatabase* [Buretta, 1997]

Multidatabase é um software desenvolvido pela *Computer Corporation of America* para recuperação de dados de bancos de dados heterogêneos. Este sistema provê ao usuário uma visão uniforme do banco de dados e uma linguagem de manipulação de dados chamada Daplex.

Multidatabase fornece um alto nível de interface para aplicações somente para leitura mantendo transparência da localização e heterogeneidade dos SGBDs participantes. O principal objetivo deste sistema gerenciador é prover uma interface para sistemas preexistentes sem modificar seus softwares.

A arquitetura do *multidatabase* (figura 10.1) é composta por dois principais componentes: o gerenciador de dados global (GDG) que é responsável pelas consultas globais, e o componente de interface de banco de dados local (IDL), responsável pela interface dos SGBDs nos vários *sites*.

O esquema global oferece um visão integrada do banco de dados distribuído e é acessado pela linguagem Daplex. Cada site possui o componente de interface local que é um intermediário entre o esquema local do site (acessado pela linguagem do BDs locais) e um esquema intermediário local (acessado em Daplex).

O gerenciamento de consultas é executado da seguinte maneira. Quando uma consulta global é executada, o GDG decompõe a consulta em várias sub-consultas Daplex sobre os esquemas locais e o banco de dados auxiliar. O esquema auxiliar

descreve dados necessários para o mapeamento entre os esquemas. Também é de responsabilidade do GDG montar os resultados parciais e retorná-lo ao usuário.

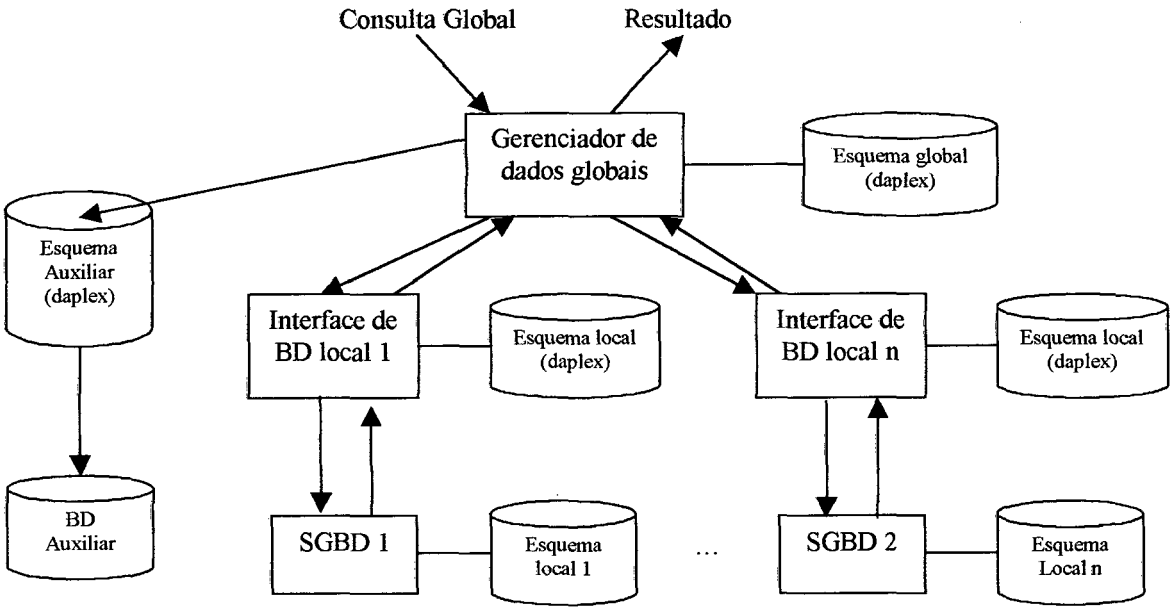

Figura 10.1: Arquitetura do SGBDD-H Multidatabase

O IDL (interface de banco de dados local) é responsável por aspectos relacionados especificamente aos bancos de dados locais. Ele traduz as consultas escritas nas linguagens dos BDs locais para Daplex e vice-versa.

O GDG é composto por cinco módulos (figura 10.2): transformador, otimizador global, decompositor e monitor enquanto o IDL é composto por três: interface de rede, otimizador local, e tradutor.

O transformador pega a consulta global em Daplex como entrada e produz como saída uma consulta global Daplex que referencia o esquema local. O otimizador global utiliza a saída do transformador para gerar um plano de consulta através do algoritmo de otimização SDD-1.

O decompositor decompõe a consulta em sub-consultas e o filtro elimina de cada sub-consulta aquelas operações que não são suportadas pelo SGBD

correspondente. Estas operações removidas serão executadas à parte. O monitor controla a execução da consulta.

A interface de rede em cada *site* é o módulo responsável por transmitir as consultas e seus resultados. O tradutor finalmente traduz a consulta para a linguagem do banco de dados local.

O *multidatabase* tem como restrição o fato de não fazer um controle ideal da consistência dos bancos de dados. Porém, várias soluções estão sendo propostas para lidar com inconsistências. A mais utilizada é agregar funções próprias para controle da inconsistência de dados

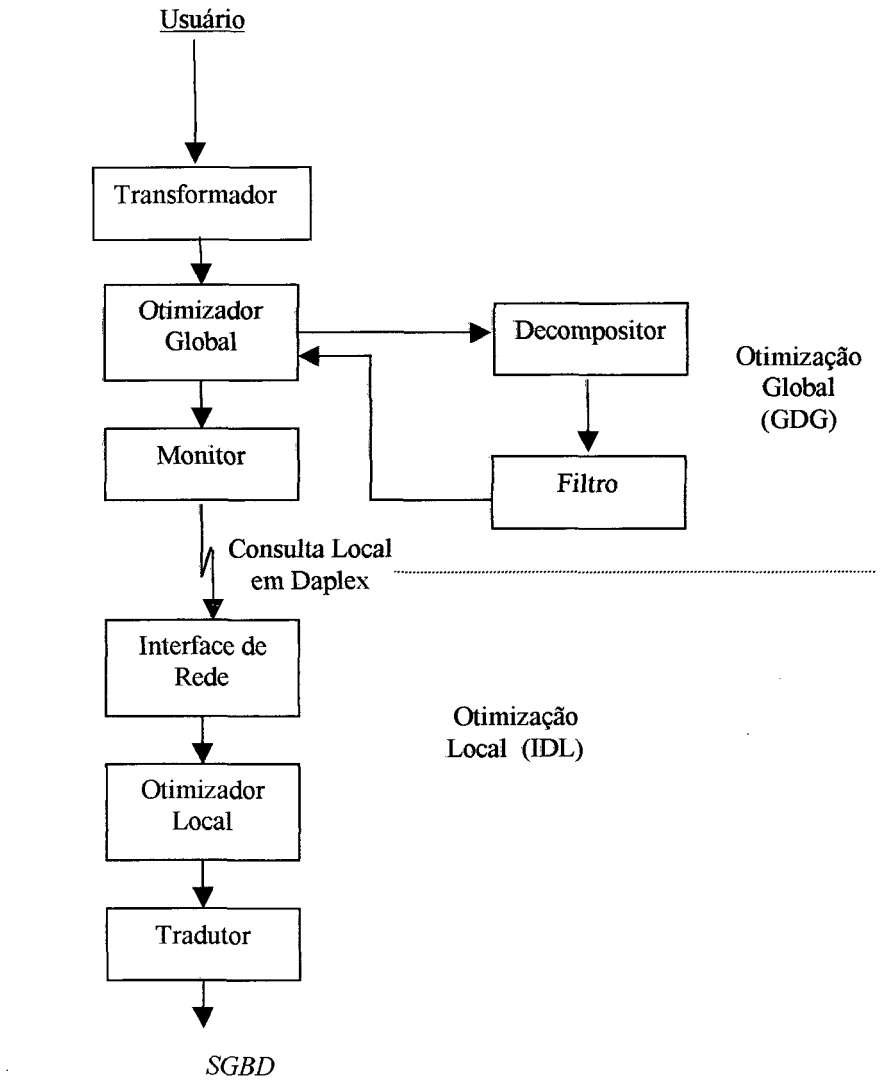

Figura 10.2: Otimização e tradução de consultas no Multidatabase

10.2 Projeto MIND [Dogac et al., 1995]

O projeto MIND (Metu Interoperable Database System) é um sistema de múltiplos bancos de dados cujo principal objetivo é desenvolver uma arquitetura e ferramentas que propiciem interoperabilidade entre sistemas de banco de dados heterogêneos existentes, relacionais e orientado a objetos. Parcialmente financiado por Motorola Inc. – USA e Sevgi Foundation – Turkey, este projeto pretende oferecer aos usuários dos múltiplos bancos de dados uma visão única e integrada.

Sua arquitetura (figura 10.3) é baseada no modelo de gerenciamento de objetos distribuídos do OMG, o que permite que usuários clientes possam acessar os bancos de dados sem qualquer conhecimento de localização ou formato de consultas ou operações envolvidas, visualizando apenas objetos homogêneos através de uma interface comum. A linguagem de consulta da interface é a OQL do OMG, e o modelo de dados comum é o ODMG-93.

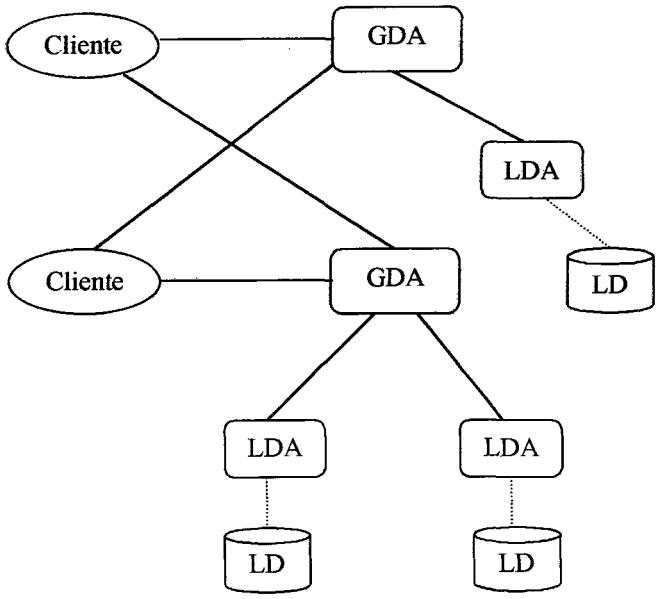

Figura 10.3: Arquitetura do MIND

Clientes acessam o sistema MIND através de uma classe de objetos chamada GDA (*Global Database Agents*). Os objetos desta classe são responsáveis por analisar e decompor as consultas. Para isso, utilizam o *Schema Information*

Manager, que possui informações sobre o esquema global. Esses objetos são responsáveis também pelo gerenciamento de transação global, que assegura a seriabilidade de transações globais planas e aninhadas sem violar a autonomia dos SGBDs locais, e por programar e controlar operações entre locais diferentes. Os objetos da classe GDA tratam operações de *commit* global e *abort* global utilizando protocolo 2PC sobre objetos da classe LDA.

Para um cliente o GDA é um servidor e para o GDA os *Local Database Agents* (LDA) são servidores. Os objetos da classe LDA são responsáveis por manter esquemas de exportação fornecidos pelos SGBDs locais, fornecer uma interface para os SGBDs locais traduzindo consultas recebidas na linguagem de consulta global para a linguagem de consulta local.

Cada banco de dados corresponde a um objeto da classe Database Object (definida no CORBA) que transfere solicitações do cliente para os SGBDs componentes. As interfaces de chamadas (*call interfaces*) dos SGBDs suportam a definição de dados em SQL, manipulação de dados, consultas e facilidades para controle de transação, implementadas em C++.

É um sistema de banco de dados com acoplamento forte e a integração de esquemas de exportação é executada manualmente, com o uso de uma linguagem de definição de objetos ODL – Object Definition Language. O administrador de banco de dados constrói o esquema integrado sobre os esquemas de exportação. A ODL permite seleção e reestruturação de elementos de esquema a partir dos esquemas locais.

MIND utiliza o modelo de transação aninhada em dois níveis: transações locais e globais. O gerente de transações é executado a nível do topo do CORBA, tratando a gerência de transação, o controle de concorrência e ordenação de eventos.

Após uma consulta ser decomposta, as sub-consultas são enviadas para os respectivos objetos LDA. O processo de otimização tem início assim que se torna disponível o primeiro resultado parcial. Para otimizar operações entre os *sites* é utilizado um mecanismo estatístico de inferência.

10.3 Projeto Jupter [Murphy e Grimson, 1995]

O projeto Jupter tem como objetivo básico permitir que sistemas autônomos e, possivelmente, heterogêneos cooperem e compartilhem informação de uma forma controlada.

Sua arquitetura consiste em um conjunto de serviços e uma linguagem para múltiplos bancos de dados, a JIL- *Jupter Interoperator Language* que permite que provedores de informação construam sistemas de informação fracamente acoplados, autônomos e interoperáveis.

A arquitetura de esquemas possui quatro níveis: esquema local, esquema de participação, esquema de exportação e esquema federado. Os componentes da arquitetura (figura 10.4) do Jupter são: *system services, application services, dictionary (metadata), query services, transaction services, data movement services, object services e negotiation services.*

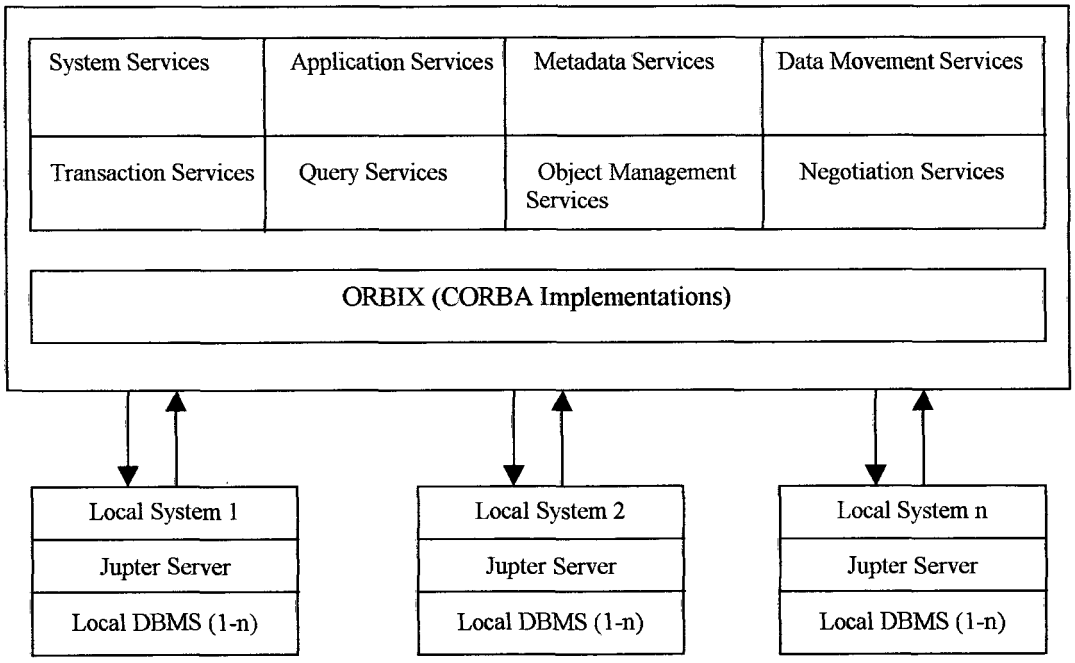

Figura 10.4: Arquitetura do Jupter

Um servidor CORBA (Orbix) é necessário em cada local onde o Interoperador Jupter estiver sendo executado. Os SGBDs locais são encapsulados como clientes e servidores Orbix. Solicitações remotas são tratadas por chamadas ao Jupter, que estabelece um objeto representante local para a solicitação remota.

As solicitações locais remotas são tratadas como se a fonte remota do dado fosse, de fato, uma fonte local; o objeto representante encaminha a solicitação para o local remoto do *Jupter* e, quando a solicitação é processada, o dado é devolvido pelo representante no site local. O objetivo é que o usuário final tenha uma única imagem do sistema, independente da quantidade de SBDs heterogêneos participantes.

10.4 HEROS - HetERogeneous Object System [Castro, 1998]

HEROS - HetERogeneous Object System é um SGBDH orientado a objetos desenvolvido no departamento de informática da PUC-Rio, sob o patrocínio do CNPq. Possui acoplamento forte, que provê aos usuários transparência de localização e replicação das informações acessadas.

O modelo de dados utilizado no HEROS para expressar o esquema global é o modelo de objetos, escolhido devido a sua expressividade e minimalidade. Para representar a informação (sobre os SGBDs componentes) necessária para permitir o acesso e interoperação (meta-informação) é utilizada a hierarquia de classes. Desta forma, representa-se no esquema do HEROS, não somente os esquemas dos sistemas a serem integrados, mas também as próprias características dos modelos de dados e SGBDs dos componentes, com as regras de mapeamento entre estes e o modelo global da federação.

Deste modo, é possível a extensibilidade da federação, o que permite que qualquer novo sistema componente possa ser integrado à federação, sem a necessidade de alterações na estrutura base do HEROS. Para integrar um novo componente, cujas características de modelo de dados ou sistema de gerência ainda não existam na federação, basta defini-lo através da criação de classes que descrevam suas características, juntamente com suas respectivas regras de mapeamento,

deixando, então, que o próprio HEROS faça a tradução de esquemas automaticamente [Castro, 1998].

A arquitetura de esquemas do HEROS é dividia em quatro níveis (figura 10.4): esquema local (EL), esquema de exportação (EEsp), esquema global (EG) e esquema externo (EE).

O esquema local é o próprio esquema do SGBD componente, expresso no seu próprio modelo de dados. Esquema de exportação é o esquema local do SGBD componente expresso no modelo de dados do HEROS. Esquema global é obtido pela integração de todos os esquemas de exportação e esquema externo representa uma visão global do esquema integrado do HEROS.

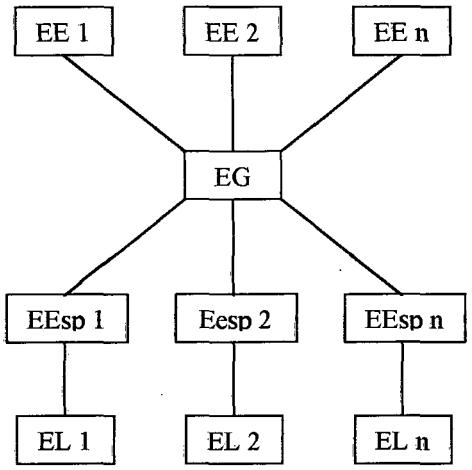

Figura 10.5: Arquitetura de esquemas do HEROS

Conforme mencionado, o HEROS usa um modelo de dados orientado a objetos, o que permite que tudo seja modelado através da representação de objetos, desde a meta-informação até as instâncias dos bancos de dados.

Os elementos do modelo de dados HEROS são representados na figura 10.5. No capítulo 6, foram descritos maiores detalhes sobre a gerência e execução de transações no HEROS.

O controle de concorrência em SGBDHs deve ser efetivado tanto no nível global quanto no nível local. Para o controle de concorrência local no HEROS, tendo em vista que a autonomia dos SGBDs componentes impede a interferência do

gerente global no controle de transações, foram restringidos os tipos de SGBDs que podem participar, através de requisitos que devem ser atendidos pelos protocolos empregados no controle da concorrência. Cada SGBD componente deve efetuar o controle da concorrência com o uso do protocolo 2PL estrito [Bernstein et al., 1987].

Para o controle de concorrência global, o HEROS apresenta um método baseado no Método de Tíquete Implícito – ITM [Georgakopoulos et al.,1994], estendido para possibilitar a garantia de serialibilidade das transações globais mesmo no caso de ocorrência de falhas acrescentando ao mecanismo original um mecanismo para controle de acessos baseado também no protocolo 2PL.

10.5 DDTS - Distributed Database Testbed System [Buretta, 1997]

DDTS foi desenvolvido por Honeywell Corporate Computer Science Center. O projeto enfatiza a modularização e flexibilidade e é composto por subsistemas que provêem serviços de interface com o usuário, tradução de consultas e execução distribuída. Como no *Multidatabase* (seção 9.1), ele é capaz de integrar SGBDs e prover algumas facilidades adicionais.

A arquitetura do DDTS consiste em um conjunto de processadores de aplicação (application processors: AP) e processadores de dados (data processors DP). Os Aps controlam a interface com os usuários e gerenciam aplicações enquanto os DPs gerenciam dados. Ambos são alocados a processadores físicos nos *sites* durante a configuração do sistema. Um subsistema de comunicação trasfere mensagens entre Aps e DPs.

Sua arquitetura em cinco níveis (seção 4.1), utiliza um esquema global que é uma descrição relacional de toda a estrutura dos bancos de dados. A linguagem de manipulação de dados utilizada é GORDAS. O DDTS é capaz de integrar SGBDs do tipo Codasyl. Seus componentes (figura 10.5) são divididos entre Aps e DPs. Processadores de aplicação (Ap) incluem quatro módulos: interface, tradutor e controlador de integridade, planejador de acesso e monitor de execução distribuída. Processadores de dados (DP) incluem dois módulos: o monitor de execução local e o módulo de operação local.

A interface provê a interação do usuário com DDTS. Este componente fornece funções para armazenamento, edição e execução de aplicações. O tradutor traduz uma consulta na linguagem GORDAS para álgebra relacional. Informações para mapeamento são armazenadas no esquemas de representação, conforme ilustra a figura 10.6.

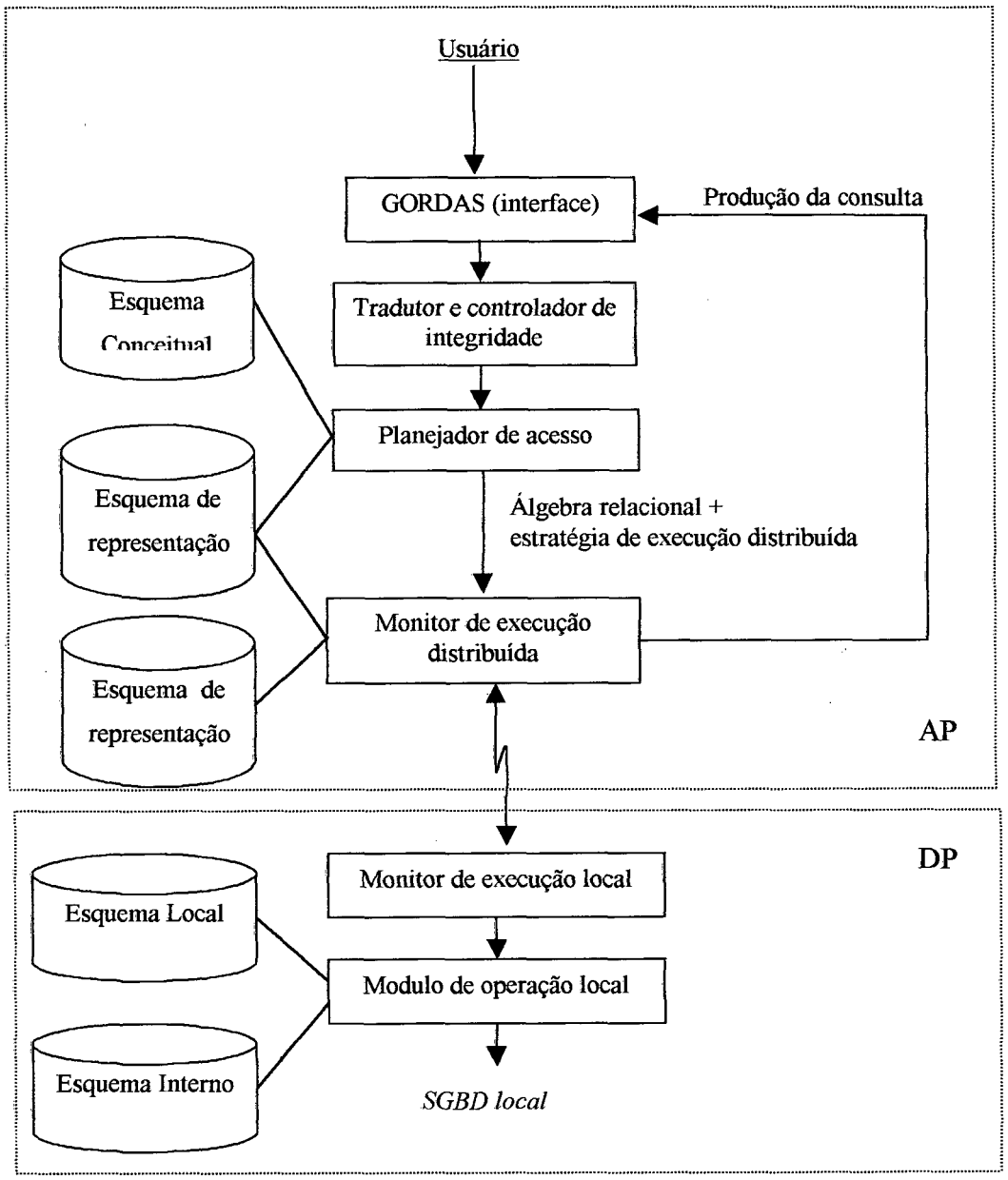

Figura 10.6: Componentes de Software do DDTS

O planejador de acesso propõe uma estratégia para um processamento eficiente das aplicações distribuídas. O algoritmo de otimização implementado no DDTS determina o custo mínimo de transmissão selecionando cópias que estão mais próximas do *site* de origem da aplicação. Deste modo a estratégia de custo mínimo é determinada.

O monitor de execução distribuída (DEM) e o monitor de execução local (LEM) cooperam na execução de transações. O DEM cria um conjunto de processos LEM. Cada processo DEM é retido até que a transação é confirmada (*commit*) ou abortada (*abort*). Os algoritmos utilizados são os 2PC (2-phase-commitment) e 2PL (2-phase-locking). Finalmente, o módulo de operação local traduz e otimiza as subtransações.

DDTS ainda tem vários problemas a serem solucionados, como otimização de consultas, controle da concorrência, regras de integridade, etc. Na tentativa de solucionar cada um destes problemas, avaliações empíricas e protótipos são desenvolvidos.

10.6 Projeto FLASH [Silva, 1998]

O FLASH é um projeto coordenado pelo Departamento de Informática da Universidade Federal de Pernambuco (DI-UFPE) cujo objetivo é investigar, propor e implementar teorias, modelos, técnicas e ferramentas de suporte à integração de sistemas heterogêneos. As áreas de atuação do projeto são: administração de sistemas, análise de desempenho e gerência de redes e integração de SGBDs.

Este projeto é uma combinação de esforços acadêmicos e industriais para buscar soluções de problemas complexos e relevantes no espírito da pesquisa cooperativa e multidisciplinar apoiada e incentivada pelo ProTeM-CC. Soluções inovadoras, que acompanhem o passo evolutivo das tecnologias de sistemas abertos e heterogêneos, somente serão possíveis através do esforço de um grande número de especialistas da indústria e da academia, trabalhando cooperada e coordenadamente. Este é o objetivo do FLASH.

No desenvolvimento do FLASH, quatro classes de resultados são esperados: inovações científicas e tecnológicas que forneçam subsídios para a integração de sistemas heterogêneos; desenvolvimento de ferramentas de suporte a administração de sistemas; introdução de novas tecnologias na Internet brasileira, no setor produtivo e no meio acadêmico; formação de recursos humanos com potencial multiplicador dos conhecimentos gerados pelo projeto.

A participação dos principais fornecedores de plataformas abertas cliente-servidor do mercado, IBM, Digital e HP, no FLASH permite que a pesquisa seja desenvolvida em um ambiente realista e os resultados possam estar em consonância com as novas tecnologias destes fornecedores. A interação com os times de P&D das empresas permitirá que alguns resultados do projeto sejam incorporados a ferramentas e plataformas de administração e gerência destes fornecedores.

Os resultados esperados são, portanto: protótipos e ferramentas; inovações tecnológicas a serem introduzidas no setor produtivo e na Internet brasileira; publicações e recursos humanos especializados.

10.6.1 Integração de SGBDs Heterogêneos e Distribuídos no FLASH

Conforme citado na seção anterior, uma das áreas de atuação do projeto é a integração de SGBDs. O objetivo é instalar e integrar bases de dados utilizando SGBDs distintos: O2, DB2, Informix, Oracle, etc.

A integração de sistemas relacionais e orientados a objeto não tem sido abordada de forma sistemática. Estudando este problema, o FLASH pretende contribuir com soluções originais na área de integração de SGBDs heterogêneos. A integração de SGBDs heterogêneos tem se concentrado em sistemas centralizados.

Algumas questões práticas são colocadas pelas empresas que não encontram soluções prontas em livros, nem em fornecedores no mercado. Dentre estas questões podemos citar:

- como, a partir de novas aplicações que justificassem o uso de SGBDOO, teríamos um diálogo com os SGBD relacionais implantados?
- como garantir uma boa integração entre as bases de dados já existentes em SGBD relacionais e os novos SGBD orientados a objetos desenvolvidos?
- como integrar esquemas conceituais e lógicos em ambientes heterogêneos?
- como garantir ao usuário a transparência na utilização de SGBD, sistemas operacionais e redes de computadores distintos?

Os experimentos de integração de SGBDs heterogêneos e distribuídos no FLASH vão buscar soluções para várias das questões acima. Os resultados dos projetos terão impacto nas empresas e organizações em que tais questões são parte do dia-a-dia da administração e integração de sistemas.

10.7 Outros SGBD Heterogêneos

Além dos SGBDH citados nas sessões anteriores, existem ainda muitos outros. A seguir citamos mais alguns, com as suas devidas referências caso necessário maiores informações sobre eles. Apesar de alguns deles serem antigos, são projetos de grande importância que contribuíram e ainda contribuem para a evolução dos sistemas gerenciadores de bancos de dados heterogêneos.

- VODAK [Muth et al., 1992] um sistema de banco de dados orientado a objetos, em desenvolvimento no Integrated Publication and Information Systems Institute (IPSI), projetado como uma plataforma para a integração de diferentes e pré-existentes sistemas de banco de dados. Os trabalhos publicados enfatizam a proposta de um modelo para as transações submetidas pelos usuários dos dados globais e a descrição dos princípios empregados na gerência destas transações.

- DOM [Georgakopoulos et al., 1993], é um projeto de um sistema conduzido nos GTE Laboratories, cujo objetivo é a integração de sistemas heterogêneos. Um modelo de transação bastante complexo é desenvolvido, mas os critérios de correção para validação das execuções ainda não foram desenvolvidos.

- PEGASUS [Rafii et al., 1991], um SGBDH que está sendo desenvolvido nos *Hewlett-Packard Laboratories*. Seu objetivo é possibilitar o acesso de aplicações globais a bancos de dados independentes e diferentes entre si, que sigam modelos de dados com orientação a objetos ou relacional. Nos trabalhos referenciados, a ênfase recai na descrição da arquitetura e nos mecanismos para solução de conflitos durante o processo de integração de esquemas locais.

- MERMAID [Templeton et al., 1987], um sistema desenvolvido na SDC, uma empresa que se tornou parte da UNISYS. Seu esquema global é relacional, suportando duas linguagens de consulta. A ênfase no sistema foi a otimização das consultas globais. A atualização dos bancos de dados não estava disponibilizada, fazendo parte dos trabalhos previstos o desenvolvimento e a incorporação ao protótipo desta funcionabilidade.

- MYRIAD [Clements et al., 1993] é um protótipo de sistema de bancos de dados federados em desenvolvimento na Universidade de Minnesota. Seu propósito é servir como ambiente para avaliação e comparação de soluções para os problemas de bancos de dados federados, como a gerência de transações e a otimização de consultas

A tabela 10.1 apresenta os sistemas gerenciadores e suas características.

Sistema	Tipo de Acoplamento	Modelo de Dados	Linguagem de Acesso Global	Características
Multidatabase	Forte	Orientado a objetos	Daplex	O multidatabase tem como restrição o fato de não fazer um controle ideal da consistência dos bancos de dados.
MIND	Forte	Orientado a objetos	GDA	MIND utiliza o modelo de transação aninhada em dois níveis: transações locais e globais. O gerente de transações é executado a nível do topo do CORBA, tratando a gerência de transação, o controle de concorrência e ordenação de eventos.
Jupter	Forte	Orientado a objetos	JIL	Os SGBDs locais são encapsulados como clientes e servidores Orbix. Solicitações remotas são tratadas por chamadas ao Jupter, que estabelece um objeto representante local para a solicitação remota.
HEROS	Forte	Orientado a objetos	OQL (da OMG)	Utiliza para o controle de concorrência global um método baseado no Método de Tíquete Implícito estendido para possibilitar a garantia de serialibilidade das transações globais.
DDTS	Fraco	Orientado a objetos	GORDAS	DDTS ainda possui vários problemas a serem solucionados, como otimização de consultas, controle da concorrência, regras de integridade, etc.
Pegasus	Fraco	Íris e Orientado a objetos	HOSQL (extensão da OSQL)	Trata conflitos e otimização de consultas.
Vodak	Fraco	Orientado a objetos	VML	Utiliza metaclasse, novas idéias para gerenciamento de transações para resolver conflitos.

Tabela 10.1: Características dos SGBDH

10.8 Comentários Finais

Neste capítulo foram descritos alguns sistemas gerenciadores de bancos de dados heterogêneos: *Multidatabase*, MIND, Jupter, HEROS e DDTS. Na última seção foi descrito um projeto da Universidade Federal de Pernambuco que tem como um dos objetivos propor e implementar modelos, técnicas e ferramentas para integração de sistemas de banco de dados heterogêneos.

Existe um grande número de sistemas já existentes, bem como projetos sendo desenvolvidos. Apesar das diferenças, todos eles possuem como principal objetivo desenvolver uma arquitetura e ferramentas que propiciem interoperabilidade entre sistemas de banco de dados heterogêneos. Devido ao grande número de pesquisas, novas soluções são propostas o que faz com que esses sistemas estejam em constantes modificações.

Capítulo 11

CONCLUSÃO

11.1 Resumo do Trabalho

Bancos de dados heterogêneos surgem quando vários bancos de dados distribuídos, autônomos e diferentes precisam compartilhar informações. A principal vantagem do uso da tecnologia de banco de dados é que nela se tem uma visão de um nível de abstração mais alto, em que os detalhes de implementação são abstraídos, sendo possível o compartilhamento e acesso a dados de uma maneira eficiente e confiável.

Neste trabalho, após um vasto levantamento bibliográfico, foi realizado um estudo acurado sobre os bancos de dados distribuídos heterogêneos. Foram pesquisadas e analisadas as principais soluções propostas e restrições neste ambiente. Foram apresentadas as principais abordagens dividindo-as em capítulos conforme o assunto: processamento e otimização de consultas, gerenciamento e concorrência de transações e finalmente interoperabilidade neste ambiente.

Apresentou-se também, um capítulo inicial sobre banco de dados distribuído homogêneo e um capítulo final com alguns projetos e sistemas gerenciadores de bancos de dados heterogêneos que se destacaram no ambiente acadêmico e comercial.

A seção seguinte, versa sobre as conclusões alcançadas após a realização deste estudo.

11.2 Conclusões

Considerando os fatores autonomia, distribuição e heterogeneidade, existem diversas maneiras de distribuir dados, originando diversas arquiteturas de sistemas de bancos de dados.

Entre elas, estão os sistemas de bancos de dados distribuídos heterogêneos. Sistemas de bancos de dados distribuídos heterogêneos são resultantes da integração de múltiplos e pré-existentes sistemas de bancos de dados interligados por redes de comunicação. O SGBDD heterogêneo é um software capaz de gerenciar os SGBDs diferentes provendo transparência não só da distribuição dos dados, mas também dos diferentes sistemas que o usuário acessa.

Geralmente os bancos de dados individuais já existem e são autônomos. Para prover interoperabilidade entre eles é necessário integrá-los. Existem duas alternativas propostas para integração: com esquema global e sem esquema global.

Através da primeira alternativa, obtém-se um esquema global que é a união dos esquemas locais dando origem a um sistema fortemente acoplado. Apesar da construção de um esquema global ser complexa, a grande vantagem deste modelo é que ele provê um nível de transparência máxima ao usuário. Porém, quando a heterogeneidade entre os esquemas a serem integrados for grande, ele torna-se inviável devido à dificuldade na automação e na adaptação a mudanças dos esquemas locais que, quando ocorrem, a integração deve ser refeita.

A segunda alternativa propõe a integração sem um esquema global, originando um sistema fracamente acoplado. É de responsabilidade do usuário criar e atualizar visões facilitando as mudanças dos esquemas componentes. Porém não fornece total transparência como no modelo com esquema global.

Independente do modelo de integração, o processamento de consultas em um sistema heterogêneo é mais complexo que nos sistemas distribuídos tradicionais (homogêneos). Esta complexidade existe principalmente devido ao fato de cada componente dos SGBDs locais possuírem capacidades (de execução e otimização) diferentes, impedindo a avaliação de custos necessária para a otimização de

consultas globais. Além disso, um *site* pode não estar disponível no começo ou interromper seu serviço posteriormente.

O processamento de consultas em sistemas multidatabase pode ser executado conforme os passos: decomposição da consulta global, fragmentação de cada subconsulta e tradução. Uma outra proposta apresentada foi a utilização da arquitetura de mediadores (sem esquema global). Ambos fornecem transparência e facilidade de acesso a dados globais, porém, devido às dificuldades encontradas na homogeneização e integração de esquemas em um esquema global, a arquitetura de mediadores tem sido apontada como uma solução mais adequada.

A otimização de consultas também é mais complexa devido à autonomia e heterogeneidade dos bancos de dados locais. Apesar de existirem vários trabalhos concluídos, muitas soluções novas estão sendo propostas. Foram apresentados no capítulo 6 vários modelos com diferentes abordagens. Entre eles, os que tratam os SGBDs componentes como uma caixa preta têm atraído mais atenção.

Existem ainda os modelos que estimam custos locais. Estes devem utilizar um algoritmo capaz de identificar os custos das consultas locais para que em uma fase posterior seja calculado o custo global. Nos últimos anos, vem sendo propostos algoritmos para otimização baseados em outros fatores diferentes de custos locais.

Em um ambiente de múltiplos bancos de dados existem transações locais, gerenciadas pelos SGBD componentes e transações globais, controladas pelo MSGBD. Cada transação global é dividida em um conjunto de sub-transações e submetidas aos sistemas locais.

Existem vários modelos de transações, adaptados para os diferentes tipos de aplicações de banco de dados. Apesar de a cada dia surgirem novas áreas de aplicação e conseqüentemente novos modelos de transação, a base destes novos modelos é o da transação plana. Os problemas de recuperação e concorrência estão estritamente ligados à noção de processamento de transações. A gerência de transações em um SGBDH tem por objetivo a obtenção de dados consistentes e a preservação da consistência global na presença de atualizações globais.

Para controle da concorrência de transações existem vários algoritmos que objetivam manter a consistência e isolamento das transações, além de satisfazer um

grande número de condições. O controle da concorrência das transações pode ser feito de várias maneiras. Classificam-se os métodos propostos para controle da concorrência [Özsu, Valduriez, 1999] em duas classes: métodos otimistas e métodos pessimistas. Ambos devem lidar com os problemas como conflitos indiretos e deadlocks.

Um dos modelos apresentados [Lee, Park, 1998] destaca-se por considerar as restrições de integridade para o gerenciamento de transações em sistemas *multidatabases*. Este método, comparado com outros modelos possui um menor índice de reinicialização de transações globais, garantindo uma maior performance e mantendo as regras de integridade locais e globais.

Faz-se necessário prover a interoperabilidade dos sistemas envolvidos na federação. Esta interoperabilidade, entretanto, ainda é um problema que tem desafiado empresas e universidades. No sentido de permitir comunicação entre SGBDs, muitas pesquisas vêm sendo desenvolvidas tratando a questão da interoperabilidade principalmente através da utilização do modelo orientado a objeto. A utilização da orientação a objetos na interoperabilidade entre sistemas em geral cresce a cada dia. Apesar da modelagem relacional dominar o mercado, o esquema conceitual global padrão orientado a objetos proporciona várias vantagens.

Para tratar e simplificar a computação em sistemas distribuídos heterogêneos, o OMG propôs o padrão CORBA, que vem sendo utilizado em inúmeros projetos para integrar sistemas de bancos de dados. Esta arquitetura possibilita que objetos em sistemas heterogêneos possam ser acessados de forma independente de sua implementação.

As pesquisas na área de sistemas gerenciadores de bancos de dados heterogêneos fizeram com que surgissem vários projetos e sistemas comerciais que tratam heterogeneidade. Foram apresentados neste trabalho alguns SGBDH. O caso do JUPTER, que possui uma estrutura de acoplamento fraco, tem uma desvantagem se levar-se em consideração que o usuário precisa ter mais conhecimento sobre os componentes para obter informação.

O HEROS, além de ser um SGBDH, detem duas grandes vantagens que se aliam a isso: seu modelo de dados orientado a objetos facilita a integração de novos

componentes e o esquema global torna transparente para o usuário os detalhes dos componentes participantes do banco de dados.

Apesar da evolução nos últimos anos, o estado atual das pesquisas em sistemas de bancos de dados heterogêneos indica ainda não haver conhecimento suficiente que possibilite a definição de padrões para os principais aspectos considerados neste trabalho: integração, modelo de transação, planejamento e otimização de caminhos de acesso a dados, controle da concorrência e interoperabilidade. A maioria dos projetos e sistemas comerciais ainda possuem muitas restrições, o que faz com que o número de pesquisas à procura de novas soluções nesta área cresça a cada dia.

11.3 Relevância do Trabalho

Com a grande utilização de sistemas de dados e a diversidade das aplicações, cresce a necessidade de acesso integrado a informações distribuídas de forma transparente ao usuário. Atualmente, com a evolução tecnológica, constata-se uma grande preocupação das empresas em garantir os investimentos realizados e ao mesmo tempo migrar para novas plataformas. Neste processo, a maior dificuldade é como conviver de forma eficiente com um ambiente heterogêneo.

Sistemas de gerenciamento de bancos de dados distribuídos heterogêneos tem uma relevância destacada para estas empresas e indústrias que possuem um parque computacional heterogêneo.

A contribuição deste estudo para a solução dos problemas de heterogeneidade entre bancos de dados foi um levantamento sobre os problemas e soluções existentes sob a visão de diferentes autores. Foi realizado um vasto estudo sobre sistemas *multidatabase* e as alternativas foram propostas para lidar com esses sistemas que são de importância vital para as empresas.

A partir dos problemas e soluções apresentadas, poderão surgir novas propostas tratando as diversas restrições impostas pelo ambiente heterogêneo.

11.4 Perspectivas Futuras

O problema de integração de sistemas de bancos de dados heterogêneos é bastante crítico. Com a velocidade que se desenvolve a tecnologia da Informática, cada vez mais os sistemas vão se tornando legados.

Desta forma, além das soluções apresentadas nesta dissertação, cada problema pode ser individualmente estudado, sendo possível o surgimento de propostas mais eficientes que as já existentes.

Após a realização deste estudo, consideramos com grande interesse a linha de pesquisa relacionada à gerência de transações em sistemas gerenciadores de bancos de dados heterogêneos. O controle da concorrência de transações nestes sistemas tem sido o foco de pesquisas devido a sua grande problemática.

A gerência de transações globais é um assunto com muitas dificuldades aguardando soluções factíveis. Apesar da decomposição de uma consulta global em sub-consultas sobre os esquemas locais e operações de composição de resultados ser semelhante aos sistemas tradicionais, se faz necessário investigar alguns fatores adicionais. As transações devem ser executadas concorrentemente, de forma transparente e confiável para a aplicação. É necessário um grande esforço para manter os SGBDs locais isolados e preservar as propriedades ACID.

Os principais problemas no controle da concorrência neste ambiente são os de conflitos indiretos e *deadlok*. Um algoritmo para o controle de concorrência ideal deve tratar satisfatoriamente estes problemas.

Uma pesquisa seqüencial a este estudo, é um aprofundamento nos principais modelos para o controle de concorrência existentes aqui apresentados, a fim de se desenvolver um novo modelo capaz de garantir a execução concorrente de transações locais e globais satisfazendo as condições impostas pelo ambiente heterogêneo.

GLOSSÁRIO

ACID – propriedades importantes que devem ser preservadas nos bancos de dados locais: atomicidade, consistência, isolamento e durabilidade.

BOTTOM-UP – termo utilizado quando o projeto do banco de dados distribuído for executado pela integração de vários bancos de dados já existentes.

CORBA – padrão CORBA (Common Object Request Broker Architecture). É uma arquitetura proposta para integrar sistemas distribuídos e heterogêneos.

DEADLOCK – um deadlock ocorre quando uma transação, para poder continuar seu processamento, aguarda por uma outra que nunca estará disponível.

ECG – esquema conceitual global.

ESQUEMA CONCEITUAL GLOBAL – esquema definido através da união de todos os esquemas locais participantes do banco de dados heterogêneo.

GSG – grafo de serialização global.

GTG – gerenciador de transações globais.

GTL – gerenciador de transações locais.

GWF – grafo *wait-for*, método utilizado para a detecção de deadlock

INTEROPERABILIDADE – capacidade dos recursos de computação interagirem para executarem tarefas conjuntamente.

MODELO FORTEMENTE ACOPLADO – são sistemas de bancos de dados distribuídos que possui um esquema global.

MODELO FRACAMENTE ACOPLADO – são sistemas de bancos de dados distribuídos que não possuem um esquema global dos dados.

MSGBD - sistema gerenciador de múltiplos bancos de dados.

MT – método de *tickets* para controle da concorrência global.

MULTIDATABASE – sistema de múltiplos bancos de dados.

MULTI-SGBD – sistema gerenciador de múltiplos bancos de dados.

OCM – otimizador de consultas em sistemas *multidatabase*.

OMA – Object Management Archicture (OMA) é um ambiente que define uma arquitetura de gerenciamento de objetos com objetivo de capacitar interoperabilidade entre softwares de fornecedores diferentes.

OMG – Object Management Group, é uma associação comercial internacional, com objetivo de desenvolver um padrão para interoperação de sistemas heterogêneos distribuídos (principalmente orientados a objetos).

PARTICIPATION SCHEMA – camada onde cada banco de dados local define os dados que deseja compartilhar.

PEC – plano de execução de consulta.

SBD – sistema de banco de dados componente.

SBDD – sistema de banco de dados distribuído.

SBDH – sistema de banco de dados heterogêneo.

SGBD – sistema gerenciador de banco de dados.

SGBDD – sistema gerenciador de banco de dados distribuído.

SGBDH – sistema gerenciador de banco de dados heterogêneo.

SGBDOO – sistema gerenciador de banco de dados orientado a objetos.

SGG – sistema de gerência global de transações.

SGL – sistema de gerência local de transações.

SISTEMAS FEDERADOS – sistema de banco de dados distribuído heterogêneo, cujos bancos de dados participantes são autônomos.

SISTEMAS LEGADOS – são aqueles sistemas que estão em uso por muito tempo, que atendem aos requisitos dos usuários e são de difícil substituição.

SMDB – sistema de múltiplos bancos de dados (*multidatabase*).

TOP-DOWN – termo utilizado quando o projeto de um sistema distribuído é executado sem um sistema já existente.

TRANSAÇÃO – unidade atômica de computação consistente e confiável, composta por uma seqüência de operações atômicas sobre os dados.

REFERÊNCIAS

[Batini et al., 1986] Batini, C.; Leuzirini M.; Navathe S.B.. **A Compartive Analysis of Methodologies for Database Schema Integration**. ACM Computer Survey, vol. 18, n. 4, 323-364, December 1986.

[Bell e Grimson, 1992] Bell, D.; Grimson, J.. **Distributed Database Systems**. Addison-Wesley, 1992, 2ª ed. Pg. 44-55.

[Bernstein et al., 1987] Bernstein, P.; Hadizlacos, V.; e Goodman, N.. **Concurrency Control and Recovery in Database Systems**, Addison-Wesley, EUA, 1987.

[Bhargava, 1999] Bhargava, B.; **Concurrency Control in Database Systems**. IEEE transactions on knowledge and data engineering, vol. 11, n. 1, January/February 1999.

[Brodie e Stonebraker, 1995] Brodie, M.; Stonebraker, M.; **Migrating Legacy Systems: gateways, interfaces & the incremental approch**. São Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

[Buchmann et al., 1991] Buchmann, A. P., Ozsu, M. T., e Georgakopoulos, D.. **Towards a Transaction Management for DOM**. Tech. Report TR-0146-06-91-165, GTE Laboratories, Inc., Estados Unidos, 1991.

[Buretta, 1997] Buretta, M.. **Data replication: tools and techniques for managing distributes information.** Wiley Computer Publishing, 1997.

[Castro, 1998] Castro, C. E. P. S.. **Integração de Legacy Systems a Sistemas de Bancos de Dados Heterogêneos.** Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Jul. 1998.

[Clements et al., 1993] Clements, D. et al.. **Myriad: Design and Implementation of a Federated Database Prototype**. Tech. Report 93-76, CS Dept., University of Minnesota, Estados Unidos, 1993.

[Dogac et al., 1995] Dogac, A.; Kilic, E.; Ozhan, et. al.. **METU Interoperable Database System.** Technical Report 6-1, Software Research and Development Center, Scientific and Technical Research Council of Turkey, Middle East Technical University, Ankara Turkiye, Jun. 1995.

[Du et al., 1992] Du, W.; Krishmamurthy R.; Shan M.C.. **Query Optimization in Heterogeneous DBMS**. In Proc. Of the 18th Int. Conf on Very Large Data Bases, 277-292, Vancouver, August 1992.

[Elmagarmid et al., 1999] Elmagarmid A. K.; Rusinkiewicz, M.; Shet, A.. **Management of Heterogeneous and Autonomous Database Systems.** San Francisco: Morgan Kaufmann, 1999.

[Elmagarmid, 1992] Elmagarmid, A. K.. **Introduction to Advanced Transaction Models, in Database Transaction Models for Advanced Aplications.** Editado por A.K. Elmagarmid, Morgan Kaufmann Pub., EUA, 1992.

[Elmagarmid e Calton, 1990] Elmagarmid, A. K. e Pu, Calton; **"Guest Editor's Introduction to the Special Issue on Heterogeneous Databases".** ACM Computing Surveys, Vol. 22, n.3, september 1990.

[Elmasri et al., 1987] Elmasri, R.; Larson, J.; Navathe, S. B.. **Integration Algorithms for Database and Logical Database Design**. Technical Report, Golden Valley, Minn.: Honey-well Corporate Research Center, 1987.

[Evredilek et al., 1997] Evrendilek C.; Dogac A.; Nural S.; Ozcan F.. **Multidatabase query optimization.** Journal of Distributed and Parallel Databases, vol. 5, n. 1, 1997.

[Ferreira e Finger, 2000] Ferreira, J. E.; Finger, M.. **Controle de concorrência e distribuição de dados: a teoria clássica, suas limitações e extensões modernas.** Escola de Computação 2000. São Paulo, 2000.

[Gardarin e Valduriez, 1989] Gardarin G.; Valduriez P.. **Relational Databases and knowlwdge Bases**. Reading, Mass: Addison-Wesley, 1989.

[Georgakopoulos et al., 1994] D. Gerogakopoulos, M. Rusinkinkiewicz e A. Sheth. **Using Tickets to Ensure Serializability of Multidatabase Transactions**. IEEE Transactions on knowledge and Data Engineering, vol. 6, n.1, February, 1994.

[Georgakopoulos, 1990] Georgakopulos, D.. **Transaction Management in Multidatabase Systems**. PhD Thesis, Department of Computer Science, University of Houston, 1990.

[Georgakopoulos et al., 1993] Georgakopoulos, D.; Susinkiewicz, M.; Sheth, A.. **Serializability of Multidatabase Transactions Through Forced Local Conflicts.** In Proceedings of the 7 th Int'l Conf. On Data Engineering, Kobe, Japão, 1991.

[Gligor e Popescu-Zeletin, 1986] Gligor, V.; Popescu-Zeletin, R.. **Transaction Management in Distributed Heterogeneous Database Management Systems**. Inf. Syst., vol. 11, n. 4, 287-297, 1986.

[Helal e Elmasri, 1995] Helal, A.; Elmasri, R.. **Standard Developments for Database and Systems Interoperability**. In: Simpósio Brasileiro de Banco de Dados, Recife, 1995.

[Hurson e Bright, 1994] Hurson, A. R.; Bright, M.W.; Pakzad, S. H.. **Current Research in Multidatabase Projects, in multidatabase Systems: An Advanced Solution for Global Information Sharing.** IEEE Computer Society Press, US, 1994.

[Lee e Park, 1998] Lee, K.; Park, S.. **Stable Transaction Management for Preserving the Global Integrity Constraints in Multidatabase Systems.** IEEE Computer Society Press, US, 1998.

[Lee et al., 1998] Lee, C.; Ke, C.; Cahng, J.; Chen, Y.. **Minimization of Resource Comsuption for Multidatabase Query Optimization.** Technical Report. Departament of Computer Science, National Cheng-Kung University, Taiwan, 1999.

[Lee e Chen, 1997] Lee, C.; Chen, C.. **Query Optimization in Multidatabase Systems Considering Schema Conflicts.** Stable IEEE Computer Society Press, US, 1997.

[Litwin, 1988] Litwin, W.. **From Database Systems to Multidatabase Systems: Why and How**. In Proc. British National Conference on Databases, Cambridge University Press, 161-188, Cambridge, 1988.

[Martin, 1994] Martin, J.. **Princípios de Análise e Projeto Baseados em Objetos**. Editora Campus, 1994.

[Morzy e Krolikowski, 1997] Morzy T.; Krolikowski Z.. **Query Optimization in Multidatabase Systems: Solutions and Open Issues**. Proceedings of the 10th International Workshop on Database & Expert Systems Applications, IEEE, 1998.

[Murphy e Grimson, 1995] Murphy, J. & Grimson, J.. **Multidatabase Interoperability in the Jupiter System**. Information and Software Technology. Vol 37, N. 9, 1995.

[Muth et al., 1992] Muth, P. et al.. **A transaction Model for an Open Publication Environment**. In Database Transaction Models for Advanced Applications, edited by Ahmed k. Elmagarmid, Morgan Kaufmann Publishers, Inc., San Mateo, Estados Unidos, 1992.

[Nicol et al., 1993] Nicol, J. R.; Wilkes, C.T.; Manola, F.. **Object Orientation in Heterogeneous Distributed Computing Systems**. IEEE Computer, vol. 26, n. 6, June 1993.

[Obermarck, 1982] R. Obermarck. **Deadlock Detection for All Resource Class**. ACM Trans. Database System, , 7(2): 187-208, June, 1982.

[Oliveira, 1997] Oliveira, E. S.. **HEROS: A interoperabilidade de seus Componentes usando CORBA.** Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 129p., 1997.

[OMG, 1993] Object Management Group. **Object Management Archicture Guide**. John Wiley & Sons, Inc. 1993.

[Özsu e Valduriez, 1999] Özsu, M. Tamer; Valduriez, Patrick. **Principles of Distributed Database Systems**. New Jersey: Prentice Hall, 2ª ed., US, 1999.

[Rafii et al., 1991] Rafii, A.; et al.. **Multidatabase Management in Pegasus**. In Proc. First Int'l Workshop on Interoperability in Multidatabase Systems, 1991.

[Ram, 1991] Ram, S.. **Guest Editor's Introduction: Heterogeneous Distributed Database Systems.** IEEE Computer, vol. 24, n. 12, December 1991.

[Salza et al., 1998] Salza, S.; Barone, G.; and Morzy, T.. **A Distributed Algorithm for Global Query Optimization in Multidatabase Systems.** Proc. Of the 2nd Int. Symp. On Advances in Databases and Information Systems, ADBIS'98, LNCS 1475, 130-141, 1998.

[Sheth e Larson, 1990] Sheth, A.P.; Larson, J. A.. **Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases.** ACM Computing Surveys, vol. 22, n. 3, Sept, 1990.

[Silva, 1994] Silva, S. D.. **Sistemas de Bancos de Dados Heterogêneos: Modelo de Execução de Gerência de Transações**. Tese de doutorado em informática, Dept. de Informática PUC-Rio. Rio de Janeiro, 1994.

[Silva, 1998] Silva, F. Q. B.. **Projeto FLASH: Formalizações da Administração de Sistemas Heterogêneos: Modelos, Técnicas e Ferramentas.** Departamento de Informática, Universidade Federal de Pernambuco, 1998.

[Soley e Kent, 1995] Soley, R.M.; Kent, W.. **The OMG Object Model Orientation in Heterogeneous Distributed Computing Systems**. IEEE Computer, vol. 26, n.6, June 1993.

[Templeton, et al., 1987] Templeton, M. et al.. **Mermaid – A Front-End to Distributed Heterogeneous Databases**. In Proc. Of the IEEE, Vol. 75, n. 5, maio de 1987.

[Uchôa et al., 1996] Uchôa, E. M. A.; Melo, R. N.. Lifschitz S. L.. **Interoperabilidade de Objetos em Sistemas de Bancos de Dados Heterogêneos**. Monografias em ciência da computação, n. 45, PUC-Rio de Janeiro, 1996.

[Uchôa, 1999] Uchôa, E.M.A.. **Framework para Integração de Sistemas de Bancos de Dados Heterogêneos.** Tese de Doutorado, Departamento de Informática, PUC - Rio de Janeiro, 1999.

[Vinoski, 1993] Vinoski, S.. **Distributed Object Computing with CORBA.** C++ Report Magazine, July/August 1993.

[Yan et al., 1997] Yan, L. L.. Özsu, M. T.; Liu, L.. **Accessing Heterogeneous Data Through Homogenization and Integration Mediators**. In 2nd Int. Conf. On Cooperative Information Systems (CoopIS97'), 130-139, June 1997.

[Yao et al., 1982] Yao, S. B.. Waddle, V.; Housel, B.. **View Modeling and Integration Using the Functional Data Model**. IEEE Trans. Software Eng., vol 8, n. 6, 544-554, November 1982.

[Zhu e Larson, 1994] Zhu, Q.. Larson, P. A.. **A query sampling method for estimating local cost parameters in a multidatabase system**. Proc. Of 10th Int. Conf. On Data Eng., 144-153, Houston, 1994.