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

Daniela Eloise Flôr

TÉCNICAS, METODOLOGIAS E FERRAMENTAS PARA PROJETO DE BANCOS DE DADOS DISTRIBUÍDOS: UM ESTUDO DE SOLUÇÕES PROPOSTAS

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, julho/2001

TÉCNICAS, METODOLOGIAS E FERRAMENTAS PARA PROJETO DE BANCO DE DADOS DISTRIBUÍDOS: UM ESTUDO DE SOLUÇÕES PROPOSTAS

Daniela Eloise Flôr

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

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

Banca Examinadora

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

Prof. Dr. Vitório Bruno Mazzola

Prof. Dr. Roberto Willrich

Prof. Dr. Rosvelter J. Coelho da Costa

"A mente que se abre para uma nova idéia
jamais volta ao seu tamanho natural..."
Albert Einstein

AGRADECIMENTOS

À Deus,
pela luz divina com a qual rege a minha vida, minha gratidão.

Ao meu orientador Prof. Dr. Murilo Silva de Camargo,
pela confiança demonstrada e pelos direcionamentos sempre oportunos,
meu obrigada.

À toda minha família,
em especial ao meu pai que não pode estar presente em mais essa conquista,
minha vitória.

Aos verdadeiros amigos,
que caminharam comigo e contribuíram para a realização deste trabalho,
nossas conquistas.

E finalmente ao meu eterno amor,
razão dos objetivos alcançados e inspiração para continuar a eterna batalha
da vida, todos os meus esforços.

RESUMO

O desenvolvimento do projeto de banco de dados distribuídos é visto como a etapa que garante o retorno dos investimentos de quem aposta nessa tecnologia. Devido as várias decisões que necessitam ser tomadas ao longo da execução do projeto, essa tarefa passa a enfrentar vários obstáculos. O primeiro deles é a escolha da abordagem, ou seja, *top-down* para o ambiente em que banco de dados será iniciado a partir do zero ou abordagem *bottom-up* para bancos de dados preexistentes e posteriormente integrados. A próxima decisão será sobre as técnicas utilizadas, para a abordagem *top-down* é necessário resolver duas questões, o particionamento e alocação dos dados. O que consiste em resolver sobre a melhor técnica de fragmentação no caso do particionamento e a alocação otimizada desses fragmentos através dos sites participantes do banco de dados. E está restrito à abordagem *bottom-up* as técnicas de integração dos dados, ou seja, com ou sem esquema global. Para contribuir com o entendimento da relação entre esses fatores, este estudo apresenta algoritmos e heurísticas específicas para algumas técnicas, ferramentas e metodologias propostas por diferentes autores no intuito de fornecer boas sugestões para uma possível escolha de solução de boa qualidade.

Palavras-chave: banco de dados distribuídos, projeto de banco de dados distribuídos, fragmentação, alocação, integração, metodologias e ferramentas.

ABSTRACT

The development of a distributed database is seen a stage that assures the investments of those that bet in this technology. Regarding to several decisions that must be taken during the project accomplishment this task begins to face some obstacles. The first among them is the approach choice, for instance, *top-down* is appropriate for the database environment that begins from zero and the *bottom-up* approach is appropriate for pre-existing database that will be integrated later. The next decision will be on the technics that will be employed. The *top-down* approach requires the solution of two topics: the partition and the allocation of data. In the partition case the solution may consist in the best fragmentation technic as well as the optimized location of these fragments through database participating sites. The *bottom-up* approach restricts the data integration technics, anyway, with or without the global scheme. For a better understanding of the relation among these factors, this study presents specific algorithms and heuristics for some technics; tools and methodologies proposed by different authors aiming to provide helpful suggestions for a choice of good quality solution.

Keywords: *distributed database, distributed database design, fragmentation, allocation, integration, methodology and tools.*

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

Capítulo 1

INTRODUÇÃO

1.1 Considerações Iniciais

A tecnologia de comunicação dos dados vêm possibilitando a expansão do conceito de banco de dados, viabilizando o trabalho com informações distribuídas em áreas geograficamente dispersas e não apenas com informações centralizadas.

Essa idéia é muito atrativa e as vantagens oferecidas por esse conceito de distribuição reforçam o aumento de pesquisas nessa área. De acordo com [Buretta 1997], grandes empresas como a IBM, Oracle e Sybase, tem investido em pesquisas no intuito de produzir soluções para o ambiente descrito.

Mas para usufruir benefícios como: autonomia local, ininterrupção das operações, transparência, independência de hardware, sistema operacional e rede, conseguidos através dessa almejada "distribuição", algumas questões importantes como custo da instalação, da comunicação entre os sites, da segurança dos dados, integridade, redundância, disponibilidade de informação entre outras, devem ser consideradas.

Não se pode ignorar as dificuldades que esse tipo de banco de dados traz, a complexidade adicional requerida para assegurar a própria coordenação entre os nós, toma a forma de aumento no custo do desenvolvimento de aplicações, maior potencialidade de defeitos, diminuição da velocidade na obtenção das informações e aumento da sobrecarga de processamento.

Para que a distribuição dos dados alcance os objetivos da organização e não seja muito custosa, é necessário que o projetista considere algumas questões na fase de projeto, como: a abordagem ao tratar do banco de dados distribuídos (BDD), e

dessa abordagem outras decisões serão derivadas como quanto fragmentar, como fragmentar, como testar os resultados, como alocar os fragmentos, ou como integrar os dados já existentes, sempre utilizando os requisitos disponíveis, como, informações sobre o banco de dados, a aplicação, a comunicação dos dados e o sistema computacional.

O presente trabalho tem por objetivo explorar as técnicas de distribuição, alocação e integração dos dados e fazer um estudo das soluções sobre técnicas, metodologias e ferramentas propostas, fornecendo assim subsídios aos projetistas quando da avaliação dos benefícios em cada etapa da distribuição.

1.2 Metodologia

Na realização deste estudo, foram realizadas pesquisas em 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 contato com autores dos trabalhos pesquisados.

1.3 Organização do Trabalho

O presente trabalho foi desenvolvido conforme a distribuição descrita a seguir. Logo no capítulo 2 são tratadas as questões sobre banco de dados distribuídos, no intuito de revisar os conceitos fornecidos por essa tecnologia e reforçar o entendimento dos próximos capítulos.

Uma revisão sobre as especificações de projeto de banco de dados são comentadas no capítulo 3, assim como algumas propostas de modelagem de dados.

Os conceitos de projeto de banco de dados distribuídos incluem os mesmos princípios do projeto centralizado, acrescidos de detalhes específicos, que são objetos de estudo tratados logo a seguir, no capítulo 4. A deficiência de um projeto pode

causar problemas na performance, flexibilidade, confiabilidade e um considerável aumento nos custos de implementação, questões como essas complementam o capítulo.

As técnicas utilizadas dependerão da abordagem adotada. Na abordagem *top-down* é tratado o particionamento e alocação, no caso *bottom-up* trata-se a integração.

No capítulo 5 o assunto particionamento é tratado em detalhes, informações sobre técnicas de fragmentação dos dados, os tipos de fragmentação, algoritmos utilizados e regras para correção de fragmentos entre outros são exploradas.

No próximo capítulo são apresentadas as questões sobre a alocação dos dados, suas vantagens e desvantagens, relação custo/benefício e principais características.

A partir do capítulo 7 é apresentado o caso da integração de bases de dados já distribuídas e posteriormente integradas.

A apresentação de algumas ferramentas e metodologias são expostas no capítulo 8, no 9 encontra-se as conclusões deste estudo e a proposta para um trabalho futuro.

Logo a seguir há um glossário e são listadas as referências bibliográficas utilizadas no desenvolvimento desta dissertação.

Capítulo 2

BANCO DE DADOS DISTRIBUÍDOS

A tecnologia de banco de dados foi desenvolvida em vários estágios, a aplicação inicial pretendia resolver problemas com os sistemas de processamento de arquivos que aconteciam em meados dos anos 60. As limitações desse tipo de processamento fizeram com que grandes empresas começassem a desenvolver bancos de dados organizacionais com informações primeiramente centralizadas.

Mais tarde o modelo relacional baseado na Álgebra Relacional proposto por Edward Codd em 1970, em conjunto com a explosão da utilização do microcomputador na década de 80 e a melhoria nas interfaces com os usuários, proporcionaram o uso de aplicações de banco de dados também no setor pessoal.

Com o advento das redes na segunda metade dos anos 80, departamentos de grandes organizações começaram a implementar grupos de trabalho em banco de dados cliente-servidor. Esse desenvolvimento culminou em um novo aspecto para essa tecnologia, a necessidade de se compartilhar informações distribuídas em locais distintos.

Segundo [Ferreira e Finger, 2000] um sistema de banco de dados distribuídos é um conjunto interligado de unidades computacionais, chamadas de *sites* ou *nós*, cada uma contendo um Sistema Gerenciador de Banco de Dados (SGBD), idênticos ou não, capazes de processar as transações locais e as transações globais que solicitam o acesso a dados localizados em outros nós.

A principal diferença entre sistemas de bancos de dados centralizados e distribuídos é que no primeiro os dados estão armazenados em um único local, enquanto no último os dados residem em diversos locais apesar serem transparentes ao usuário, conforme ilustrado respectivamente nas figuras 2.1 e 2.2.

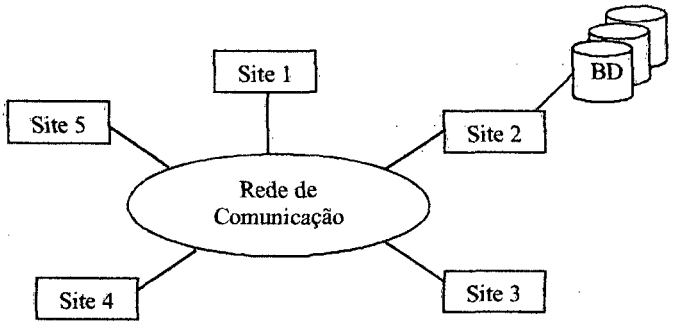


Figura 2.1 - Ambiente centralizado em uma rede de computadores

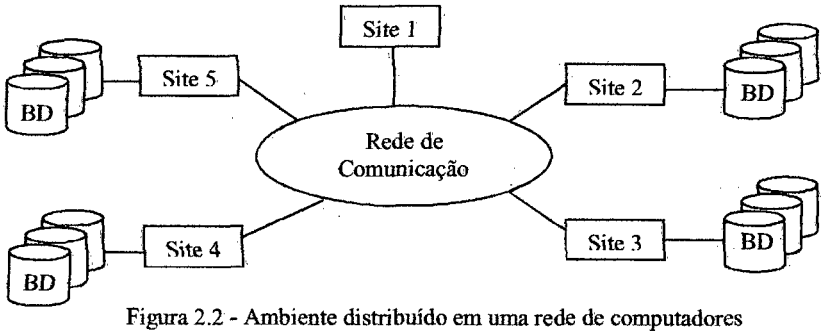


Figura 2.2 - Ambiente distribuído em uma rede de computadores

Cada site de um banco de dados distribuído (BDD) será 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 comunicação entre os nós, através de meios de comunicação como pares trançados, coaxial, fibras óticas, linhas telefônicas, ligações de microondas e canais de satélite, de sites que podem estar em áreas de diferentes portes.

Para gerenciar esse conjunto de informações é necessário um Sistema Gerenciador de Banco de Dados (SGBD).

2.1 Sistema Gerenciador de Banco de Dados

Por [Teorey, 1999] um sistema gerenciador de banco de dados é um software que consiste de um conjunto de informações inter-relacionadas e um conjunto de programas capaz de torná-las disponíveis quando solicitadas. Ele é o responsável em prover um ambiente que seja adequado e garanta a eficiência em operações como: gerenciamento de transação e controle de concorrência, estrutura de armazenamento lógico e físico, recuperação de informações, manipulação de dados, segurança das informações armazenadas, consistência e integridade dos dados, suporte a linguagem de manipulação, entre outras.

No caso distribuído, um sistema gerenciador de banco de dados distribuído (SGBDD) é um software que gerencia um banco de dados distribuído responsável pelo gerenciamento global e local dos dados [Özsu e Valduriez, 1999], de tal modo que os aspectos da distribuição ficam transparentes para o usuário, ou seja, a transparência garante a separação entre a semântica de alto nível do detalhamento de implementação do sistema, desta forma o usuário poderá trabalhar com informações armazenadas em sites diferentes sem perceber, tendo a ilusão de integração lógica de dados, sem requerer a integração física do banco de dados.

Vários tipos de transparência em banco de dados podem ser relevantes, como por exemplo: transparência de distribuição, de replicação, de fragmentação dos dados e a independência dos dados.

A transparência de distribuição impede que o usuário tenha detalhes sobre a operacionalidade da rede onde trafegam os dados e sobre a sua localização.

A transparência de replicação consiste em garantir a independência e a atualização de cópias, estejam elas replicadas ou não. Se estiverem, o SGBDD através de protocolos específicos assegurará que todas as cópias sejam atualizadas consistentemente, sem o envolvimento do programa de aplicação.

A transparência de fragmentação está relacionada à independência entre o modo de fragmentação dos dados e as aplicações dos usuários.

A independência de dados diz respeito à independência entre a definição e organização dos dados e as aplicações dos usuários.

A transparência em um SBDD, garante que os programas de aplicação não sabem e nem precisarão saber onde os dados estão localizados. O SGBDD encontra os dados, onde quer que eles estejam, sem o envolvimento do programa de aplicação.

Diante disto, 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, conforme ilustrado na figura 2.3, sendo que a realidade é ilustrada na figura 2.4.

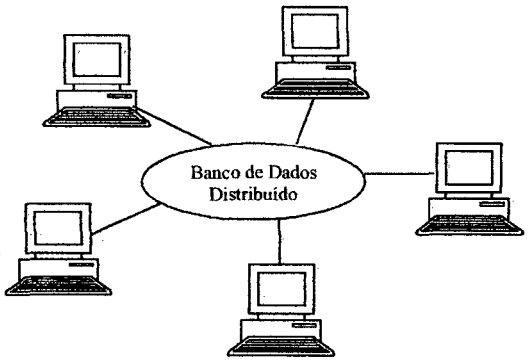


Figura 2.3 - Banco de dados distribuído (visão do usuário)

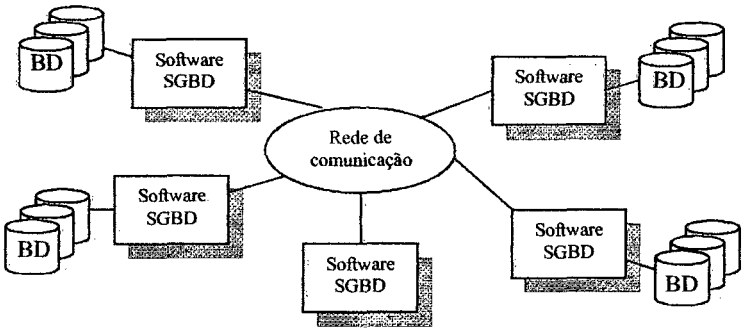


Figura 2.4 - Banco de dados distribuído (realidade)

Outra característica importante do SGBDD é o grau de autonomia assegurado em cada site. O termo autonomia refere-se à distribuição de controle e indica o grau no qual os 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.

Um outro conjunto de operações executáveis pelo SGBDD, é o processamento de consultas distribuídas, onde a partir de uma consulta do usuário é utilizada uma linguagem derivada do cálculo relacional.

2.2 Arquiteturas de Sistemas Gerenciadores de Banco de Dados Distribuídos

Segundo [Bell e Grimson, 1992] os SGBDDs podem ser classificados, segundo aspectos arquiteturais, em: SGBDDs Homogêneos, de acordo com a abordagem *top-down* e SGBDDs Heterogêneos de acordo com o abordagem *bottom-up.*

A arquitetura de um banco de dados é responsável por identificar e definir a função e o inter-relacionamento das interações entre os componentes do sistema. A arquitetura do sistema gerenciador de banco de dados distribuído homogênea, de acordo com a abordagem top-down, ou seja, quando o banco de dados distribuídos é inicializado do zero, pode ser analisada na figura a seguir.

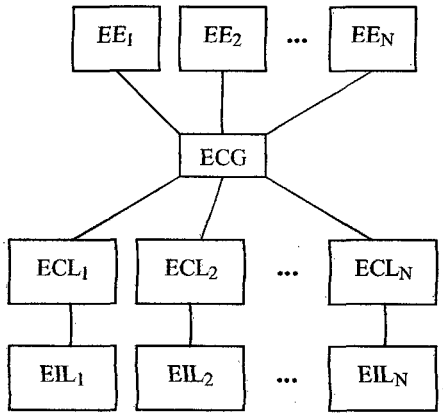


Figura 2.5 - Arquitetura data-lógica de um SGBDD

Na ilustração a sigla EE, representa os esquemas externos que são as visões definidas aos usuários globais. O esquema conceitual global – ECG, é formado pela união dos esquemas conceituais locais – ECL, que por sua vez possuem o EIL, esquema interno local, referente ao armazenamento físico dos dados em cada site. Essa arquitetura é denominada ponto-a-ponto (peer-to-peer), uma vez que cada site possui um SGBD completo, a figura 2.6 demonstra os seus componentes.

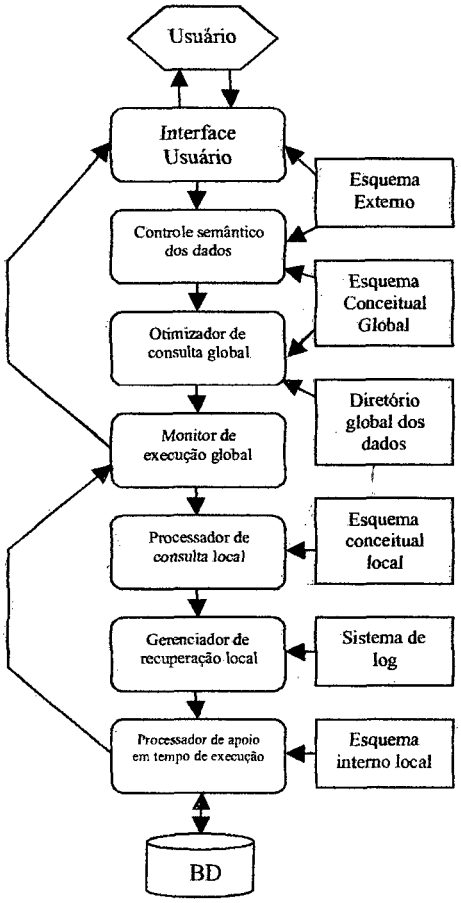


Figura 2.6 - Componentes da arquitetura ponto-a-ponto

A próxima ilustração demonstra uma nova arquitetura data-lógica, a de multi-SGBD. No caso os esquemas conceituas locais – ECL, possuem seu próprio EEL – esquema externo local.

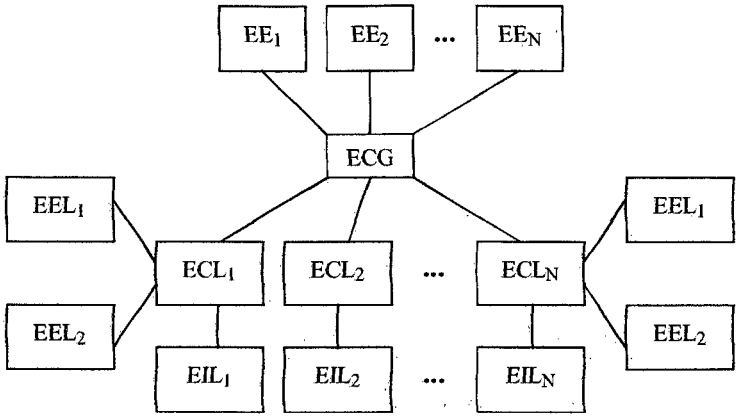


Figura 2.7 - Arquitetura data-lógica de multi-SGBD

Esta arquitetura atende a abordagem *bottom-up*, onde o projeto do banco de dados distribuído heterogêneo acontece a partir de base de dados já existentes. A maior diferença entre esta arquitetura e a anterior é a forma do mapeamento entre os esquemas, conforme ilustrou as figuras 2.5 e 2.7. A próxima ilustração revela os componentes de um SGBDD conforme a arquitetura de multi-SGBD.

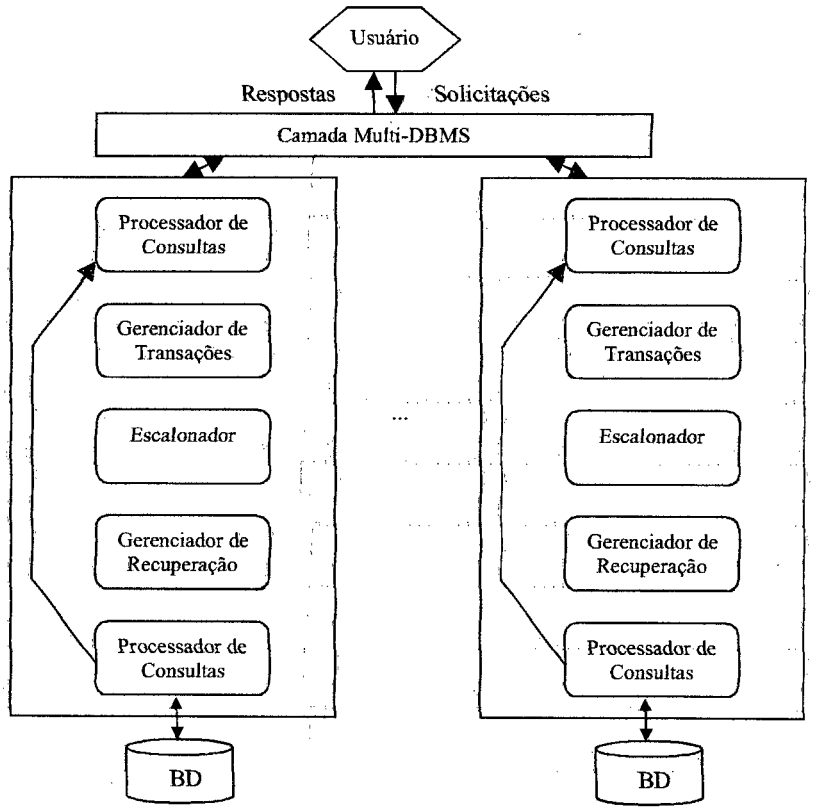


Figura 2.8 – Componentes da arquitetura data-lógica de multi-SGBD

A tabela abaixo resume as arquiteturas data-lógicas descritas anteriormente.

<table>
<tr><th>Aspectos</th><th>Arquitetura ponto-a-ponto</th><th>Arquitetura multi-SGBD</th></tr>
<tr><td>Ambiente</td><td>Homogêneo</td><td>Heterogêneo</td></tr>
<tr><td>Abordagem do BDD</td><td>Top-down</td><td>Bottom-up</td></tr>
<tr><td>SGBD nos sites</td><td>normalmente idênticos</td><td>normalmente diferentes</td></tr>
<tr><td>Semelhanças</td><td colspan="2">EIL – esquema interno local
ECL – esquema conceitual local
ECG – esquema conceitual global
EE – esquema externo</td></tr>
<tr><td rowspan="2">Diferenças</td><td></td><td>EEL – possui esquema externo local</td></tr>
<tr><td colspan="2">Mapeamento entre os esquemas</td></tr>
<tr><td>Autonomia</td><td>Acoplamento Forte</td><td>Acoplamento Controlado</td></tr>
</table>

Tabela 2.1 – Resumo das arquiteturas de sistemas gerenciadores de banco de dados distribuídos

2.3 Vantagens de um Banco de Dados Distribuídos

De acordo com [Özsu e Valduriez, 1999] existem diversas razões para a construção de um banco de dados distribuídos, como o compartilhamento dos dados, a transparência no gerenciamento dos dados, aumento da confiabilidade e disponibilidade dos dados, aumento da performance com a aceleração do processamento de consulta, facilidade e economia na expansão do sistema.

Uma dessas razões, a confiabilidade através de transações distribuídas é uma característica importante, através dela o acesso a dados compartilhados ocorre mesmo quando uma falha em algum dos sites acontece, nesse caso os sites remanescentes serão capazes de continuar operando de maneira confiável e eficiente.

O aumento da performance acontece quando dados fragmentados são alocados em sites onde eles serão mais utilizados, sendo que desta forma as informações disponibilizadas para o processamento requerido estarão a nível local. Outros fatores que auxiliam nesse benefício são as consultas paralelas executadas em casos de processamento a nível global. Essas consultas são subdivididas e executadas em paralelo, acelerando seu processamento e minimizando o período de resposta.

No ambiente discutido é muito mais fácil distribuir novas bases de dados entre os sites existentes, obtendo o aumento de flexibilidade, já que pode-se utilizar diferentes SGBDs e diferentes ferramentas, ocasionando dessa forma facilidade e economia na expansão do sistema.

Caso ocorra uma falha em um dos nós do sistema distribuído, os nós remanescentes podem ser capazes de continuar operando, aumentando a disponibilidade.

2.4 Desvantagens de um Banco de Dados Distribuídos

Entretanto, juntamente com os benefícios dessa tecnologia muitos problemas são detectados, conforme [Özsu e Valduriez, 1999] questões específicas com a replicação dos dados, como a manutenção das cópias, principalmente quando algum site estiver temporariamente inutilizado e sem o prejuízo no sincronismo do acesso, criam uma complexidade adicional em assegurar a coordenação entre os sites.

Essa complexidade adicional pode ser expressa de várias formas, como maior requisição de custos em nome da comunicação entre os sites, pois é importante garantir que a malha de interconexão entre os nós não se torne o gargalo do sistema, desenvolvimento de softwares apropriados que possuem naturalmente uma propensão a uma maior potencialidade de defeitos, a diversidade dos ambientes operacionais, além de crescente sobrecarga de processamento.

O controle da distribuição é um ponto crucial nesse ambiente, questões como a fragmentação do banco de dados e a forma de alocação das informações devem ser estudadas para evitar problemas de sincronismo e coordenação.

A forma em que os dados foram distribuídos podem ocasionar em uma consulta global o aumento da sobrecarga do processamento, visto que informações devem ser resgatadas em outros sites.

Outro aspecto que deve ser vigiado, é devido a utilização de redes, que deixam os dados vulneráveis, obrigando a um controle rígido de autorização de acesso, revisão de privilégios, criptografia dos dados, entre outras medidas que devem garantir a segurança do sistema.

A probabilidade de ocorrência de falhas nos sites pode ocorrer durante o trabalho de atualização dos dados decorrente de transações, o sistema deve garantir a recuperação das informações garantindo a confiabilidade do banco de dados distribuídos.

Os fatores complicadores comentados são refletidos em outros aspectos dos BDD como na fase de projeto, no processamento de consulta, no gerenciamento de diretórios, no controle de concorrência, gerenciamento de *deadlocks*, no suporte de sistema operacional próprio para essas operações, nos banco de dados heterogêneos e etc., conforme ilustrado na figura 2.9.

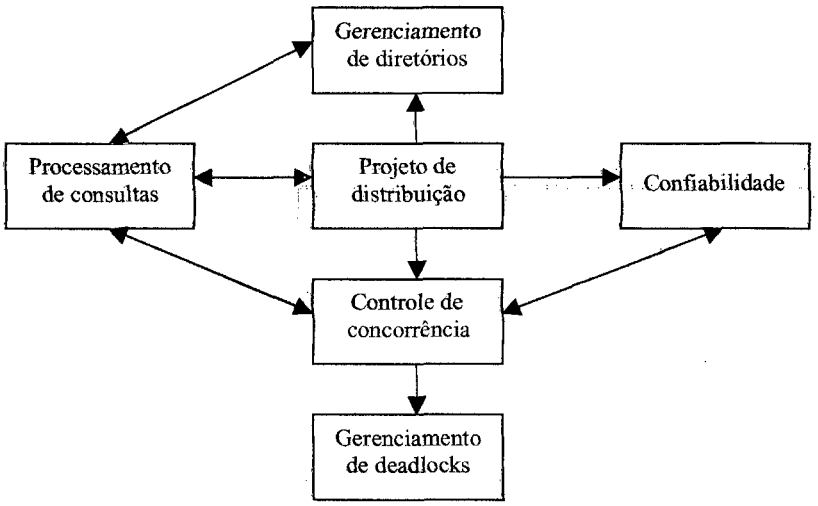


Figura 2.9 - Relacionamento entre os fatores complicadores

2.5 Questões de Pesquisas em Banco de Dados Distribuídos

2.5.1 Processamento de consulta

No ambiente distribuído o processamento das consultas passa por um processo de divisão. Primeiramente o cálculo relacional que deu origem a consulta é reescrito de maneira normalizada, logo em seguida é realizada a análise semântica para verificação de erros, os predicados redundantes são eliminados e por último é transformado em consulta algébrica, representada pela álgebra relacional.

Após a decomposição da consulta, os dados que se encontram fragmentados e replicados precisarão ser localizados. Os dados da consulta global são otimizados, no intuito de encontrar a melhor estratégia para a realização das consultas locais e

inclusive analisar as operações de comunicação e custo de transmissão de dados pela rede. Os sites locais por sua vez se encarregam de otimizar as consultas realizadas nele.

Os acessos simultâneos a recursos compartilhados normalmente são acompanhados de transações que preservam as propriedades ACID: atomicidade, ou seja as transações são realizadas integralmente ou não, consistência, a integridade dos dados deve ser mantida, isolamento, as alterações devem ser invisíveis, serializáveis e duráveis, as atualizações confirmadas devem persistir.

2.5.2 Controle de concorrência

O controle de concorrência permite o acesso simultâneo de forma transparente ao ambiente compartilhado. Para que isso ocorra é necessário que o gerenciador garanta que o BDD estará sempre consistente, mesmo que não o esteja em determinado instante por ocasião da execução de uma transação, ao final da mesma, esse estado deverá ser revertido e voltar a assegurar a propriedade.

O controle de concorrência utiliza algoritmos classificados em pessimista e otimista, existem vários algoritmos, todos tem por objetivo manter as propriedades das transações.

As falhas que podem ocorrer em um sistema distribuído são as mesmas de um sistema centralizado, como erros de hardware e software, entretanto alguns tipos adicionais de falhas também devem ser considerados, como a falha em um site, na comunicação, perda de informações, entre outras.

2.5.3 Recuperação de falhas

Para que um sistema seja robusto, é necessário que detecte falhas e reconfigure o sistema para que ele possa continuar funcionando, a partir da situação original antes da ocorrência da falha, garantindo dessa forma a confiabilidade.

2.5.4 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.

2.5.5 Projeto de Banco de Dados Distribuídos

O projeto de banco de dados distribuídos inclui os mesmo princípios de projeto de bancos de dados centralizados, mas envolvem outras decisões referentes ao projeto de distribuição. Esse assunto é visto com maiores detalhes no próximo capítulo.

2.6 Considerações Finais

Os conceitos iniciais de banco de dados distribuídos foram abordados neste capítulo, bem como suas vantagens e desvantagens, o sistema gerenciador de banco de dados e sua arquitetura entre outros, a fim de revisar conceitos necessários para o entendimento do resto do trabalho.

Para assegurar que as unidades computacionais, chamadas de *sites* ou *nós*, operem eficientemente, um bom projeto de banco de dados deve ser realizado, diante disto o capítulo seguinte começa a tratar projeto de banco de dados ressaltando, suas notações fases e importância.

Capítulo 3

PROJETO DE BANCO DE DADOS

Segundo [Machado e Abreu, 1995] a construção de sistemas com aplicações em banco de dados obedecem a fases bem definidas. Uma boa metodologia estabelece um caminho único a ser seguido, provendo uma lista de todas as atividades a serem realizadas, estabelecendo pontos de checagem para auditoria e controle do projeto, introduzindo consistência ao longo do desenvolvimento.

Existem várias metodologias que definem o ciclo de vida de um sistema, no qual estão mostradas as fases que compõem o caminho a ser seguido pelos analistas e programadores, até a produção do sistema na sua versão operacional. Cada fase pode ser vista como o refinamento da etapa anterior.

Algumas metodologias como a "*Ciclo de vida tradicional ou em cascata*", "*Ciclo de vida da análise estruturada*", "*Ciclo de vida da Engenharia de Software*", "*Ciclo de vida da engenharia da informação*" entre outras, são bastante utilizadas no processo de desenvolvimento de sistemas de informação.

Muitas delas já não apresentam o mesmo vigor de utilização de quando foram desenvolvidas, e praticamente formam um conjunto evolutivo no processo de captar as reais necessidades que o usuário possui em termos de automação de sua realidade.

Seja qual for a metodologia seguida, quando se fala em desenvolver um sistema é necessário definir qual o primeiro passo nas atividades de análise. Torna-se cada vez mais claro a extrema necessidade de obter o conhecimento sobre o que são e como são as informações, os dados do negócio, pois eles refletem o próprio negócio da organização. Com base neste enfoque, define-se como primeiro passo nas atividades de análise a modelagem de dados.

O objetivo da modelagem de dados é possibilitar a apresentação de uma visão única, não redundante e resumida dos dados de uma aplicação.

Quando passamos a adotar uma metodologia baseada em modelo de dados, com trabalhos iniciais e extensos de modelagem, conseguimos obter um domínio dos negócios da empresa, tendo uma impressão global da mesma.

Em qualquer dos ciclos de vida citados estão presentes a modelagem dos dados que culmina no projeto de banco de dados e conseqüentemente na sua construção, pode-se dividir o projeto em três fases bem distintas, representadas na figura 3.1: modelagem conceitual, projeto lógico e projeto físico.

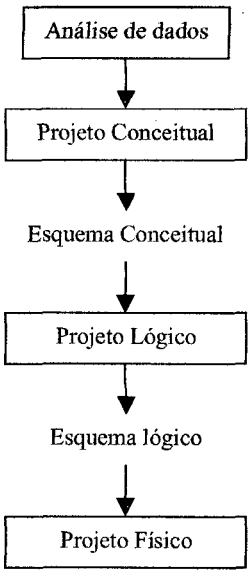


Figura 3.1 – Fases da modelagem dos dados

O que se espera com a execução do projeto é situar a seqüência das atividades a serem concluídas em uma ordem que possa resultar em ganhos de produtividade e confiabilidade do sistema, eliminando-se sensivelmente as falhas detectadas após sua implantação.

3.1 Modelagem Conceitual

Essa etapa consiste em examinar e compreender o ambiente a ser modelado, as exigências do banco de dados são determinadas entrevistando os usuários dos dados e produzindo uma especificação formal de exigências incluindo, requisitos de dados para o processamento, relacionamento natural entre os dados, plataforma de software para a implementação do banco de dados, análise das necessidades futuras e determinação das informações que resultarão na Modelagem Conceitual.

O Modelo Conceitual nada mais é que a descrição da realidade do ambiente do problema, essa descrição é independente das restrições de implementação. Este modelo constitui-se de uma visão global abstrata dos principais dados e relacionamentos, que podem aparecer na estrutura do banco de dados, conhecida como macrodefinição.

Espera-se que o modelo a ser utilizado para a definição da modelagem conceitual seja capaz de modelar qualquer realidade, ter uma forma de trabalho bastante simples e deve ter características gráficas que sejam bastante simples de construir e entender. A técnica de representação mais conhecida e largamente utilizada dessa modelagem é o Modelo Entidade-Relacionamento (MER), criado em 1976 por Peter Chen, que teve por base a teoria dos conjuntos e a teoria relacional criada por Edward Codd em 1970.

O MER possui várias notações gráficas representadas através de um Diagrama Entidade-Relacionamento (DER). Existem muitas variações desse diagrama e nenhum padrão prevalece aceito totalmente, apesar do mais utilizado ser o proposto por Peter Chen.

Para uma melhor compreensão deste modelo, o capítulo apresenta o diagrama idealizado por Peter Chen e outras propostas mais contemporâneas utilizada também por diversos autores como Ramakrisnan e Gehrke, Elsmari e Navathe, e etc.

3.1.1 Modelo entidade-relacionamento por Peter Chen

[Heuser, 1998] apresenta conceitos básicos e avançados do modelo proposto por Peter Chen, como entidade, atributos, relacionamentos, cardinalidade, generalização/especialização e agregação.

Entidade: esse conceito representa um objeto ou um conjunto de objeto da realidade modelada, é representada graficamente através de um retângulo que contém o nome da entidade, ex.:

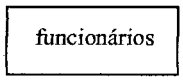


Figura 3.2 - Entidade

O exemplo mencionado designa o conjunto de todos os funcionários sobre os quais se deseja manter informações.

Atributos: todo objeto para ser uma entidade possui propriedades que são descritas por atributos que descrevem as instâncias ou ocorrências de uma entidade, conforme a próxima ilustração.

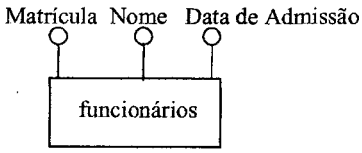


Figura 3.3 - Atributos

Com a figura percebe-se que o funcionário possui um número de matrícula, um nome e a sua data de admissão, esses dados que caracterizam o objeto funcionário são os atributos inerentes à entidade funcionário.

Relacionamento: é uma propriedade das informações mantidas sobre cada entidade, onde o foco é a associação entre os objetos. Exemplificando, pode ser necessário saber quais os funcionários trabalham em quais departamentos, essa informação pode ser representada da seguinte forma:

Figura 3.4 - Relacionamento

Na representação o relacionamento é identificado através de um losango, ligado aos retângulos representativos das entidades por linhas. O modelo expressa que o banco de dados mantém informações sobre um conjunto de objetos classificados como funcionários, um conjunto de objetos classificados como departamentos e um conjunto de associações, que ligam um departamento a um funcionário.

Assim como no caso das entidades, quando se fizer necessário apontar uma associação em particular dentro do conjunto, refere-se a *ocorrência* de relacionamento.

Do conceito de relacionamento deriva muitos outros conceitos, como o de relacionamento entre múltiplas entidades, relacionamentos múltiplos muitos-para-muitos, relacionamentos um-para-muitos, relacionamentos um-para-um, e a propriedade de relacionamento entre a mesma entidade ocasionando um outro conceito conhecido como auto-relacionamento.

Cardinalidade: é a ocorrência de entidades em um dado relacionamento. No exemplo do departamento e dos funcionários, podemos concluir que um departamento pode ter vários funcionários e um funcionários está lotado em apenas um departamento. A notação dos relacionamentos n:n (muitos-para-muitos), 1:n (um-para-muitos) e 1:1 (um-para-um), é representados por:

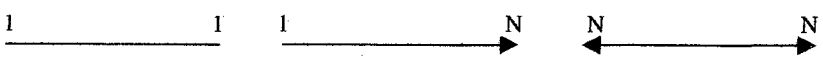


Figura 3.5 - Cardinalidade

Ainda em [Heuser, 1998] algumas extensões do Modelo ER são apresentadas, como o conceito de generalização/especialização e agregação.

Os conceitos descritos podem gerar outras especificações como **generalização/especialização**, que é um conjunto de propriedades particulares que acontecem em algumas ocorrências de uma entidade, representadas na figura 3.6.

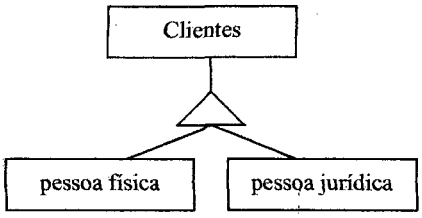


Figura 3.6 – Generalização/especialização

Analisando superficialmente, pode-se definir entidades para cada uma destas classes, ao invés disso, através da colocação de um atributo qualificador, permitiu-se a distinção entre cada classe de dados, generalizando todas estas classes em uma única.

A esta qualificação por atributos que permite identificar um grupo, uma classe dentro da classe genérica, denomina-se de especialização.

A generalização/especialização pode ser de dois tipos: *total* ou *parcial*. Em uma generalização/especialização total para cada ocorrência da entidade genérica existe sempre uma ocorrência em uma das entidades especializadas. Esse tipo de generalização/especialização é simbolizado por um "t", caso contrário é parcial, simbolizado por um "p".

A **agregação** é uma forma de construir um relacionamento envolvendo algum outro relacionamento e não uma entidade.

3.1.2 Diagrama entidade-relacionamento pela notação Merise

Na Europa há uma proposta de metodologia de desenvolvimento de sistemas conhecida por Notação Merise, utilizada principalmente na França, encontrada em muitos livros de autores europeus, como Ramakrisnan e Gehrke, Elmasri e Navathe.

As principais diferenças, desta notação para a de Peter Chen estão na representação gráfica do relacionamento no DER na qual é representado por uma elipse e a cardinalidade é invertida com relação à demonstrada por Peter Chen.

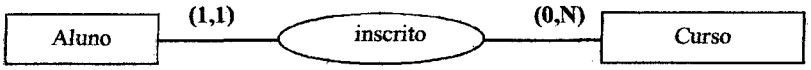


Figura 3.7 - DER conforme notação Merise

3.1.3 Diagrama entidade-relacionamento pela notação da Engenharia da Informação

A engenharia da informação é um método proposto no início da década de 80 por Martin e Finkelstein. A característica que o distingue de outros métodos é a ênfase que dá à modelagem de dados. Essa notação também é conhecida como James Martin.

Nesta notação os relacionamentos não são representados graficamente e a cardinalidade é representada através dos seguintes símbolos:

Cardinalidade Mínima 0: ○

Cardinalidade Máxima N: 

Cardinalidade (*mínima*, máxima) 1: |

Figura 3.8 - DER conforme notação da Engenharia da Informação

3.1.4 Diagrama entidade-relacionamento por Elmasri e Navathe

Esta notação é mais detalhada na representação de atributos no DER do que a notação convencionada por Peter Chen. Ela representa os atributos dentro de elipses no DER. Veja o exemplo da figura 3.9.

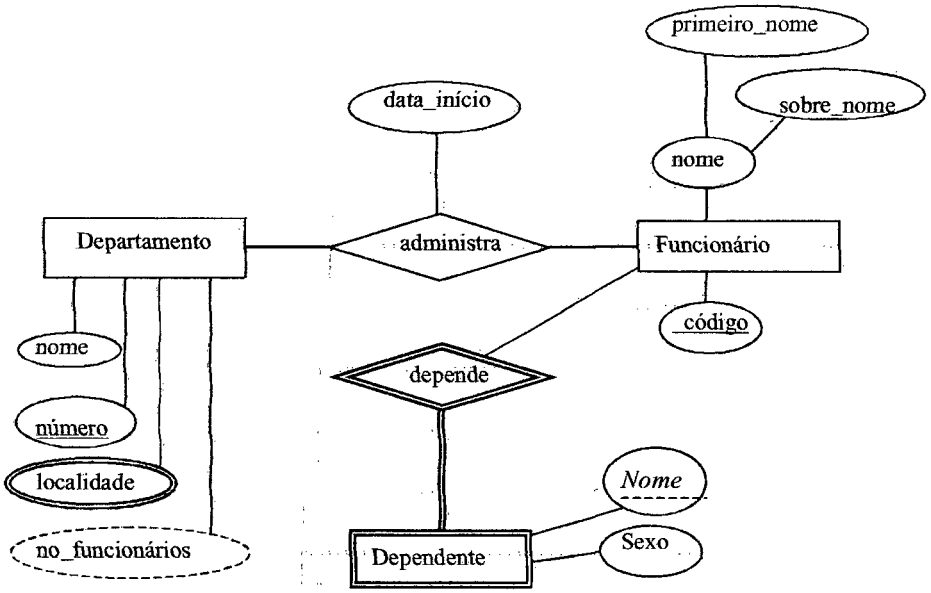


Figura 3.9 - Representação pela notação de Elmasri & Navathe

No exemplo, nome é um atributo composto por dois outros atributos (primeiro_nome e sobre_nome), código é um atributo chave (sublinhado dentro da elipse) ou um atributo identificador, ambos fazem parte do jogo de entidade Funcionário.

No jogo de entidades "Departamento" o atributo no_funcionários é um atributo derivado (elipse tracejada) ou virtual, pois não é armazenado fisicamente sendo obtido como resultado de uma operação. Localidade é um atributo multivalorado (elipse com contorno duplo) pois aceita múltiplos tipos de valores.

No jogo de entidades "Dependente" o atributo nome é parte da chave (sublinhado de forma tracejado dentro da elipse), através do relacionamento identificador depende a outra parte da chave (o atributo código) que é herdada do jogo de entidade Funcionário.

No jogo de relacionamentos "Administra" o atributo data_início é um atributo descritivo do relacionamento.

O *relacionamento identificador* é representado através do jogo de relacionamentos *depende* no qual o losango tem contorno duplo e a linha que o liga ao jogo de entidades dependente é dupla.

A *entidade fraca* é representada através do jogo de entidades *Dependente* na qual o retângulo tem contorno duplo.

A tabela abaixo mostra as diferenças entre as notações descritas.

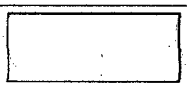
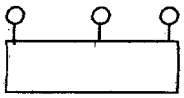
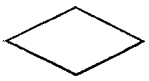
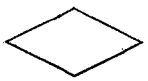
Conceitos	**Notação por Peter Chen**	**Notação Merise**	**Notação da Engenharia da Informação**	**Notação por Elmasri e Navathe**
Entidade				
Atributo				
Relacionamento				
Cardinalidade	1 —— 1 1 ——▶ N N ◀——▶ N	(1,1), (0,N), (N,N)	Mínima: ○ Máxima: ⟩— (Mínima,máxima): |	

Tabela 3.1 – Demonstrativo das representações do DER por várias notações

3.2 Projeto Lógico

Após a descrição do Modelo Conceitual tem-se início a construção do projeto lógico, que descreve as estruturas contidas na modelagem conceitual considerando ainda as características específicas do SGBD adotado. Nesta etapa o esquema conceitual global será transformado em um esquema específico, que mostrará todos os dados e seus relacionamentos. A metodologia de desenvolvimento do esquema global é o mesmo para um banco de dados distribuído ou centralizado [Kern, 1994].

Para que um banco de dados, de qualquer tipo, seja eficaz, sua estrutura lógica deve refletir o modo como os usuários vêem seu mundo. Este trabalho acontece através do entendimento do modelo do universo do usuário, uma vez modelados os requisitos dos usuários, estes devem ser transferidos para um projeto de banco de dados que reflita exatamente o modelo sugerido pelo usuário.

Os SBGDs existentes no mercado geralmente adotam o modelo hierárquico, de rede ou o modelo relacional. Dentre os três, o modelo relacional é, com larga margem o mais difundido. Para qualquer modelo adotado, entretanto, há uma condição fundamental para o sucesso de um sistema de banco de dados: se a etapa de modelagem lógica for inconsistente, o sistema todo estará comprometido.

A abordagem relacional está baseada no princípio de que as informações em uma base de dados podem ser consideradas como relações matemáticas e que estão representadas de maneira uniforme, através do uso de tabelas bidimensionais. O modelo relacional fornece dois formalismos para a manipulação de banco de dados: a álgebra relacional e o cálculo relacional.

A álgebra relacional consiste em operações cujos nomes vêm da teoria dos conjuntos, mais algumas operações relacionais especialmente criadas como: união, interseção, diferença, produto cartesiano, projeção, seleção, junção e divisão.

O cálculo relacional é um outro tipo de manipulação dos dados, derivado de um ramo da matemática chamada de cálculo predicado, ele é do tipo não-procedural, é uma linguagem para expressar o que se quer sem expressar como consegui-lo.

3.2.1 Principais vantagens da abordagem relacional

De acordo com [Machado e Abreu, 1995] a abordagem relacional apresenta diversas vantagens como independência total dos dados, visão múltipla dos dados, redução acentuada na atividade de desenvolvimento de aplicações e o tempo gasto em manutenção, melhoria na segurança dos dados e mais agilidade na questão gerencial da informação ligada ao processo decisório da organização.

Ao definir o modelo relacional, Edward Cood, estabeleceu um conjunto de doze regras para a determinação de um banco de dados ser realmente relacional. Se seis dessas regras forem cumpridas o SGBD pode ser qualificado como completo relacionalmente. Segundo estas regras, discute-se a fidelidade de um SGBD ao modelo relacional. A regras são:

1) Toda informação num banco de dados relacional é apresentada a nível lógico por valores em tabelas;

2) Todo dado em um banco de dados relacional tem a garantia de ser logicamente acessível, recorrendo-se a uma combinação do nome da tabela, um valor de chave e o nome da coluna;

3) Tratamento sistemático de valores nulos;

4) O dicionário de dados relacional ativo é baseado no modelo relacional;

5) O SGBD relacional deve ter uma linguagem para definição, detalhamento e manipulação dos dados;

6) Tratamento das atualizações de visões dos dados;

7) Tratamento de alto nível para inserção, atualização e eliminação de dados;

8) Independência dos dados físicos (mudanças na memória e no método de acesso);

9) Independência de dados lógicos (mudanças de qualquer tipo nas tabelas básicas, ex.: divisão de uma tabela por linha ou coluna);

10) Independência das restrições de integridade;

11) Independência de distribuição;

12) Não subversão das regras de integridade ou restrições quando se utiliza uma linguagem de baixo nível.

3.2.2 Derivação do modelo ER para o projeto lógico relacional

Após a criação do MER o desenvolvedor irá passar as visões do modelo conceitual para o modelo lógico relacional, no qual os dados são vistos como estruturas de dados voltados para as características do modelo lógico relacional, visando a implementação do banco de dados.

Existem regras precisas para a derivação do modelo ER para o projeto lógico relacional que não dão margem a erro, uma dessas regras é o mapeamento de entidades, ou seja, toda entidade torna-se uma tabela carregando todos os atributos definidos. O mapeamento dos atributos, onde os atributos das entidades e relacionamentos devem ser gerados na ordem que minimize o consumo de espaço de armazenamento e torne mais eficiente a recuperação.

O mapeamento dos relacionamentos, onde os relacionamentos entre as entidades dão origem a chave primária da entidade na própria entidade e a chave estrangeira na entidade que se relaciona, gerando uma estrutura de acesso a partir desta chave, entre outras.

Uma vez projetado o modelo ER seguido as regras comentadas, as tabelas que representam o MER num nível mais baixo podem ser obtidas de forma clara. É evidente que nesta passagem devem ser observadas as características do SGBD que será utilizado, pois existem diferenças que podem auxiliar ou dificultar algumas soluções propostas, alguns SBGDs necessitam de adaptações específicas.

3.3 Normalização

[Heuser, 1998] explica que o conceito de normalização também foi introduzido por Edward Codd em 1970. Essa técnica é um processo matemático formal que tem seus fundamentos na teoria dos conjuntos, é composta pela Primeira Forma Normal (1FN), Segunda Forma Normal (2FN), Terceira Forma Normal (2FN), Quarta Forma Normal (QFN) e finalmente a Quinta Forma Normal (5FN)

Através deste processo pode-se, gradativamente, substituir um conjunto de entidades e relacionamentos por um outro, o qual se apresenta "purificado" em relação às anomalias de atualização (inclusão, alteração e exclusão) que podem causar certos problemas, tais como: grupos repetitivos de dados, dependências parciais em relação a uma chave concatenada, redundâncias de dados desnecessários, perdas acidentais de informação, dificuldade na representação de fatos da realidade observada e dependências transitivas entre atributos.

Observa-se em alguns casos que o alto grau de normalização torna-se cada vez mais restrito, e dependendo do SGBD utilizado, essas restrições podem se tornar benéficas ou não.

[Heuser, 1998] comenta ainda que alguns estudos fomentaram o debate sobre as semânticas da normalização, sua utilidade e facilidade em relação à implementação física, chegando inclusive a utilização de banco de dados não normalizados por apresentarem maior ligação com a realidade e por terem vínculos matemáticos mais amenizados.

3.4 Projeto Físico

Após as etapas anteriores é gerado uma estrutura física do banco de dados, chamada de projeto físico, que envolve a seleção de índices e agrupamento de dados. O projeto lógico irá simplificar o ambiente para o projeto do banco de dados reduzindo o número de dependência dos dados.

O objetivo destes passos são o de representação precisa da realidade, onde é preservada a integridade de dados por normalização das tabelas criadas quando o modelo de ER é transformado em um modelo de relacional, otimizando a performance do sistema.

A partir do esquema específico criado na fase anterior, o modelo físico irá descrever as estruturas físicas de armazenamento e acesso aos dados de acordo com as características e os recursos do SGBD.

3.5 Projeto de Distribuição dos Dados

O projeto da distribuição de um banco de dados tem como objetivo gerar esquemas conceituais locais a partir de um esquema conceitual global, que é obtido através das etapas de um projeto tradicional, [Özsu e Valduriez, 1999].

Dentro dessa etapa de distribuição e dependendo da abordagem adotada estão as fases que tratam da fragmentação de um esquema global, alocação dos dados fragmentados e integração de dados já distribuídos.

O projeto deve ser realizado de maneira que o SGBDD apresente um desempenho ótimo ou próximo do ótimo ao realizar consultas sobre o BDD resultante. Medidas de desempenho normalmente utilizadas são o custo de comunicação ou o tempo de resposta do sistema.

Além do esquema global, outros dados utilizados como entrada para o projeto são os padrões de acesso das aplicações aos dados e a topologia do sistema distribuído.

[Ferreira e Finger, 200] comentam sobre a escolha em relação a quando realizar esta tarefa, visto que está sujeita a argumentação quanto aos detalhes sobre a implementação e a estimativa da performance.

3.6 Considerações Finais

Neste capítulo foram abordados os conceitos sobre projeto de banco de dados, modelagem dos dados, sendo apresentadas para isso algumas notações propostas por Peter Chen, Ramakrisnan e Gehrke, Elsmari e Navathe, Martin e Finkelstein, logo a seguir foram tratados o projeto lógico, físico e de distribuição.

No próximo capítulo serão tratadas as considerações sobre o caso do projeto de banco de dados distribuído.

Capítulo 4

PROJETO DE BANCO DE DADOS DISTRIBUÍDOS

Neste capítulo será apresentado as decisões que acrescem o projeto de banco de dados centralizado para incorporar o projeto de banco de dados distribuído [Kort e Silberschatz, 1995]. É importante frisar que neste estudo as técnicas, metodologias e ferramentas tratam os dados estaticamente, mas nem sempre a necessidade é essa, benefícios oriundos do dinamismo como a flexibilidade, disponibilidade e portabilidade, exigem que as bases de dados estejam preparadas para atender essa recente modalidade.

A seção a seguir traz uma introdução sobre o tratamento de dados dinamicamente, o resto do trabalho permanece abordando o ambiente estático.

4.1 Ambiente Dinâmico

Em um ambiente dinâmico, é comum projetar e reprojetar o BD. O reprojeto pode ser limitado apenas nas partes afetadas por mudanças, ou ser totalmente refeito, requerendo uma completa redistribuição, ao contrário do que ocorre no ambiente estático onde o projeto é realizado apenas uma vez e depois apenas administrado.

Aspectos físicos como característica de rede, disponibilidade de armazenamento nos sites e aspectos lógicos como migração da aplicação de um site para outro e acesso padrão passam por mudanças que obrigam a reescrever o projeto.

A Universidade de Trondheim está trabalhando com um sistema conhecido por Tetranode que trata de ambientes com dados dinâmicos, utilizando o simulador DBSIM, a performance apresentada até agora tem se apresentado promissora.

4.2 Classificação das Alternativas para a Distribuição de Dados

[Özsu e Valduriez, 1999] apresentam uma taxonomia que classifica as possíveis alternativas para a distribuição de dados, conforme apresentado na figura abaixo, a distribuição dos dados pode acontecer ao longo de três dimensões ortogonais, e os problemas surgidos são considerados dentro dessa amplitude:

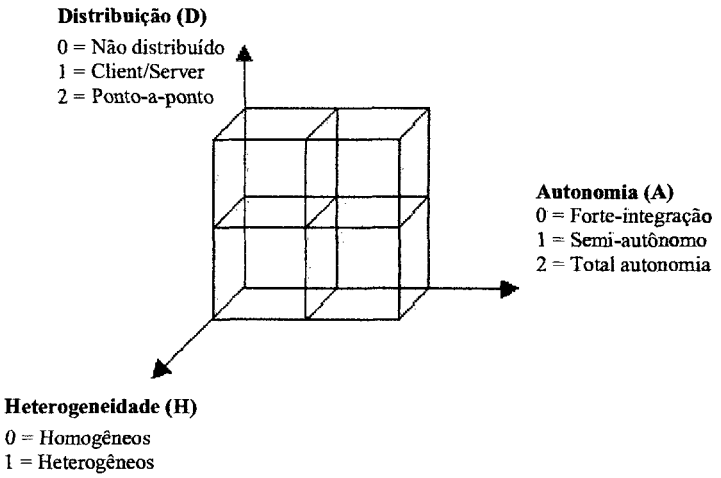


Figura 4.1 – Quadro de distribuição do problema

A primeira dimensão a ser considerada é em relação ao nível de distribuição, três possibilidades são discutidas:

- *Não distribuído*: cada aplicação e seus dados são executados em um site, e não há a comunicação com qualquer outro programa ou acesso a dados em outros site;

- *Client/Server*: todos os programas são replicados em todos os sites, mas os arquivos de dados não são. Conseqüentemente, requisições dos usuários são manuseadas nos sites que originaram as consultas e necessariamente os arquivos de dados são movidos em volta da rede;

- *Ponto-a-ponto*: nesta possibilidade, dados e programas serão compartilhados, sendo que cada site pode requerer um serviço de outro programa em um segundo site, que, novamente pode ter acesso a arquivos de dados locais de um terceiro site.

A segunda dimensão sugerida é sobre autonomia, seu conceito 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 dos dados.

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.

Considera-se a existência de três tipos de autonomia: forte integração, semi-autonomia e total autonomia (ou total isolamento), que correspondem, respectivamente, a três tipos de sistemas:

- *Sistemas fortemente integrados*: uma visão única de toda a base de dados é disponibilizada aos usuários que querem compartilhar as informações que podem estar fisicamente armazenadas em múltiplas bases de dados;

- *Sistemas semi-autônomos*: consistem de SGBDs que podem operar independentemente, mas que participam de um conjunto de sistemas para permitir o compartilhamento de seus dados;

- *Sistemas totalmente isolados:* os componentes individuais são SGBDs stand alone que desconhecem a existência de outros SGBDs, bem como a maneira de se comunicar com eles.

Entretanto, entre nenhuma autonomia e total autonomia provavelmente existem muitos níveis intermediários, devendo-se delineá-los e defini-los precisamente, identificando-se o grau de compartilhamento de dados existentes em cada um deles e discutindo-se modelos de transação apropriados para cada um dos níveis de autonomia delineados.

A última questão é sobre a heterogeneidade, pode-se dizer que esta é uma 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. Os dados podem ter diferentes interpretações dentro de uma organização. As diferenças apontadas por [Özsu e Valduriez, 1999] em relação a dados compartilhados e dados + programas compartilhados, levam a ambientes de distribuição distintos, o homogêneo e o heterogêneo.

4.3 Sistemas Homogêneos

Sistemas de banco 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 sistemas operacionais forem diferentes, conforme pode ser visto na figura seguinte. Estes sistemas são semelhantes aos sistemas centralizados, porém, em vez de armazenar os dados em um único site, eles serão distribuídos em vários sites da rede.

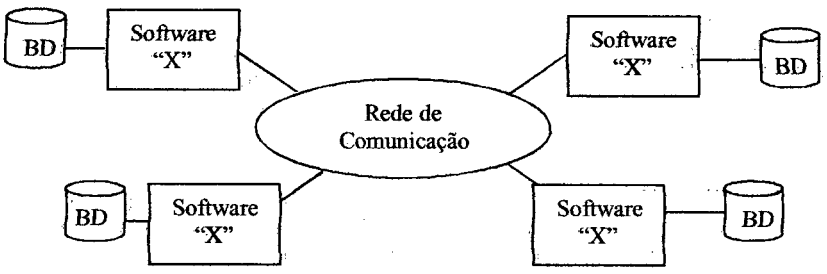


Figura 4.2 - Banco de dados distribuído homogêneo

Os SGBDDs homogêneos podem ser divididos conforme sua autonomia nas três classes anteriormente citadas: autônomos, não-autônomos e semi-autônomos.

Em um SGBDD homogêneo autônomo existem usuários locais e globais, sendo que cada banco de dados local desconhece outros bancos de dados, 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, existindo apenas 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.

4.4 Sistemas Heterogêneos

Por sua vez os sistemas heterogêneos são compostos por uma variedade de sistemas de banco de dados preexistentes, localizados em vários ambientes heterogêneos de hardware e software.

O acesso e a manipulação de dados originários de banco de dados preexistentes e heterogêneos que se encontram espalhados pela rede, requer um software específico que gerencie essas informações e forneça interoperabilidade entre os banco de dados locais, este tipo de sistema é chamado de Sistema Gerenciador de Banco de Dados Distribuídos Heterogêneos (SGBDDH).

Considera-se um BDD heterogêneo [Özsu e Valduriez, 1999] aquele que usa pelo menos dois tipos de SGBDs diferentes, confira na figura 4.3. Portanto, SGBDH fornece transparência não só da distribuição dos dados, mas também dos diferentes sistemas que o usuário acessa.

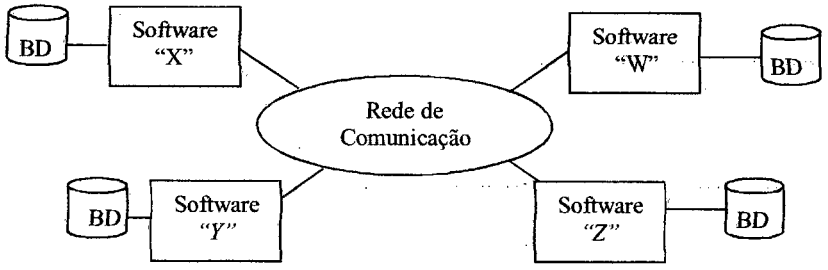


Figura 4.3 - Banco de dados distribuído heterogêneo

De acordo com [Mânica, 2001] 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 modelo entidade-relacionamento apresentado no capítulo 3.

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].

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.

As principais dificuldades encontradas no modelo de banco de dados heterogêneos são: na definição de um modelo de integração, assunto tratado no capítulo 7, no processamento e otimização de consultas, na gerência e controle da concorrência de transações e na interoperabilidade entre os sistemas [Özsu e Valduriez, 1999].

4.5 Estratégias de Projeto de Banco de Dados Distribuídos

Duas estratégias são identificadas [Ceri e Pelagatti, 1985] para projetar um banco de dados distribuídos. A abordagem *Top-Down* e a *Bottom-up*.

Em ambos os casos de estratégias de distribuição, é importante ressaltar que a distribuição de dados deve atingir seus objetivos para que em qualquer dos ambientes as informações sejam providas corretamente.

4.6 Abordagem *Top-down*

A abordagem *Top-down*, acontece quando o projeto do sistema ocorre a partir do zero, normalmente em sistemas homogêneos. Essa idéia é apresentada na figura 4.4.

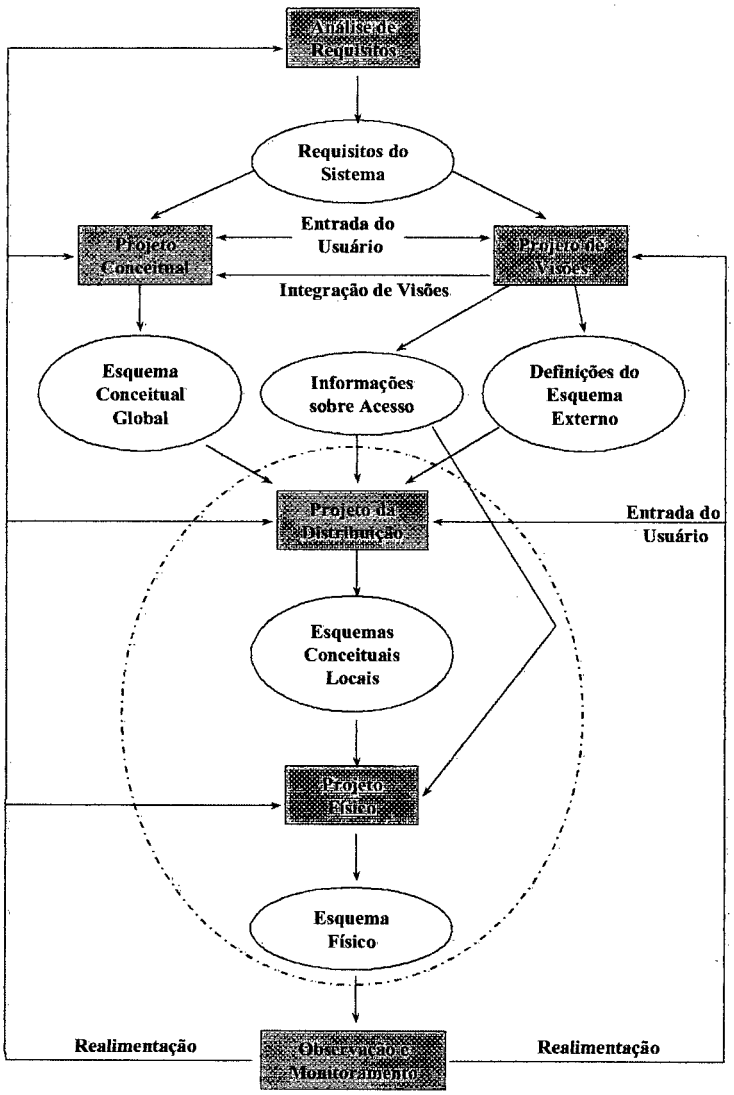


Figura 4.4 – Projeto para abordagem top-down

A atividade começa com a análise de requisitos que define o ambiente e o objetivo do sistema, extraindo respostas dos dados e do processamento necessário para os usuários em potencial do banco de dados. [Yao, et al., 1982].

Os objetivos são divididos em duas atividades paralelas: o projeto conceitual e o projeto de visões.

O projeto conceitual examina as entidades e os relacionamentos entre elas. Essa atividade de acordo com [Davenport, 1981] inclui a análise de entidade e a análise funcional. A análise de entidade está concentrada em determinar as entidades, seus atributos, e o relacionamento entre eles. A análise funcional por sua vez, determina a função principal desta para o modelo.

O projeto de visões é uma atividade que modela conceitualmente as aplicações existentes e as futuras aplicações. Essa integração permite que requisitos das entidades e dos relacionamentos sejam abrangidos no esquema conceitual global. Existe um relacionamento entre o projeto conceitual e o projeto de visões. O projeto conceitual inicia a integração com as consultas dos usuários.

O trabalho realizado até então segue o processo idêntico ao projeto de banco de dados centralizado. O esquema conceitual global e as informações de acesso que são resultados do projeto de visões são os primeiros passos no projeto da distribuição.

A seguir é necessário projetar o esquema conceitual global distribuindo entidades sobre os sites da rede formando o esquema conceitual local.

Os relacionamentos distribuídos são comumente chamados de fragmentos. Esse projeto de distribuição consiste em dois passos: o particionamento e a alocação das partições.

4.6.1. Particionamento dos dados

Em [Ceri e Pelagatti, 1985] o particionamento consiste em subdividir a relação global determinando as "unidades lógicas de armazenamento".

Tuplas ou atributos individuais não são considerados unidades apropriadas. Três técnicas podem ser utilizadas nessa fase: a fragmentação horizontal, vertical e híbrida.

A fragmentação horizontal faz o agrupamento de tuplas, enquanto que a fragmentação vertical se encarrega dos atributos. Já a fragmentação híbrida é uma combinação dos dois tipos, a constituição dos agrupamentos garante que tuplas ou atributos tenham as mesmas propriedades.

Essa etapa envolve fazer decisões sobre como fragmentar, qual é uma unidade razoável de distribuição, quanto fragmentar, como testar a fragmentação, quando fragmentar e etc.

Discute-se em [Mesquita e Finger, 1998] sobre quando começar a tratar da fragmentação de um esquema global e alocação dos dados fragmentados. A idéia de iniciar a distribuição na fase de projeto lógico deixa como conseqüência a desatualização do modelo obtido no nível conceitual, sendo que este não mais será considerado como documentação do projeto.

A proposta de elevação do projeto de distribuição para o nível conceitual utilizando o Modelo Entidade-Relacionamento, suas extensões e variações notacionais com suas devidas propostas de alterações estão em [Mesquita, 1998]. O capítulo 5 irá tratar estes assuntos mais detalhadamente.

4.6.2 Alocação dos dados

Assumindo que um banco de dados foi fragmentado adequadamente, dá se início o projeto de alocação dos fragmentos onde é necessário decidir o local de armazenamento desses fragmentos pelos vários sites da rede.

Em [Özsu e Valduriez, 1999] o conceito formal do problema da alocação de banco de dados envolve procurar a distribuição ótima dos fragmentos através dos sites, ou seja, considere que há um conjunto de fragmentos $F = \{F_1, F_2, ..., F_n\}$ e uma rede formada pelos sites $S = \{S_1, S_2, ..., S_m\}$ no qual o conjunto de aplicações $Q = \{q_1, q_2, ..., q_q\}$ é executado, a questão seria distribuir F em S.

O caminho mais fácil seria o de considerar cada fragmento como um arquivo separado, entretanto, essa idéia não é conveniente, pois os fragmentos não são modelados como arquivos individuais, a quantidade de fragmentos em comparação com a relação global é muito diferente, por isso vários modelos analíticos podem não computar a solução e o comportamento de um sistema distribuído no uso dos dados é muito mais sofisticado.

Quando da alocação dos dados o projeto pode acontecer de duas formas: com ou sem redundância. Na opção sem redundância há apenas uma simples cópia do banco de dados, como a encontrada no modelo centralizado. Na alternativa com redundância, a relação é particionada em diversos fragmentos, armazenados em sites diferentes.

A distinção entre essas duas técnicas do projeto de distribuição é conceitualmente relevante, o primeiro é visto como um critério lógico, enquanto a alocação é uma questão física de distribuição dos fragmentos entre os sites. O capítulo 6 abordará vários aspectos ligados a alocação dos dados.

4.6.3 Objetivos do projeto de distribuição de dados

De acordo com [Ceri e Pelagatti, 1985] no projeto da distribuição dos dados, alguns objetivos devem ser alcançados, como proporcionar a maximização do processamento nos sites locais, isto pode ser possível considerando o número de referências locais e remotas correspondentes a cada fragmento, e selecionando a melhor solução entre elas. O processamento local não serve apenas para reduzir o acesso a dados remotos, mas também para aumentar a simplicidade no controle de execução das aplicações.

A disponibilidade e a confiabilidade na distribuição dos dados é outro fator importante, visto que o alto grau de disponibilidade proporciona a utilização de outros fragmentos enquanto alguns estão sendo utilizados por outras aplicações.

A confiabilidade é a garantia de que mesmo em caso de sinistros como fogo, terremoto ou sabotagem, as informações estarão disponíveis em outros locais geograficamente dispersos.

O processo de distribuição de dados permite a distribuição da carga de trabalho em torno dos sites, maximizando o grau de paralelismo na execução de aplicações.

O custo de armazenamento e disponibilidade dos dados, deve ser controlado pois várias informações são disponibilizadas em locais de armazenamento diferentes. Características como limitação da disponibilidade de armazenamento local, custo de transmissão da aplicação, entre outras, devem ser avaliadas e algum critério para prover a distribuição deve ser utilizado no início do projeto e depois outros critérios podem ir sendo introduzidos para a otimização.

4.7 Abordagem *Bottom-up*

A abordagem *Bottom-up* ocorre quando da integração de bancos de dados já existentes, que funcionavam como unidades independentes, sendo necessário integrá-las produzindo uma especificação global [Ferreira e Finger, 2000].

O crescimento desses bancos de dados não visavam uma futura integração com outros sistemas, apenas surgiram e evoluíram para atender necessidades características de seu ambiente [Ceri e Pelagatti, 1985].

Essa evolução reforçou a necessidade de interligar unidades isoladas, preservando os esquemas locais existentes e criando um esquema global, promovendo a fusão da definição dos dados e resolvendo conflitos surgidos entre diferentes representações de alguns dados.

Normalmente, os dados a serem integrados 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 de forma que preserve os investimentos feitos nas aplicações locais.

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).

4.7.1 Integração de banco de dados

O ponto de partida do projeto de integração dos dados inicia-se nos esquemas que representam a porção de dados armazenada em cada local, e o projeto de integração consiste em identificar os dados comuns a cada esquema, bem como suas diferenças.

Os problemas do projeto de distribuição segundo a abordagem *bottom-up* estão na construção do esquema global dos dados, com os seguintes passos para construí-lo: reconhecer similaridades, reconhecer conflitos de nome, diferenças de domínio, de escala, de estruturas e o tratamento da inconsistência dos dados durante as operações. Esses aspectos serão abordados com mais detalhes no capítulo 7.

4.7.2 Objetivos do projeto de integração dos dados

É importante que o projeto de integração seja desenvolvido de forma que preserve as características de cada sistema autônomo.

Outro aspecto importante é que a integração não viole a 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.

A próxima tabela resume os aspectos das duas abordagens distintas na qual um banco de dados distribuídos pode ser considerado.

Aspectos	Homogêneo	Heterogêneo
Abordagem do BDD	*Top-down*	*Bottom-up*
Técnicas utilizadas	Particionamento e alocação	Integração
Camada do SGBD	SGBDD	SGBDDH
Autonomia	Acoplamento Forte	Acoplamento Controlado
Heterogeneidade	Hardware, Software e Rede	SGBD, Hardware, Software e Rede
Transparência	Total	Depende do esquema de integração
Usuários	Globais	Globais e locais

Tabela 4.1 – Aspectos dos ambientes heterogêneo e homogêneo

4.8 Considerações Finais

Neste capítulo foram abordados detalhes sobre projeto de banco de dados distribuídos. Foram tratados os objetivos do projeto de distribuição de dados, as abordagens para o projeto de banco de dados distribuídos e suas respectivas técnicas, que incluem o projeto de esquema global, fragmentação, alocação e o caso da integração.

Nos próximos capítulos as técnicas utilizadas segundo cada abordagem serão revistas detalhadamente.

Capítulo 5

PARTICIONAMENTO DOS DADOS

A partição da relação global dá vida a diversos fragmentos, que são armazenados em diferentes nós, e podem ser definidos por expressões da linguagem relacional [Ceri e Pelagatti, 1985].

Esses fragmentos de relações (sub-relações), irão depender da execução concorrente de um número de transações que acessam diferentes porções de uma relação, sendo que as visões geradas de acordo com as necessidades dos aplicativos e que não podem ser definidas em um único fragmento, irão requerer processamento extra. Essas sub-relações exigem um rígido controle semântico dos dados, especialmente nas restrições de integridade.

Com respeito à fragmentação, uma importante questão é o tamanho apropriado das unidades de distribuição. Uma relação não é uma unidade adequada, por várias razões. Primeiro, visões de aplicação são geralmente sub-relações de relações. Então, a localidade do acesso da aplicação não é definida na relação inteira mas em sub-relações. Portanto é natural considerar sub-relações de relações de unidades distribuídas.

Segundo, se a aplicação que tem visões definidas de uma dada relação reside em sites diferentes, duas alternativas podem ser seguidas, uma com a relação inteira não replicada e armazenada em apenas um site, ou replicada em todos ou em quantos sites a aplicação residir.

O resultado anterior é desnecessariamente um grande volume de acesso a dados remotos. No outro a replicação tinha sido desnecessária, o que causa problemas na execução de atualizações e pode não ser desejável se o armazenamento for limitado.

Finalmente com a decomposição da relação em fragmentos, cada unidade pode ser tratada isoladamente, permitindo que um certo número de transações sejam executadas ao mesmo tempo.

A fragmentação da relação normalmente resulta em execução paralela de uma simples consulta por divisão dentro de sistemas de sub-consultas que operam em fragmentos. Esses fragmentos aumentam o nível de concorrência e o nível de transferência de dados [Bell e Grimson, 1992].

Nessa etapa algumas informações são requisitadas no momento da fragmentação, podemos citar por exemplo as informações advindas dos banco de dados, ou seja entidades, relacionamentos e atributos, das aplicações são retiradas informações sobre os predicados utilizados nas visões, das redes de comunicação extraem informações sobre a comunicação entre os sites e do sistema são retirados dados sobre armazenamento e processamento [Özsu e Valduriez, 1999].

5.1 Grau de Fragmentação

O grau da fragmentação é uma importante decisão que afeta a performance da execução de consulta. De fato, essa questão é um grande problema. Possui dois extremos, um fragmentar tudo, e o outro não fragmentar.

Há efeitos colaterais em qualquer das duas extremidades. Por isso é necessário procurar um nível de fragmentação compromissado com ambos os extremos.

Assim o nível pode ser definido com respeito à aplicação que vai rodar sobre ele [Ceri e Pelagatti, 1985]. Geralmente a aplicação precisa ser caracterizada com respeito ao número de parâmetros, e de acordo com eles os fragmentos individuais podem ser identificados.

5.2 Correção de Fragmentação

Segundo [Özsu e Valduriez, 1999] algumas regras são necessárias para garantir a consistência do banco de dados. Essas regras consistem em manter a completude, a reconstrutibilidade e a disjuntividade.

A ***completude*** trata de garantir que depois de decomposta uma relação R em fragmentos R_1, R_2,...,R_n, a mesma só será completa novamente se e somente se cada item de dado em R puder ser encontrado também em algum R_i.

A ***reconstrução*** consiste em prover algum operador relacional que ao ser aplicado aos fragmentos reconstruam as relações globais, esse operador pode ser diferente de acordo com a forma de fragmentação.

A ***disjuntividade*** não permite que um item de dado esteja presente em mais de um fragmento, ou seja se uma relação R é decomposta em fragmentos R_1, R_2,...,R_n, e um item de dado D_i está em R_j então D_i não deve estar em nenhum outro fragmento $R_k(k \neq j)$. As exceções a esta regra são os atributos chaves das relações, que devem estar presentes em todos os fragmentos verticais das mesmas a fim de permitir a reconstrução das relações globais.

5.3 Alternativas de Fragmentação

- **Fragmentação horizontal**: Divide a relação pela associação de tuplas da relação em um ou mais fragmentos, é subdividida em fragmentação horizontal primária e fragmentação horizontal derivada.

- **Fragmentação vertical**: Divide a relação pela decomposição de determinados atributos em diferentes fragmentos.

- **Fragmentação híbrida**: Cada fragmento é obtido como resultado da aplicação dos métodos para fragmentação horizontal e vertical na relação, ou em um fragmento da relação que foi previamente obtido.

Essas possibilidades podem ser aplicadas sucessivamente a uma mesma relação, resultando em diversos fragmentos.

5.4 Fragmentação Horizontal

Na fragmentação horizontal o particionamento acontece ao longo das tuplas. Em cada fragmento há um subconjunto de tuplas da relação original. Existem duas versões do particionamento horizontal: o primário e o derivado. A fragmentação horizontal primária acontece usando os predicados que são definidos na relação. A fragmentação derivada é o particionamento da fragmentação resultante dos predicados definidos na fragmentação horizontal primária.

Algumas informações adicionais sobre o banco de dados são exigidas nessa etapa, como o esquema conceitual global, alguns detalhes da aplicação também são exigidos, como informações prévias sobre a qualidade das informações que guiarão a atividade de fragmentação, e também a informação quantitativa que será incorporada dentro do modelo de alocação.

A informação qualitativa consiste dos predicados usados nas consultas. Se não for possível analisar todos, pelo menos os mais importantes deverão ser. É importante determinar os predicados simples e o conjunto de predicado simples conhecido como predicados minterm.

Dada uma relação R, onde $R(A_1, A_2, ..., A_n)$ e A_i é um atributo definido sobre o domínio D_i, um predicado simples p_j definido em R tem a forma:

$$p_j : A_i \ \theta \ \text{Valor}$$

Onde $\theta \in \{=, <, >, \neq, \geq, \leq\}$ e Valor pertence ao domínio do atributo A_i. Usa-se Pr_i para denotar o conjunto de todos os predicados simples definidos na relação R_i. Os membros de Pr_i são denotados por p_{ij}.

O conjunto de predicados simples é conhecido como predicado minterm. Onde dado um conjunto $Pr_i = \{p_{i1}, p_{i2}, ..., p_{in}\}$ de predicados simples de uma relação R_i, o conjunto de predicados minterm $M_i = \{m_{i1}, m_{i2}, ..., m_{iz}\}$ seja definido por:

$$M_i = \{m_{ij} \mid m_{ij} = \bigwedge_{p_{ik} \in Pr_i} p_{ik}^*\},\ 1 \le k \le m,\ 1 \le j \le z$$

Onde $p_{ik}^* = p_{ik}$ ou $p_{ik}^* = \neg p_{ik}$.

Assim cada predicado simples pode ocorrer em um predicado minterm em sua forma natural ou negada. É importante ressaltar que a forma negada do predicado simples deve ser tratada como complemento do mesmo.

Considere a relação *Conta* abaixo, o conjunto de predicados simples pode ser descrito da seguinte forma:

Código_Cliente	Nome_Cliente	Nome_agência	Número_conta	Saldo
01234	Leonora Farias	Paranavaí	065.987-2	200,00
02345	Humberto Oliveira	Paranavaí	034.867-0	150,00
03456	José Rodrigues	Maringá	076.231-5	460,00
04567	Jair Souza	Maringá	736.211-4	600,00
05678	Evangelista Bueno	Paranavaí	004.721-6	100,00
06789	Katarina Borges	Londrina	098.657-4	640,00
07890	Henrique Willeman	Londrina	034.829-7	120,00

Figura 5.1 - Relação *Conta*

p_1: Nome_Agência = "Paranavaí"

p_2: Saldo ≥ 250,00

O conjunto dos predicados simples $\{p_1, p_2\}$ origina o conjunto minterm M composto por $\{m_1, m_2, m_3$ e $m_4\}$, conforme demonstrado abaixo:

m_1: Nome_Agência = "Paranavaí" $\wedge$ Saldo $\geq$ 250,00

m_2: $\neg$ (Nome_Agência = "Paranavaí") $\wedge$ Saldo $\geq$ 250,00

m_3: Nome_Agência = "Paranavaí" $\wedge \neg$ (Saldo $\geq$ 250,00)

m_4: $\neg$ (Nome_Agência = "Paranavaí") $\wedge \neg$ (Saldo $\geq$ 250,00)

De acordo com [Özsu e Valduriez, 1999] outra informação relevante retirada da aplicação são os dados quantitativos sobre os usuários das aplicações. É necessário saber dois conjuntos de dados, a seletividade minterm e a freqüência de acesso.

A **Seletividade minterm** é a proporção de tuplas da relação que seria acessada por uma consulta a qual é especificada de acordo com um predicado minterm Mi, e é representada por sel(m_i), pode variar de 0 a 1.

A **Freqüência de acesso** é a freqüência com a qual uma aplicação acessa os dados. Se Q = {q_1, q_2, ..., q_n} é o conjunto de consulta dos usuários, então $acc(q_i)$ indica a freqüência de acesso de uma consulta q_i em um dado período. Neste caso a freqüência de acesso de um predicado minterm também pode ser definida.

5.4.1 Fragmentação horizontal primária

A fragmentação horizontal primária é definida pela operação de seleção da relação proprietária do esquema do banco de dados, ou seja, dada uma relação R, ela é particionada em um número de subconjuntos R_1, R_2...., R_n, cada tupla da relação R deve pertencer a pelo menos um fragmento, de modo que a relação original possa ser reconstruída, se necessário.

Um fragmento pode ser definido como uma seleção sobre a relação global *R*. Isto é, usamos um predicado P_i para construir o fragmento R_i, conforme segue:

$$R_i = \sigma_{P_i}(R),\ 1 \leq i \leq n$$

Onde σ é o operador de seleção da Álgebra relacional. Note que se P_i estiver em forma normal conjuntiva, será um predicado minterm (m_i). Um conjunto de fragmentos horizontais também é conhecido como fragmentos minterm.

Suponha que a relação *R* seja a relação *Conta*. Essa relação poderá ser dividida em *n* fragmentos diferentes, cada um dos quais formado pelas tuplas de uma agência em particular. Se o banco possuir apenas duas agências – Paranavaí e Maringá – então haverá dois fragmentos:

$$Conta_1 = \sigma_{\text{nome\_agência} = \text{"Paranavaí"}}(Conta)$$

$$Conta_2 = \sigma_{\text{nome\_agência} = \text{"Maringá"}}(Conta)$$

A próxima figura mostra esses dois fragmentos. O fragmento $Conta_1$ será armazenado no site Paranavaí. O fragmento $Conta_2$ será armazenado no site Maringá.

Código_Cliente	Nome_Cliente	Nome_agência	Número_conta	Saldo
01234	Leonora Farias	Paranavaí	065.987-2	200,00
02345	Humberto Oliveira	Paranavaí	034.867-0	150,00
05678	Evangelista Bueno	Paranavaí	004.721-6	100,00

$Conta_1$

Código_Cliente	Nome_Cliente	Nome_agência	Número_conta	Saldo
03456	José Rodrigues	Maringá	076.231-5	460,00
04567	Jair Souza	Maringá	736.211-4	600,00

$Conta_2$

Figura 5.2 – Fragmentação horizontal primária da relação *Conta*

Para reconstruir a relação R adota-se a união de todos os fragmentos, isto é:

$$R = R_1 \cup R_2 \cup ... \cup R_n$$

Dois aspectos do predicado simples são considerados importantes, sua completeza e minimalidade. Um conjunto de predicado simples *Pr* é dito ser completo se e somente se tiver igualdade na probabilidade de acesso pela aplicação de qualquer tupla pertencente a qualquer fragmento minterm que seja definido de acordo com *Pr*.

No exemplo da fragmentação da relação *Conta* a divisão das tuplas aconteceu por agência, sendo assim os fragmentos serão acessados de forma igualitária..

A minimalidade do predicado simples consiste da relevância do predicado para determinar uma fragmentação. O algoritmo utilizado para gerar um conjunto de predicado completo e mínimo a partir do predicado simples é o **COM_MIN,** apresentado em [Özsu e Valduriez, 1999].

Dado um conjunto de predicados simples *Pr,* o algoritmo COM_MIN gera um conjunto de predicado completo e mínimo. A regra fundamental de completude e minimalidade garantem que uma relação será fragmentada em pelo menos duas partes que serão acessadas diferentemente por pelo menos uma aplicação.

O algoritmo inicia procurando um predicado que seja relevante e que particione uma dada relação. A iteração do laço do-until adiciona predicados ao conjunto, assegurando a minimalidade em cada passo. Então, ao fim o conjunto *Pr'* é tanto mínimo quanto completo.

O segundo passo no processo de projetar a fragmentação horizontal primária é derivar o conjunto de predicados minterm que pode ser definido pelos predicados do conjunto *Pr'*. Esses predicados minterm determinam os fragmentos que serão usados como candidatos no passo da alocação.

O terceiro passo é eliminar alguns fragmentos minterm que podem ser iguais. A eliminação é executada identificando quais minterm podem ser contraditórios ao conjunto de implicação *I*.

Por exemplo: se $Pr' = \{p_1, p_2\}$ então

$$p_1 : att = valor\_1$$

$$p_2 : att = valor\_2$$

e o domínio de att é {valor_1, valor_2}, é óbvio que *I* contém duas implicações:

$$i_1 : (att = valor\_1) \Rightarrow \neg (att = valor\_2)$$

$$i_2 : \neg (att = valor\_1) \Rightarrow (att = valor\_2)$$

Podemos definir quatro predicados minterm:

$$m_1 : \quad (att = valor\_1) \wedge \quad (att = valor\_2)$$

$$m_2 : \quad (att = valor\_1) \wedge \neg \ (att = valor\_2)$$

$$m_3 : \neg (att = valor\_1) \wedge \quad (att = valor\_2)$$

$$m_4 : \neg (att = valor\_1) \wedge \neg (att = valor\_2)$$

Neste caso o predicado minterm m_1 e m_4 são contraditórios para as implicações *I* e podem ser eliminados de M, o algoritmo responsável por essa eliminação é o Algoritmo **PHORIZONTAL.**

O algoritmo PHORIZONTAL exposto em [Özsu e Valduriez, 1999] aplica o algoritmo COM_MIN e elimina os predicados minterms contraditórios.

5.4.2 Fragmentação horizontal derivada

Dada uma relação proprietária R, uma relação membro S e os fragmentos R_1, ..., R_n de R, a fragmentação derivada de S em relação a R consiste na determinação de n fragmentos S_1, ..., S_n onde S_i , para $1 \leq i \leq n$, é obtido através da semijunção entre S e R_i pelo atributo de ligação entre as duas relações. Se uma relação possuir mais de uma proprietária, a decisão sobre a fragmentação a ser realizada deve ser baseada no benefício trazido por cada uma das fragmentações possíveis [Özsu e Valduriez, 1999].

Considerando a relação *Conta* da ilustração 5.1 como relação proprietária e a figura abaixo como relação membro, pode-se aplicar a fragmentação horizontal derivada para dividir em dois grupos de acordo com o valor do saldo, originando a figura 5.4.

$$Conta_a = \sigma_{Saldo \geq 150,00}(Conta)$$

$$Conta_b = \sigma_{Saldo < 150,00}(Conta)$$

Código_Cliente	Nome_Cliente	Nome_agência
01234	Leonora Farias	Paranavaí
02345	Humberto Oliveira	Paranavaí
03456	José Rodrigues	Maringá
04567	Jair Souza	Maringá
05678	Evangelista Bueno	Paranavaí
06789	Katarina Borges	Londrina
07890	Henrique Willeman	Londrina

Figura 5.3 - Relação membro

Código_Cliente	Nome_Cliente	Nome_agência
01234	Leonora Farias	Paranavaí
02345	Humberto Oliveira	Paranavaí
03456	José Rodrigues	Maringá
04567	Jair Souza	Maringá
06789	Katarina Borges	Londrina

$Conta_a$

Código_Cliente	Nome_Cliente	Nome_agência
05678	Evangelista Bueno	Paranavaí
07890	Henrique Willeman	Londrina

$Conta_b$

Figura 5.4 - Fragmentação horizontal derivada do fragmento *Conta*

Para fazer uma fragmentação horizontal derivada três coisas são necessárias: a relação proprietária, a relação membro, e o conjunto dos predicados de semijunção entre a relação proprietária e a membro.

No exemplo acima o atributo SALDO da relação proprietária exposta na figura 5.1 foi responsável pela divisão em grupos da relação membro através do predicado indicado.

Existe mais que uma possibilidade de fragmentação, a escolha é baseada em dois critérios: a fragmentação com características de melhor junção e a fragmentação usada em muitas aplicações.

5.4.3 Verificando a consistência na fragmentação horizontal

Os critérios para garantir a corretude da fragmentação também são aplicados neste caso. Em relação a completude no caso da fragmentação horizontal primária é baseada na seleção dos predicados usados. Se a seleção de predicados é completa, a fragmentação resultante também é completa. Desde que a base do algoritmo de fragmentação seja um conjunto completo e mínimo de predicados, Pr', a completeza é garantida desde que enganos não sejam cometidos quando da definição de Pr'.

No caso da fragmentação horizontal derivada a completude é um pouco mais difícil de definir. A dificuldade é devido ao fato de que o predicado determinante da fragmentação envolve duas relações.

A reconstrução de uma relação global através dos fragmentos é executada pelo operador união de ambos os tipos de fragmentação horizontal.

A disjuntividade é mais fácil estabelecer no caso de fragmentação primária do que derivada. Ela é garantida devido a determinação do predicado minterm serem mutuamente exclusivo.

5.5 Fragmentação vertical

O objetivo da fragmentação vertical é particionar uma relação em um conjunto de relações menores de forma que cada aplicação do usuário utilize o menor número de fragmentos possíveis [Özsu e Valduriez, 1999]. Neste contexto, uma fragmentação ótima é aquela que produz um esquema de fragmentação que minimiza o tempo de execução da aplicação nestes fragmentos, reduzindo o acesso da mesma a dados irrelevantes.

Cada fragmento R_i de R é definido por :

$$Ri = \prod_{Ri} (R)$$

Onde Π é o operador de projeção. A fragmentação deve ser feita de tal modo que possamos reconstruir a relação R a partir dos fragmentos por meio de uma junção natural:

$$R = R_1 \bowtie R_2 \bowtie R_3 \bowtie \ldots \bowtie R_n$$

Um modo de garantir que a relação R possa ser reconstruída é incluir os atributos da chave primária de R em cada um dos R_i. Mais genericamente, qualquer super-chave pode ser usada. Muitas vezes é conveniente acrescentar um atributo especial, chamado *identificador de tupla* ou *tupla_id*, no esquema R.

O valor de *tupla_id* serve como chave candidata incrementada ao esquema, e será incluída em cada um dos R_i. O endereçamento físico ou lógico para uma tupla pode ser usado como *tupla_id*, uma vez que cada tupla possui um único endereço.

Para ilustração da fragmentação vertical, consideremos o exemplo de uma empresa bancária conforme visto na figura 5.1 que incorpora o esquema:

Conta = (código_cliente, nome_cliente, nome_agência, número_conta, saldo)

A figura abaixo mostra a relação *Conta:*

Código_Cliente	Nome_Cliente	Nome_agência	Número_conta	Saldo
01234	Leonora Farias	Paranavaí	065.987-2	200,00
02345	Humberto Oliveira	Paranavaí	034.867-0	150,00
03456	José Rodrigues	Maringá	076.231-5	460,00
04567	Jair Souza	Maringá	736.211-4	600,00
05678	Evangelista Bueno	Paranavaí	004.721-6	100,00
06789	Katarina Borges	Londrina	098.657-4	640,00
07890	Henrique Willeman	Londrina	034.829-7	120,00

Figura 5.5 - Relação *Conta*

Na próxima figura identificamos a relação *Conta* mais os identificadores de tuplas:

Tupla_id	Código_Cliente	Nome_Cliente	Nome_agência	Número_conta	Saldo
1	01234	Leonora Farias	Paranavaí	065.987-2	200,00
2	02345	Humberto Oliveira	Paranavaí	034.867-0	150,00
3	03456	José Rodrigues	Maringá	076.231-5	460,00
4	04567	Jair Souza	Maringá	736.211-4	600,00
5	05678	Evangelista Bueno	Paranavaí	004.721-6	100,00
6	06789	Katarina Borges	Londrina	098.657-4	640,00
7	07890	Henrique Willeman	Londrina	034.829-7	120,00

Figura 5.6 - Relação *Conta* com identificador de tupla

Nessa outra ilustração apresentamos a decomposição vertical do esquema *Conta* $\cup$ {*identificador_tupla*} em:

$Conta_1$= (tupla_id, codigo_cliente, nome_cliente, nome_agência)

$Conta_2$= (tupla_id, número_conta, saldo)

Tupla_id	Código_Cliente	Nome_Cliente	Nome_agência
1	01234	Leonora Farias	Paranavaí
2	02345	Humberto Oliveira	Paranavaí
3	03456	José Rodrigues	Maringá
4	04567	Jair Souza	Maringá
5	05678	Evangelista Bueno	Paranavaí
6	06789	Katarina Borges	Londrina
7	07890	Henrique Willeman	Londrina

$Conta_1$

Tupla_id	Número_conta	Saldo
1	065.987-2	200,00
2	034.867-0	150,00
3	076.231-5	460,00
4	736.211-4	600,00
5	004.721-6	100,00
6	098.657-4	640,00
7	034.829-7	120,00

$Conta_2$

Figura 5.7 - Fragmentação vertical da relação *Conta* com identificador de tupla

As duas relações apresentadas resultam do cálculo:

$$\text{Conta}_1 = \prod_{\text{conta1}}(conta)$$

$$\text{Conta}_2 = \prod_{\text{conta2}}(conta)$$

Para reconstrução da relação *Conta* original a partir de seus fragmentos, faremos:

$$\prod_{conta}(Conta_1 \rhd\lhd Conta_2)$$

Note que a expressão entre parênteses acima, é uma forma especial da junção natural. O atributo de junção é a *tupla_id.* Embora o atributo *tupla_id* facilite a implementação do particionamento vertical, esse benefício pode não ser claro para os usuários, já que se trata de um artifício interno à implementação e viola a independência dos dados – que é uma das principais virtudes do modelo relacional.

Diz [Özsu e Valduriez, 1999] que a maior informação requisitada pela fragmentação vertical é em relação a aplicação, ou seja, quais atributos são usualmente acessados juntos, há uma necessidade de mensurar quão relacionados são os atributos.

A maior parte dos algoritmos para fragmentação vertical utiliza matrizes de afinidades de atributos que são construídas a partir de matrizes de utilização de atributos, sobre a freqüência de acesso [Navathe, Ceri e Wiederhold, 1984].

Considere $Q = \{q_1, q_2, ..., q_n\}$ como um conjunto de consultas de usuários que podem ser executadas na relação $R\{A_1, A_2, ..., A_n\}$. No caso da relação *Conta* algumas consultas do tipo selecionar nome e saldo dos correntistas, selecionar saldo e nome da agência, entre outros podem ser usadas para compor o conjunto das consultas. Então para cada consulta q_i e cada atributo A_j, associa-se um valor, demonstrado por $uso(q_i,A_j)$ e definido a seguir:

$$uso(q_i,A_j) = \begin{cases} 1 \text{ se o atributo } A_j \text{ é referenciado pela consulta } q_i \\ 0, \text{ caso contrário} \end{cases}$$

O vetor $uso(q_i, A_j)$ para cada aplicação é facilmente definido pelo projetista que conhece as aplicações que podem ser executadas no banco de dados.

	A_1	A_2	A_3	A_4
q_1	1	0	1	1
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

Figura 5.8 – Matriz de uso dos atributos

A matriz de uso dos atributos geralmente não é suficiente para formar a base de particionamento de atributo e fragmentação. Isto por que os valores não representam o peso da freqüência de aplicação.

A matriz de afinidade de atributos quantifica com que intensidade cada par de atributos de *R* é acessado conjuntamente. Mensurar a afinidade entre dois atributos A_i e A_j da relação $R(A_1, A_2, ..., A_n)$ com respeito ao conjunto de aplicações $Q = \{q_1, q_2,...,q_n\}$ é definido por:

$$aff(A_i, A_j) = \sum_{k|use(qk,Ai)=1 \wedge use(qk,Aj)=1} \sum ref_i(q_k)\ acc_i(q_k)$$

Onde $ref_i(q_k)$ é o número de acesso do atributo (A_i, A_j) para cada execução da aplicação q_k em cada site S_l e $acc_l(q_k)$ é a quantidade prevista da freqüência de acesso da aplicação definida e modificada para incluir freqüência em diferentes sites.

O resultado dessa computação é uma matriz $n \times n$, cada elemento é mensurado pela definição acima. Essa matriz é conhecida como ***Matriz de afinidade dos atributos*** (*AA*).

Em [Ozsu e Valduriez, 1999] a partir de uma tabela de freqüências de acesso das consultas, é possível criar a matriz de afinidade dos atributos, conforme ilustra a figura 5.8.

$acc_1(q_1) = 15$	$acc_2(q_1) = 20$	$acc_3(q_1) = 15$
$acc_1(q_2) = 5$	$acc_2(q_2) = 0$	$acc_3(q_2) = 5$
$acc_1(q_3) = 25$	$acc_2(q_3) = 25$	$acc_3(q_3) = 25$
$acc_1(q_4) = 3$	$acc_2(q_4) = 0$	$acc_3(q_4) = 3$

Tabela 5.1 – Freqüência de acesso das consultas

	A_1	A_2	A_3	A_4
A_1	45	0	45	0
A_2	0	80	5	75
A_3	45	5	53	3
A_4	0	75	3	78

Figura 5.9 – Matriz de afinidade dos atributos

A fragmentação vertical tem sido estudada dentro do contexto centralizado bem como no distribuído. [Ceri e Pelagatti, 1985] afirmam que a fragmentação vertical é mais complexa que a fragmentação horizontal, devido ao número total de alternativas disponíveis, dois recursos de heurística são discutidos:

Agrupamento: primeiramente sugerido para o ambiente centralizado e depois para o distribuído, consiste em considerar cada atributo como um fragmento, e em cada passo, juntar alguns fragmentos de acordo com a satisfação de algum critério. Essa heurística fornece um conjunto de fragmentos final sobrepostos.

Particionamento: considera uma relação e decide uma divisão benéfica baseada no comportamento do acesso da aplicação para com os atributos, assim como o outro exemplo essa heurística foi primeiramente discutida no ambiente centralizado e depois estendida ao distribuído. As heurísticas de particionamento geram conjuntos de fragmentos não sobrepostos e são normalmente mais eficientes que as de agrupamento.

5.5.1 Algoritmo de agrupamento

O algoritmo apresentado em [Özsu e Valduriez, 1999] que realiza a fragmentação vertical, faz permutações nas linhas e colunas da matriz de afinidade de atributos, gerando uma matriz de afinidade agrupada (*CA*) na qual podem ser identificados agrupamentos de altos valores de afinidade que irão orientar a formação dos fragmentos verticais. A permuta é feita de tal modo que maximize a afinidade global (*AM*):

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1}) + aff(A_{i-1}, A_j) + aff(A_{i+1}, A_j)]$$

$$\text{Onde}: aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) = aff(A_i, A_{n+1}) = 0$$

A matriz de afinidade de atributo (*AA*) é simétrica, o que reduz a fórmula acima para:

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})]$$

A matriz de afinidade agrupada é obtida através do Algoritmo de Ligação de Energia (Binding Energy Algorithm - BEA) que consiste em uma heurística gulosa que a cada passo procura determinar a posição mais proveitosa para uma coluna da matriz.

O objetivo do BEA é maximizar uma função de energia utilizada como medida de afinidade global, ou seja, uma função que cresce com o agrupamento de altos valores de afinidade da matriz e diminui com sua dispersão.

Para o segundo passo do algoritmo, é necessário definir qual é o significado da contribuição de qualquer atributo para mensurar a afinidade. Esta contribuição pode ser derivada como a seguir. Renomeamos que a medida global de afinidade AM tinha sido previamente definido como:

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})]$$

E rescrevemos como:

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} [aff(A_i, A_j)\ aff(A_i, A_{j-1}) + aff(A_i, A_j)\ aff(A_i, A_{j+1})]$$

$$= \sum_{j=1}^{n} [\sum_{i=1}^{n} aff(A_i, A_j)\ aff(A_i, A_{j-1}) + \sum_{i=1}^{n} aff(A_i, A_j)\ aff(A_i, A_{j+1})]$$

Definindo o laço entre dois atributos A_x e A_y como:

$$bond(A_x, A_y) = \sum_{z=1}^{n} aff(A_z, A_x)\ aff(A_x, A_y)$$

Então AM pode ser reescrito como:

$$AM = \sum_{j=1}^{n} [bond(Aj, Aj-1) + bond(Aj, Aj+1)]$$

Agora considere os seguintes atributos:

$$\underbrace{A_1\ A_2 \cdots A_{i-1}}_{AM'}\ \ A_i\ \ A_j\ \ \underbrace{A_{j+1} \cdots A_n}_{AM''}$$

A mensuração da afinidade global para estes atributos pode ser rescrita como :

$$AM_{old} = AM' + AM'' + bond(A_{i-1}, A_i) + bond(A_i, A_j) + bond(A_j, A_i) + bond(A_j, A_{j+1})]$$

$$\sum_{l=1}^{n} [bond(A_l, A_{l-1}) + bond(A_l, A_{l+1})] + \sum_{l=i+2}^{n} bond[(A_l, A_{l-1}) + bond(A_l, A_{l+1})] + 2bond(A_i, A_j)$$

Agora considerando a inserção de um novo atributo A_k entre atributos A_i e A_j dentro da matriz de afinidade de agrupamento, o novo cáluclo de afinidade global pode ser similarmente escrito como:

$$AM_{new} = AM' + AM'' + bond(A_i, A_k) + bond(A_k, A_i) + bond(A_k, A_j) + bond(A_j, A_k)$$

$$= AM' + AM'' + 2bond(A_i, A_k) + 2bond(A_k, A_i)$$

Assim, a contribuição de rede para o cálculo da afinidade global colocando atributo A_k entre A_i e A_j é:

$$cont(A_i, A_k, A_j) = AM_{new} - AM_{old} = 2bond(A_i, A_k) + 2bond(A_k, A_j) - 2bond(A_i, A_j)$$

Com a aplicação do algoritmo BEA na matriz de afinidade dos atributos da figura 5.8 desenvolve-se a matriz de afinidade agrupada (*CA*). A permutação das colunas inicia com as colunas A_1 e A_2 dando origem a figura abaixo.

	A_1	A_2
A_1	45	0
A_2	0	80
A_3	45	5
A_4	0	75

Figura 5.10 – Inicialização do algoritmo BEA

A partir dessa etapa será inserida a próxima coluna A3, três alternativas são identificadas resultando nas seguintes ordenações (A_3,A_1,A_2), (A_1,A_3,A_2) e (A_1,A_2,A_3). Deve-se calcular a afinidade global de cada alternativa. Note que para calcular a afinidade global da alternativa (A_3,A_1,A_2) coloca-se como primeiro atributo o A_0 e na alternativa (A_1,A_2,A_3) será necessário o atributo A_4.

Cálculo da primeira alternativa (A_3,A_1,A_2):

$$cont(A_0, A_3, A_1) = 2bond(A_0, A_3) + 2bond(A_3, A_1) - 2bond(A_0, A_1)$$
$$cont(A_0, A_3, A_1) = 2*(0) + 2*(45*45 + 5*0 + 53*45 + 3*0) - 2*(0)$$
$$cont(A_0, A_3, A_1) = 8820$$

Cálculo da segunda alternativa (A_1,A_3,A_2):

$$cont(A_1, A_3, A_2) = 2bond(A_1, A_3) + 2bond(A_3, A_2) - 2bond(A_1, A_2)$$
$$cont(A_1, A_3, A_2) = 2*(4410) + 2*(890) - 2*(225)$$
$$cont(A_1, A_3, A_2) = 10150$$

Cálculo da terceira alternativa (A_2,A_3,A_4):

$$cont(A_2, A_3, A_4) = 2bond(A_2, A_3) + 2bond(A_3, A_4) - 2bond(A_2, A_4)$$
$$cont(A_2, A_3, A_4) = 2*(890) + 2*(0) - 2*(0)$$
$$cont(A_2, A_3, A_4) = 1780$$

Dentre as três alternativas possíveis a opção (A_1,A_3,A_2) conseguiu o maior índice de afinidade de agrupamento, resultando na figura 5.10

	A_1	A_3	A_2
A_1	45	45	0
A_2	0	5	80
A_3	45	53	5
A_4	0	3	75

Figura 5.11 – Matriz de afinidade de agrupamento com os atributos A_1, A_2 e A_3

O mesmo cálculo é realizado para a inserção do atributo A4, sendo que após a organização das linhas e das colunas o resultando do algoritmo BEA será a matriz de afinidade de agrupamento a seguir.

	A_1	A_3	A_2	A_4
A_1	45	45	0	0
A_3	45	53	5	3
A_2	0	5	80	75
A_4	0	3	75	78

Figura 5.12 – Matriz de afinidade de agrupamento

Percebe-se na matriz de afinidade de agrupamento (CA) a criação de dois grupos distintos, um com o agrupamento que contém alto valor de afinidade e o outro que contém baixo valor de afinidade entre os atributos. O exemplo mostra como os fragmentos da relação podem ser divididos.

5.5.2 Algoritmo de particionamento

Após a obtenção da matriz de afinidade agrupada pelo algoritmo BEA, um procedimento de particionamento é aplicado para identificar os agrupamentos ótimos determinando os fragmentos correspondentes. O procedimento base considera a divisão da matriz em dois grupos de forma que uma medida de avaliação seja maximizada. A medida de avaliação adequada é uma função que tenha valor alto quando for freqüente o acesso de consultas a apenas um dos fragmentos e valor baixo quando for freqüente o acesso de consultas a ambos os fragmentos.

Considerando a matriz de atributos agrupados da figura abaixo, se o ponto ao longo da diagonal é fixo, dois conjuntos de atributos são identificados.

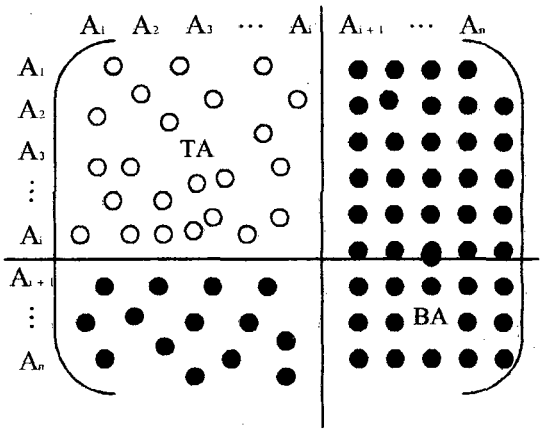


Figura 5.13 – Localizando o ponto de divisão da relação

Forma-se um conjunto $\{A_1, A_2, ..., A_i\}$ no canto superior esquerdo e um segundo conjunto no canto superior direito e em baixo. Chama-se o primeiro conjunto de *top* e o segundo de *bottom* e denota-se os conjuntos de atributos de *TA* e *BA*, respectivamente.

Considerando o conjunto de aplicação $Q = \{q_1, q_2, ..., q_q\}$ e definindo o conjunto de aplicações que acessam apenas *TA*, e apenas *BA*, ou ambos. Esses conjuntos são definidos a seguir:

$$AQ(q_i) = \{A_j \mid use(q_i, A_j) = 1\}$$

$$TQ = \{q_i \mid AQ(q_i) \subseteq TA\}$$

$$BQ = \{q_i \mid AQ(q_i) \subseteq BA\}$$

$$OQ = Q - \{TQ \cup BQ\}$$

A primeira dessas equações define o conjunto de atributos acessados pela aplicação q_i; TQ e BQ são o conjunto de aplicações que acessam apenas TA ou BA, respectivamente, e OQ é o conjunto de aplicações que acessam ambos.

Existe um problema de otimização, se há n atributos de uma relação, há n – 1 posições possíveis onde a divisão pode ocorrer ao longo da diagonal da matriz de agrupamento de atributos, a melhor posição para a divisão é a produzida pelos conjuntos TQ and BQ tal que o acesso total para apenas um fragmento é maximizado enquanto o acesso total para ambos os fragmentos são minimizados. Define-se o custo dessa equação da seguinte forma:

$$CQ = \sum_{q_i \in Q} \sum_{\forall S_j} ref_j(q_i)\, acc_j(q_i)$$

$$CTQ = \sum_{q_i \in TQ} \sum_{\forall S_j} ref_j(q_i)\, acc_j(q_i)$$

$$CBQ = \sum_{q_i \in BQ} \sum_{\forall S_j} ref_j(q_i)\, acc_j(q_i)$$

$$COQ = \sum_{q_i \in OQ} \sum_{\forall S_j} ref_j(q_i)\, acc_j(q_i)$$

Cada uma das equações acima contém o número total de acesso aos atributos pelas aplicações. Baseado nesse cálculo, o problema de otimização é definido como encontrando o ponto x ($1 \leq x \leq n$) tal que a expressão abaixo seja maximizada:

$$Z = CTQ * CBQ - COQ^2$$

Uma importante característica dessa expressão é aquela que define dois fragmentos desde que os valores de CTQ e CBQ sejam quase que igualmente possível. Isso habilita uma distribuição equilibrada dos fragmentos através dos vários sites. É claro que o algoritmo de particionamento tem uma complexidade linear em termos de número de atributos da relação, isto é, $O(n)$.

Há duas complicações nesse caso. A primeira é com respeito ao particionamento, o procedimento divide o conjunto de atributos de dois modos. Um conjunto maior de atributos, é bem provável que n modos de particionamento possam ser necessários. Projetar n modos de particionamento é possível, mas computacionalmente caro.

Ao longo da diagonal da matriz CA, é necessário tentar 1, 2, ..., n – 1 pontos de partição, e para cada é necessário checar qual maximiza z. Assim a complexidade de tal algoritmo é $O(n^2)$. Uma alternativa de solução é aplicar a recursividade do algoritmo de particionamento para cada um dos fragmentos obtidos durante a iteração. Seria computado TQ, BQ e OQ, bem como associado o cálculo do acesso para cada fragmento, e os dividir mais adiante.

A segunda complicação relata o local do bloco de atributos que formam um fragmento. Outra discussão é assumida sobre o ponto de divisão ser único e simples e dividir a matriz CA em bloco superior esquerdo e a segunda parte é formada pelo restante dos atributos.

A fragmentação da relação é completada através da aplicação recursiva do procedimento para realizar a busca pela melhor divisão em *n* grupos. Essa segunda alternativa apresenta o inconveniente de uma complexidade O(2n).

Assumindo que o procedimento de troca, seja conhecido como SHIFT e tenha sido implementado, o algoritmo de particionamento dado, é o PARTITION, que possui como entrada a matriz de afinidade de agrupamento CA, a relação *R*, o atributo usado e a matriz de freqüência de acesso.

A saída é o conjunto de fragmentos FR = $\{R_1, R_2\}$, onde Ri $\subseteq \{A_1, A_2, ..., A_n\}$ e $R_1 \cap R_2$ é igual ao atributo chave da relação *R*. Note que para *n* modos de particionamento, essa rotina deva ser invocada interativamente, ou implementada como um procedimento recursivo.

Uma outra técnica baseada no mapeamento da matriz AA ficou conhecida como Algoritmo Gráfico, de complexidade O(n2). Ele identifica ciclos no grafo e o particiona, e cada subgrafo formado dá origem a um fragmento vertical.

5.5.3 Algoritmo gráfico

Uma proposta de algoritmo gráfico foi feita por [Navathe e Ra, 1989]. Nesta, cada nó representa um atributo e cada aresta o valor de afinidade entre os atributos correspondentes.

O algoritmo inicia-se com a matriz de afinidade dos atributos AA que é considerada como um grafo completo chamado de *grafo de afinidade* no qual cada valor dos arcos representa a afinidade entre dois atributos, conforme demonstra a figura abaixo:

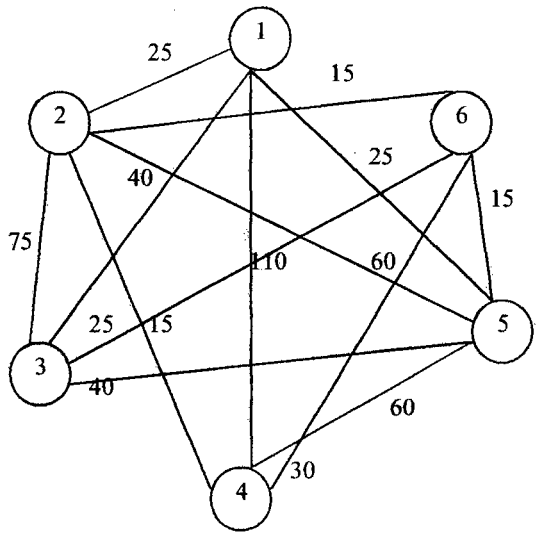


Figura 5.14 – Grafo de afinidade após a exclusão de arcos com o valor 0

Uma árvore linearmente conectada é gerada, considerando cada ciclo como um fragmento.

Este algoritmo apresenta duas grandes vantagens, não utiliza particionamento, onde a cada iteração dois novos problemas são criados aumentando a complexidade, e elimina as mudanças arbitrárias propostas pelas funções empíricas.

Primeiramente o algoritmo constroe um grafo de afinidade de atributos dos objetos existentes, inicia de qualquer nó e seleciona um arco que esteja linearmente conectado que possui o maior valor entre as possibilidades de escolha. Quando o próximo arco for selecionado forma um ciclo primitivo se um ciclo nó não existir, checar a possibilidade de um ciclo, e se existir marque como um ciclo de afinidade.

Considere este ciclo como um candidato a partição. Se o ciclo não existir, descarte o arco e volte a selecionar um outro arco. Quando o próximo arco selecionado não formar um ciclo cheque a possibilidade de extensão do ciclo por um novo arco. A complexidade do algoritmo pode ser considerada $O(n^2)$.

5.5.4 Fragmentação vertical através de função de custo

Em [Chakravarthy et alii, 1992] encontra-se um exemplo de abordagem que, ao invés de utilizar matrizes de afinidades de atributos, emprega uma função de custo explícita para avaliar a qualidade de uma fragmentação vertical e sugere a utilização de algum método heurístico para a busca de uma boa solução.

A função de custo é minimizada quando é atingida uma solução de compromisso entre o custo de acesso a atributos irrelevantes de um fragmento por uma transação e o custo de transmissão de atributos relevantes. Enquanto a minimização do primeiro componente de custo tende a produzir fragmentos de um só atributo, o segundo componente é minimizado quando a relação não é fragmentada.

Como na fase de fragmentação ainda não existem dados sobre a alocação dos fragmentos, é sugerido que o custo de transmissão seja calculado como uma média do custo para as diversas alocações possíveis ou que seja escolhido ou o pior ou o melhor caso de alocação.

5.5.5 Verificando a consistência na fragmentação vertical

É importante assegurar a consistência na fragmentação vertical, para isso a característica completude é garantida desde que cada atributo A da relação global *R* seja nomeado para um fragmento.

$$A = \cup Ri$$

A reconstrução pode ser possível pela operação de junção. Assim, para a relação *R* com fragmentação vertical FR = $\{R_1, R_2, ..., R_r\}$ e atributo chave k,

$$R = \rhd\lhd_k R_i, \forall R_i \in F_R$$

Outro ponto importante é que cada *R*i deve conter o atributo chave de *R*, ou deve conter um sistema que assegura uma identificação das tuplas.

A disjuntividade não é tão importante na fragmentação vertical quanto é na horizontal. Hás dois casos: quando os identificadores de tuplas são replicados em cada fragmento, asseguram e gerenciam as entidades, totalmente invisíveis aos usuários e quando o atributo chave é replicado em cada fragmento, também asseguram e gerenciam as entidades, totalmente invisíveis aos usuários.

5.6 Fragmentação híbrida

Em muitos casos a aplicação simples da fragmentação horizontal ou vertical não é suficiente. Neste caso a fragmentação vertical pode ser seguida pela fragmentação horizontal, ou vice e versa, produzindo uma estrutura tripla de particionamento. Esses dois tipos de estratégias de particionamentos são aplicados um após o outro, essa alternativa é chamada de fragmentação híbrida, também conhecida por fragmentação mista em [Navathe et al., 1994].

A relação *R* é dividida em determinado número de fragmentos R_1, R_2,...R_n. Cada um dos fragmentos é resultado da aplicação da fragmentação horizontal ou vertical do esquema da relação *R*, ou sobre um fragmento de *R* obtido anteriormente.

Suponha que a relação *R* seja a relação *Conta* da figura 5.4, essa relação é dividida inicialmente nos fragmentos $conta_1$ e $conta_2$,. Podemos agora, dividir os fragmentos $conta_1$, usando a fragmentação horizontal do esquema nos dois seguintes fragmentos:

$$Conta_{1a} = \sigma_{\text{nome\_agência} = \text{"Paranavaí"}}(Conta_1)$$

$$Conta_{1b} = \sigma_{\text{nome\_agência} = \text{"Maringá"}}(Conta_1)$$

Assim, a relação *R* é dividida em três fragmentos: $conta_{1a}$, $conta_{1b}$, e $conta_2$. Cada um desses fragmentos pode pertencer a um site diferente.

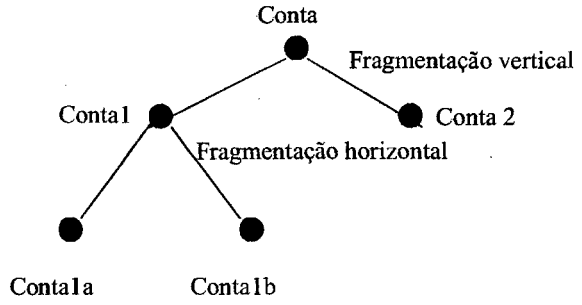


Figura 5.15 – Fragmentação Híbrida

5.6.1 Verificando a consistência na fragmentação híbrida

A garantia de consistência dos dados na fragmentação híbrida é derivada das propriedades comentadas no caso horizontal e vertical. Os operadores para a reconstrução de relações particionadas por este método são aplicados de forma inversa, a árvore da próxima figura é para reconstrução da fragmentação ocorrida figura 5.14.

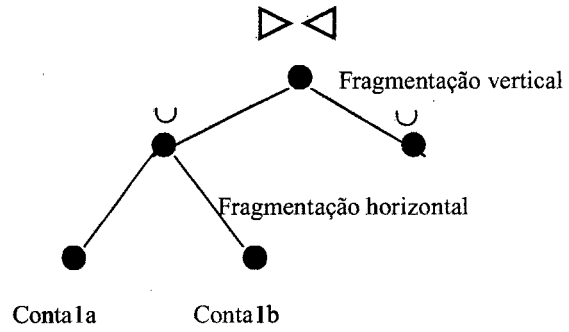


Figura 5.16 – Reconstrução da relação *Conta a*pós a fragmentação híbrida

O próximo quadro resume os aspectos sobre as três opções de fragmentação que foram considerados.

Aspectos	Fragmentação Horizontal		Fragmentação Vertical	Fragmentação Híbrida
	Primária	Derivada		
Particionamento	Ao longo das tuplas		Ao longo dos atributos	tuplas e atributos
Algoritmos ou Heurísticas aplicadas	COM_MIN PHORIZONTAL		Agrupamento Particionamento Gráfico	Combinação das técnicas citadas
Consistência	Completude Reconstrutibilidade Disjuntividade		Completude Reconstrutibilidade	Combinação das características citadas
Operador de Reconstrutibilidade	$\cup$ (união)		$\triangleright\triangleleft$ (junção)	Combinação dos operadores citados

Tabela 5.2 – Resumo dos aspectos das alternativas de fragmentação

5.7 Considerações Finais

O particionamento dos dados é uma importante decisão que faz parte da abordagem *top-down*, neste capítulo foram abordados assuntos relacionados a essa decisão como por exemplo tipos de fragmentação: horizontal, derivada e híbrida, grau de fragmentação, algoritmos como agrupamento, particionamento e algoritmo gráfico, entre outros.

Considerando que a fragmentação tenha sido corretamente realizada o próximo passo é a alocação desses fragmentos entre os sites, assunto do próximo capítulo.

Capítulo 6

ALOCAÇÃO DOS DADOS

De acordo com [Özsu e Valduriez, 1999], o problema da alocação de banco de dados envolve procurar a distribuição ótima dos fragmentos através dos sites, ou seja, considere que há um conjunto de fragmentos $F = \{F_1, F_2, ..., F_n\}$ e uma rede formada pelos sites $S = \{S_1, S_2, ..., S_m\}$ no qual o conjunto de aplicações $Q = \{q_1, q_2, ..., q_q\}$ é executado, a questão seria distribuir F em S.

Segundo [Ceri e Pelagatti, 1985], a questão da alocação dos fragmentos tem sido amplamente analisada no contexto do "*Problema de Alocação de Arquivos*" descrito em vários modelos analíticos que constróem uma alocação ótima dos arquivos sobre diferentes suposições e com diferentes objetivos.

Existem diversos modelos para o problema de alocação de arquivos, mas as soluções obtidas para estes modelos não são aplicáveis ao problema de alocação em bancos de dados distribuídos pelas seguintes razões [Apers, 1988]:

- Os objetos a serem alocados não são conhecidos a priori. Conforme comentado anteriormente, as relações não são adequadas como unidades de alocação porque usuários em diferentes nós podem estar interessados em diferentes fragmentos de uma relação;

- A forma como os dados são acessados é bem mais complexa. No problema de alocação de arquivos as únicas transmissões necessárias para combinação dos dados de diferentes nós são as transmissões dos nós que contêm os arquivos para o nó onde o resultado será computado. Em processamento de consultas distribuídas, transmissões de dados entre os nós onde os fragmentos são alocados podem ser necessárias. Como conseqüência, os fragmentos não podem ser alocados independentemente uns dos outros.

- Os fragmentos não podem ser tratados como arquivos isolados, a alocação de um fragmento tem um impacto sobre os outros fragmentos que podem ser acessados juntos, sua estrutura e comportamento acabariam sendo ignorados;

- Há muitos fragmentos originários da relação global, e muitos modelos analíticos não computam problemas que envolvam muitas variáveis;

- As aplicações fazem "acesso remoto aos arquivos" do sistema, enquanto que as aplicações de banco de dados distribuídos podem fazer uso mais sofisticado dos dados através de modelagem apropriada, como o processamento de consulta.

- O tratamento de arquivos isolados não prevê o custo da garantia de integridade, bem como mecanismos para controle de concorrência.

Isto posto, é necessário separar o tradicional *problema de alocação de arquivo* (file alocation problem - FAP) do problema de alocação de fragmentos dentro de projeto de banco de dados distribuídos, referenciado como problema de alocação de banco de dados (database alocation problem - DAP).

Além de alocar os fragmentos nos sites é importante que esse processo seja otimizado. A questão da otimização pode ser conceituada levando em consideração dois aspectos importantes, sejam eles a performance e o custo mínimo.

A função do custo mínimo pretende procurar um esquema de alocação que minimize a seguinte combinação de custos: custo do armazenamento de cada F_i através dos sites S_j, o custo da consulta de F_i em cada site S_j, o custo da alteração de F_i em todos os sites onde estão armazenados, e o custo da comunicação dos dados. O esquema de alocação escolhido deve ainda manter a performance, através da minimização do tempo de resposta e da maximização do processamento em cada site.

A função do custo mínimo aliada a performance garantem que o esquema de alocação possibilite responder as consultas dos usuários em um menor tempo possível, enquanto o custo do processamento seja mínimo. Alguns modelos não consideram os dois aspectos ao mesmo tempo, e o motivo pelo qual modelos mais aprimorados não foram desenvolvidos é a complexidade.

A questão é que obter a solução ótima provavelmente é computacionalmente impossível, considerando o grande número de fragmentos e sites que formam um banco de dados distribuídos.

Problemas como esses, são complexos e não possuem uma solução satisfatória, já que o ambiente apropriado deveria avaliar a distribuição dos dados mensurando a *otimização* do comportamento das aplicações. Entretanto para isso seria necessária otimização total de aplicações importantes para cada possibilidade de alocação dos dados; com isso seria solucionada a otimização de consulta, considerada um "subproblema" da alocação de dados, só que isto iria requerer uma quantia excessiva de simulação ou uma excessiva complexidade analítica computacional.

Na fase de distribuição é imprescindível que alguns dados sejam recolhidos, como informações quantitativas sobre o banco de dados, as aplicações, a rede de comunicação, a capacidade de processamento, e limitações de armazenamento de cada site.

Para executar a fragmentação horizontal, foi definido a seletividade dos predicados minterms. É preciso estender esse conceito e definir a seletividade dos fragmentos F_j com respeito a consulta q_i. Este valor é o número de tuplas de F_j que serão acessadas pela consulta q_i, pode ser representado por $sel_i(F_j)$.

Essa informação é retirada do banco de dados, outra questão é sobre o tamanho dos fragmentos. O tamanho do fragmento F_j dado por:

$$tamanho(F_j) = card(F_j) * comprimento(F_j)$$

Onde $comprimento(F_j)$ é o comprimento em bytes das tuplas do fragmento F_j.

Muitas informações das aplicações já são compiladas durante a atividade de fragmentação, mas algumas são requisitadas pelo modelo de alocação, dois aspectos importantes são o número de acesso de leitura que uma consulta q_i faz no fragmento F_j durante sua execução, representado por RR_{ij}, e o número de acesso de alteração, representado por UR_{ij}. Isto pode, por exemplo, quantificar o número de blocos acessado por uma consulta.

É preciso definir duas matrizes *UM* e *RM*, na qual os elementos u_{ij} e r_{ij}, são especificados a seguir:

$$u_{ij} = \begin{cases} 1 \text{ se a consulta } q_i \text{ altera o fragmento } F_j \\ 0 \text{ em caso contrário} \end{cases}$$

$$r_{ij} = \begin{cases} 1 \text{ se a consulta } q_i \text{ recupera fragmentos } F_j \\ 0 \text{ em caso contrário} \end{cases}$$

O vetor o de valor o(i) é definido, onde *o(i)* especifica o site originário da consulta q_i. Dessa forma defini-se o tempo de resposta máximo permitido em cada aplicação.

Em cada site, é necessário saber sobre o armazenamento e a capacidade de processamento. Obviamente esses valores podem ser computados por meios de funções elaboradas ou por simples estimativas. O custo da unidade de armazenamento de dados em um dado site S_k pode ser representado por USC_k. Há também a necessidade de especificar a medida de custo de processamento unitário de trabalho LPC_k como o custo dos sites S_k. A unidade de trabalho deve ser identificada para que *RR* e *UR* seja mensurados.

Existem outros modelos que produzem uma rede simples onde o custo de comunicação é definido em termos de um quadro dos dados. Assim g_{ij} representa o custo de comunicação por quadro entre os sites S_i e S_j. Para habilitar o cálculo do número de mensagens, utiliza-se o tamanho de f em bytes de um quadro.

6.1 Análise da Função de Custo e da Performance

Discute-se um modelo de alocação que pressupõe minimizar o custo total de processamento e armazenamento enquanto tenta-se encontrar uma restrição para o tempo de resposta. O modelo usado é:

Min (Custo Total)

Sujeito a restrição sobre tempo de resposta, restrição sobre armazenamento e restrição sobre processamento.

Baseado nos requisitos de informações pode-se expandir os componentes desse modelo, a variável de x_{ij}, é definida como:

$$x_{ij} = \begin{cases} 1 \text{ se o fragmento } F_i \text{ é armazenado em } S_j \\ 0 \text{ em caso contrário} \end{cases}$$

A função do custo total tem dois componentes: processamento de consulta e armazenamento. Pode ser expresso como:

$$TOC = \sum_{\forall qi \in Q} QPC_i + \sum_{\forall Sk \in S} \sum_{\forall Fj \in F} STC_{jk}$$

Onde QPC_i é o custo do processamento de consulta da aplicação q_i, e STC_{jk} é o custo de armazenamento do fragmento F_j nos sites S_k.

O custo do armazenamento total de todos os sites para todos os fragmentos, é dado pela expressão abaixo:

$$STC_{jk} = USC_k * tamanho(F_j) * x_{jk}$$

O custo do processamento de consulta é mais difícil de especificar. Muitos modelos do *Problema de Alocação de Arquivo* (FAP) separam essa questão em dois componentes: o custo de processamento para recuperação, e o custo de processamento para alteração. [Özsu e Valduriez, 1999] utilizam um modelo do *Problema de Alocação de Banco de Dados* (DAP) que considera o custo do processamento (PC) e o custo de transmissão (TC). Assim o custo do processamento de consulta para aplicação q_i é:

$$QPC_i = PC_i + TC_i$$

O componente do processamento PC, consiste no custo de três fatores, o custo de acesso (AC), custo da execução da integridade (IE), e o custo do controle de concorrência (CC):

$$PC_i = AC_i + IE_i + CC_i$$

A especificação detalhada de cada um desses fatores de custo depende do algoritmo usado para realizar esses questões. Entretanto, para demonstrar esse ponto, AC está especificado com alguns detalhes.

$$AC_i = \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} (u_{ij} * UR_{ij} + r_{ij} * RR_{ij}) * x_{jk} * LPC_k$$

O primeiro dos dois termos calcula o número de acesso de consultas do usuário q_i para o fragmento F_j. Note que $(UR_{ij} + RR_{ij})$ oferece o número total de acesso de alteração e recuperação. Considera-se que o custo local de processamento seja idêntico para todos os sites, adiciona-se o número total de acesso para todos os fragmentos referenciados por q_i. Multiplica-se por LPC_k dado o custo desses acesso nos sites S_k. Utiliza-se x_{jk} para selecionar apenas os custos de valores para os sites onde os fragmentos são armazenados.

A função do custo de acesso assume que o processamento de consulta envolve decompor o conjunto em várias sub-consultas, cada qual trabalha com fragmento armazenado em seu site, transmitindo o resultado para o site originário da consulta.

O custo da garantia de integridade é um fato que pode ser específico como o componente de processamento, exceto que o custo da unidade de processamento local provavelmente muda para refletir o verdadeiro custo da garantia de integridade.

A função do custo de transmissão pode ser formulada ao longo da função de custo de acesso. Entretanto, os requisitos para a transmissão de dados para alteração e para recuperação são totalmente diferentes. Em consultas de alteração é necessário informar todos os sites onde existem réplicas, enquanto que nas consultas de recuperação, é suficiente acessar apenas uma das cópias.

Além disso, ao término da atualização do pedido, considerando que não há transmissão de dados de volta ao site originário, ao contrário de uma mensagem de confirmação, considerando que as consultas de recuperação podem resultar em uma significante transmissão de dados.

O componente de alteração da função de transmissão é:

$$TCU_i = \sum_{\forall Sk \in S} \sum_{\forall Fj \in F} u_{ij} * x_{jk} * g_{o(i),\,k} + \sum_{\forall Sk \in S} \sum_{\forall Fj \in F} u_{ij} * x_{jk} * g_{k,\,o(i)}$$

O primeiro termo é para o segundo a mensagem de alteração do site originário $o(i)$ de q_i para todos os fragmentos replicados que necessitam ser alterados. O segundo termo é para confirmação.

O custo de recuperação pode ser especifico como:

$$TCR_i = \sum_{\forall Fj \in F} \min_{Sk \in S} \left(u_{ij} * x_{jk} * g_{o(i),\,k} + r_{ij} * x_{jk} * \frac{sel_i(F_j) * lenght(F_j)}{fsize} * g_{k,\,o(i)}\right)$$

O primeiro termo *TCR* representa o custo da transmissão e recuperação de consultas dos sites que tem cópias de fragmentos que precisam ser acessados. O segundo termo transmite os resultados para os sites originários e o site que tem rendimento mínimo total de custo de transmissão deve ser selecionado para execução da operação. Agora a função do custo total de transmissão para uma consulta q_i pode ser descrita como:

$$TC_i = TCU_i + TCR_i$$

As restrições do tempo de resposta devem ser especificados em termos de tempo e pode ser representado como:

Tempo de execução de $q_i \leq$ *tempo máximo de resposta de* q_i, $\forall q_i \in Q$

As restrições de armazenamento são apresentadas da seguinte forma:

$$\sum_{\forall Fj \in F} STC_{JK} \leq \text{capacidade de armazenamento em cada site } S_k, \forall S_k \in S$$

As restrições de processamento são especificadas como:

$$\sum_{\forall q_i \in Q} \textit{carga de processamento de } q_i \textit{ no site } S_k \leq \textit{capacidade de processamento de } S_k, \forall S_k \in S$$

Está completo o desenvolvimento do modelo de distribuição, estes aspectos são questões importantes a serem identificadas no modelo de alocação.

6.2 Soluções possíveis

As formulações para a solução apresentam-se como sendo um problema NP-completo, diferentes propostas de solução do problema tem sido provada, mas sem resultados contundentes, sendo assim outros métodos devem ser adotados como por exemplo abordagens heurísticas ou métodos de "branch-and-bound".

Um grande número de heurísticas tem sido aplicado para tentar solucionar o modelo FAP e DAP. Outras suposições tem sido estudadas para reduzir a complexidade do problema. Uma das estratégias considera que todos os candidatos a particionamento tenham sido determinados juntos com seus custos e benefícios em termos de processamento de consulta. O problema, então, é modelado como escolha de particionamento ótimo e colocação de cada relação.

Outra simplificação freqüentemente empregada é ignorar a replicação a princípio e procurar uma solução ótima não replicada. Em um segundo passo pratica-se a replicação controlada pela aplicação de algoritmos gananciosos. Para essas heurísticas, entretanto, não há dados suficientes para determinar como encontrar uma solução tida como ótima.

No método de *branch-and-bound*, a cada passo estima-se, para cada uma das operações possíveis na solução parcialmente especificada corrente, um limite inferior para o custo da melhor solução completamente especificada que pode ser obtida como conseqüência da realização da operação e então a operação mais promissora é realizada.

O método de *branch-and-bound* fornece soluções ótimas, mas experimentos indicaram que o método heurístico é capaz de fornecer soluções bem próximas do ótimo a um custo computacional mais viável.

6.3 Estratégia para Alocação dos Dados

Em [Teorey, 1999] alguns ambientes para alocação de dados são citados, como o centralizado, o particionado, o de dados replicados e o de replicação seletiva.

No caso centralizado, todos os dados são localizados em um único site, essa implementação é simples, entretanto, o tamanho do banco de dados é limitado pela disponibilidade de armazenamento do site central. A ocorrência de falhas tornará os dados indisponíveis, visto que os mesmos serão armazenados em conjunto.

No ambiente particionado, o banco de dados será dividido e armazenado em locais distintos, essa alternativa é viável quando a confiabilidade do banco de dados é insuficiente.

A replicação dos dados aloca uma cópia dos fragmentos do banco de dados nos sites que participam do SBDD. Essa estratégia de completa redundância é apropriada quando a questão da disponibilidade da informação é crítica, e operações de alteração são suportadas.

Na replicação seletiva, a duplicação acontece quando é requerida. Fragmentos podem ser considerados críticos ou não. No caso de fragmentos não críticos o armazenamento de apenas uma cópia já é satisfatório, no outro caso os requisitos de disponibilidade e performance influenciam no nível de duplicação exigido.

De acordo com [Ceri e Pelagatti, 1985] é importante distinguir se o projeto de BDD implementará uma alocação redundante ou não-redundante. No caso da não-redundância a situação é mais simples, o método é o de escolher o ambiente que melhor se ajuste a alocação, conhecido como *"best-fit"*, esse ambiente não armazena fragmentos relacionados no mesmo site. O site escolhido será o que apresentar uma referência significativa de consultas e alterações.

A replicação advinda do método de redundância introduz complexidade na etapa de projeto, visto que: o grau de replicação de cada fragmento se torna uma variável do problema; e modelar aplicação de leitura é complicado visto que aplicações podem selecionar entre vários sites alternativos para acessar os fragmentos;

Para determinar a alocação redundante de fragmentos, outros dois métodos podem ser usados, o "all beneficial sites" (ABS) e a alocação progressiva de fragmentos.

No primeiro método é determinado o conjunto de todos os sites onde o benefício da alocação de uma cópia de um fragmento é maior que o custo, e alocar a cópia do fragmento em cada elemento do site.

Na segunda opção, é determinado primeiro a solução de não-replicação, e progressivamente introduzido cópias replicadas onde houver benefício, o processo é terminado quando replicações adicionais forem benéficas.

Ambos os métodos apresentam desvantagens, no método ABS, quantificar custo e benefício para cada fragmento individual é um fator crítico, no método da "replicação adicional" é um ambiente tipicamente de heurística, com esse método, aumentar o grau da redundância é progressivamente menos benéfico. Ambos a disponibilidade e a confiança do sistema aumenta se existir duas ou três cópias de cada fragmento.

O custo/benefício da replicação pode ser estimado através de custos de armazenamento, comunicação e disponibilidade dos dados.

6.4 Medida de Custo e Benefício para Alocação de Fragmentos

Existem algumas fórmulas para avaliar o custo benefício da alocação de fragmentos de uma relação *R*. Considere:

i é o índice de fragmentos

j é o índice de sites

k é o índice de aplicação

f_{kj} é a freqüência da aplicação *k* no site *j*

r_{ki} é o número referente a recuperação do fragmento *i* pela aplicação *k*

u_{ki} é o número referente a alteração do fragmento *i* pela aplicação *k*

$n_{ki} = r_{ki} + u_{ki}$

No caso da fragmentação horizontal, usando método "*best-fit*" para alocação não replicada, coloca-se R_i no site onde o número de referência a R_i for máximo. O número de referência local a R_i no site *j* é:

$$B_{ij} = \sum_k f_{kj} n_{ki}$$

R_i é alocado no site j^* tal que B_{ij^*} seja máximo.

Usando o método ABS para aplicação da replicação, coloca-se R_i em todos os sites *j* onde o custo referente a recuperação de fragmentos pelas aplicações sejam maiores que o custo referente a alteração, para aplicações de qualquer outro site. B_{ij} é avaliado pela seguinte diferença:

$$B_{ij} = \sum_k f_{kj} r_{ki} - C \times \sum_k \sum_{j' \neq j} f_{kj'} u_{ki}$$

C é uma constante que mede a relação entre o custo do acesso para recuperação e para alteração, o acesso de alteração é mais caro, então requer um maior número de controle de mensagens e operações locais (assim, $C \geq 1$).

R_i é alocado em todos os sites $j*$ tal que B_{ij} seja positivo; quando todos B_{ij} são negativos, uma simples cópia de R_i é colocado no site tal que B_{ij} seja máximo.

Usando o método de "replicação adicional", é possível medir o benefício da colocação de novas cópias de R_i aumentando a confiança e disponibilidade do sistema. O benefício pode não crescer proporcionalmente ao grau de redundância de R_i.

Considere que d_i representa o grau de redundância de R_i e F_i denota o benefício de ter R_i completamente replicado em cada site. A função $\beta(d_i)$ é introduzida para medir o benefício, enquanto avaliam-se os benefícios é introduzindo uma nova cópia de R_i no site j modificando a fórmula do caso 2 para:

$$B_{ij} = \sum_k f_{ij} r_{ki} - C \times \sum_k \sum_{j' \neq j} f_{ij'} u_{ki} + \beta(d_i)$$

Essa fórmula é usada para calcular o grau de replicação.

Na fragmentação vertical os benefícios da partição vertical de um fragmento R_i alocado no site r, de um fragmento R_s alocado em um site s e um fragmento R_t alocado em um site t.

- Há dois conjuntos A_s e A_t de aplicações, emitidas dos sites s ou t, que usa atributos de R_s ou R_t e ficam nos sites s ou t, respectivamente, essas aplicações são uma referência remota.

- Há um conjunto A_1 de aplicações antigamente local para r que usa apenas atributos de R_s ou R_t , essas aplicações agora necessitam fazer uma referência remota adicional;

- Há um conjunto A_2 de aplicações antigamente local para r que referencia atributos de ambos R_s e R_t ; essas aplicações fazem duas referências remotas;

- Há um conjunto A_3 de aplicações de sites diferentes quando r, s, ou t que referencia atributos de ambos R_s e R_t , essas aplicações fazem uma referência remota adicional.

Avaliam-se os benefícios dessa partição como:

$$B_{ist} = \sum_{k \in As} f_{ks} n_{ki} + \sum_{k \in Al} f_{kl} n_{ki} - \sum_{k \in Al} f_{kr} n_{ki} - \sum_{k \in A2} 2 \text{ x } f_{kr} n_{ki} - \sum_{k \in A3} \sum_{j \in r,s,l} f_{kj} n_{ki}$$

Essa fórmula calcula o número de acessos, em ordem para distinguir entre acesso de recuperação e alteração, levando em conta os custos diferentes, isto é suficiente para usar (r_{ki} + C x u_{ki}) ao invés de n_{ki}.

Essa fórmula pode ser usada no algoritmo de particionamento exaustivo para determinar se particionar R_i no site i em R_s no site s e R_t no site t é conveniente tentar todas as possíveis combinações dos sites s e t; muito cuidado deve ser tomado quando r = s ou r = t.

6.5 Considerações Finais

O aspectos relacionados a alocação, como análise da função de custo e da performance, possíveis soluções, estratégia para alocação dos dados, vistos neste capítulo fazem parte da estratégia de procedimentos da abordagem *top-donw*, ou seja, quando a criação do banco de dados se dá a partir do zero.

Faz parte da estratégia da outra abordagem a *bottom-up*, a integração. A integração dos dados, assunto do próximo capítulo, trata de prover interoperabilidade entre bancos de dados preexistentes que passaram a formar um único banco de dados heterogêneo.

Capítulo 7

INTEGRAÇÃO DE BANCO DE DADOS

A integração de banco de dados é uma das técnicas da abordagem *bottom-up*, que consiste no processo de integração de bancos de dados preexistentes para formar um único e coeso multidatabase.

O obstáculo na definição do modelo de integração de dados utilizado pelo SGBDH é que alguns estudos 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.

7.1 Integração através de Modelos com Esquema Conceitual Global

Uma alternativa para integração de bancos de dados [Özsu e Valduriez, 1999], é através da especificação de um esquema conceitual global a partir dos esquemas conceituais locais. [Bell e Grimson, 1992] 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, conforme visto na figura 7.1. Deste modo, o esquema global será um conjunto formado por esquemas locais.

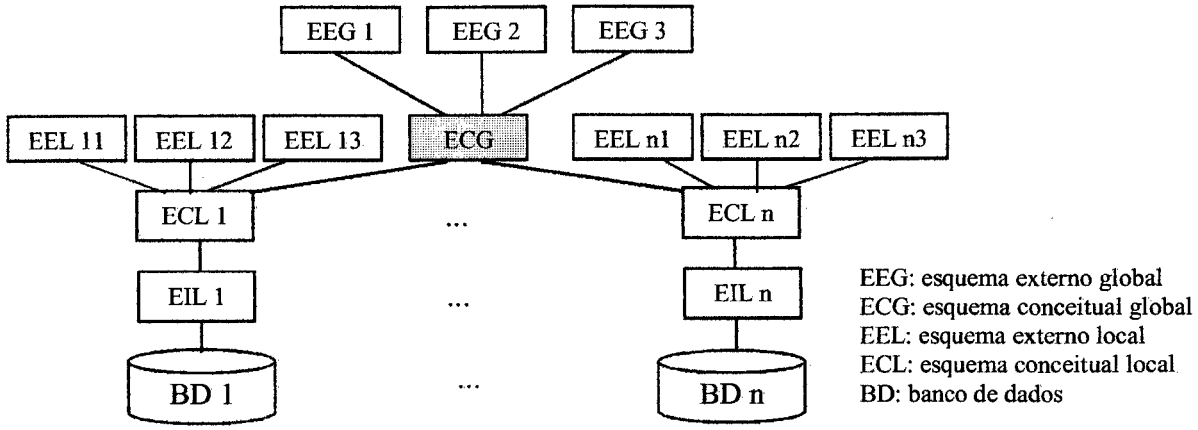


Figura 7.1 - Arquitetura com esquema conceitual global

Uma outra alternativa, é a utilização de uma camada extra chamada "*participation schema*", visto na figura 7.2. 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*".

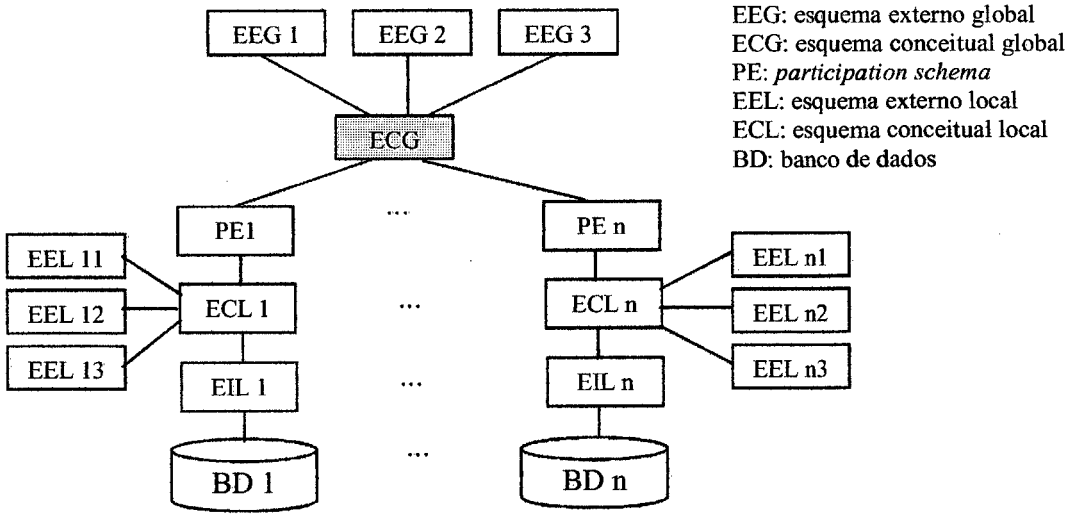


Figura 7.2 - Arquitetura com esquema conceitual global e "participation schema"

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*.

7.1.1 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.

Devido à grande comercialização de bancos de dados relacionais nos últimos anos, atualmente há um grande interesse na integração destes SBDs. 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 7.3 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 é uma metodologia de integração que envolve a manipulação de dois esquemas por vez. Já na metodologia n-ária ocorre a manipulação de mais que dois esquemas por vez.

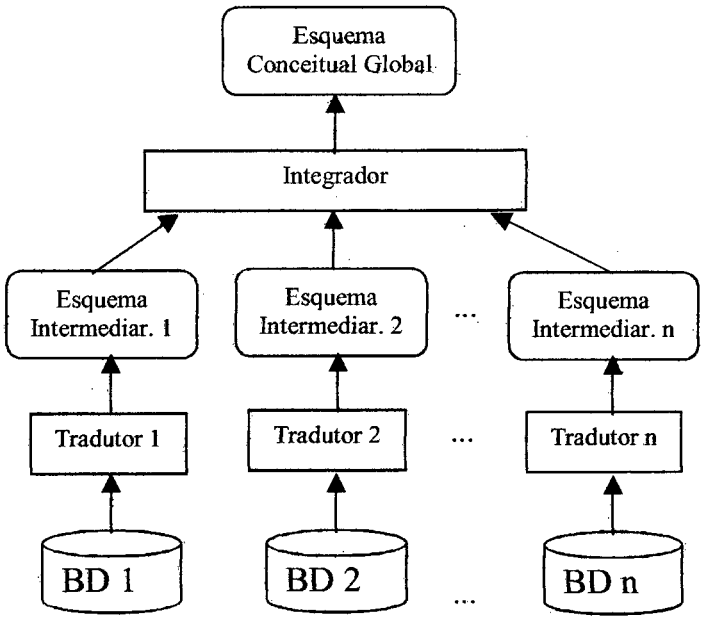


Figura 7.3 - Integração de Bancos de dados: tradução e integração

Na figura 7.3 todos os esquemas são integrados de uma vez, produzindo o esquema conceitual global após uma única iteração chamamos de "one pass integration". 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.

A integração de esquemas pelo método n-ária iterativo ocorre com várias iterações, 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 alii, 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 envolve duas tarefas [Yan e 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 alii, 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 alii, 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.

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.

7.2 Integração através de Modelos Sem Esquema Conceitual Global

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 é relativamente baixo.

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 7.4).

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 7.5).

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.

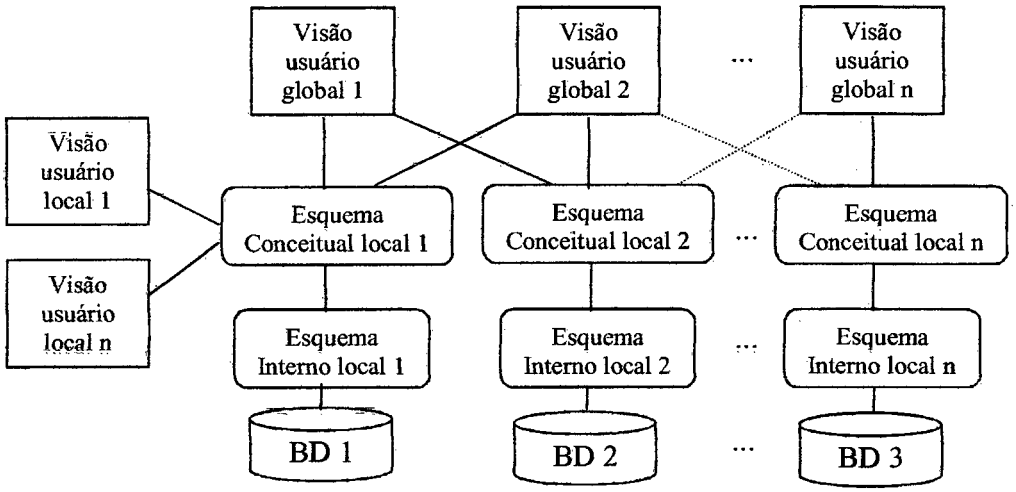


Figura 7.4 - Modelo de um multidatabase fracamente acoplado sem export schema

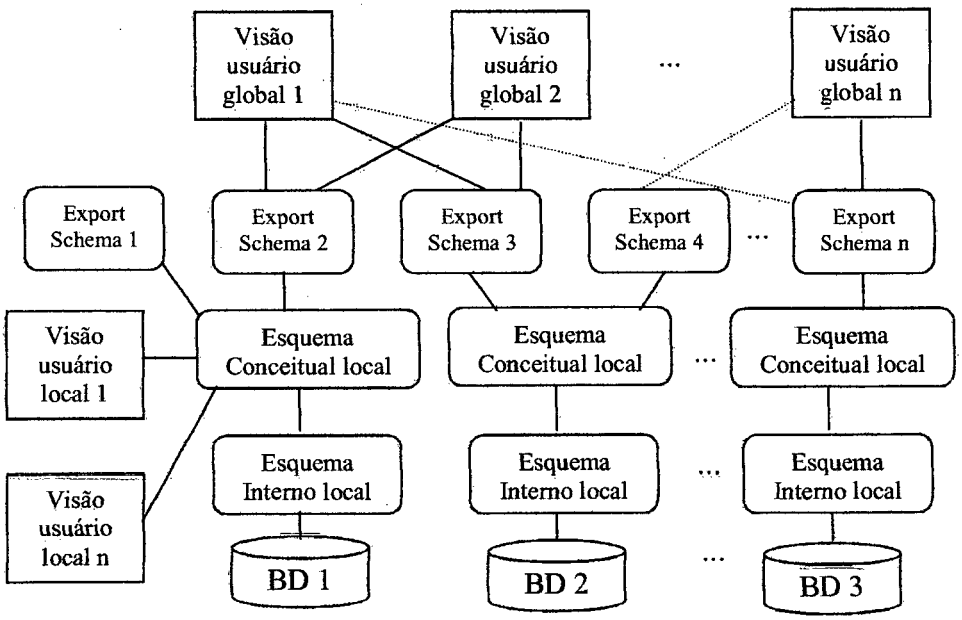


Figura 7.5 - Modelo de um multidatabase fracamente acoplado com export schema

A vantagem da utilização de um sistema sem um ECG, é que estes 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.

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

7.3 Modelos de integração de base de dados

A tabela abaixo 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 7.1: Principais modelos de integração

7.4 Considerações Finais

A abordagem *bottom-up* possui dois modelos que provêem a integração. As arquiteturas que utilizam um esquema global e as que não o especificam. Discute-se muito a necessidade de um esquema global, propondo diferentes propostas para integração.

Neste capítulo, procedeu-se a apresentação destes modelos, e no próximo estão descritas algumas metodologias e ferramentas propostas para projeto de banco de dados distribuídos.

Capítulo 8

FERRAMENTAS E METODOLOGIAS PARA O PROJETO DE DISTRIBUIÇÃO

Devido a problemática apresentada sobre os procedimentos a serem adotados com banco de dados distribuídos, especificamente na área de projeto, algumas ferramentas e metodologias vêem sendo propostas, autores e instituições apostam em pesquisas e desenvolvimento de produtos no intuito de minimizar os erros nessa fase. Nas seções seguintes são apresentadas algumas dessas metodologias e ferramentas e suas principais características.

8.1 Por que utilizar Metodologias no Desenvolvimento de Projeto?

Existem algumas vantagens em utilizar uma boa metodologia de projeto de banco de dados. A metodologia dará condições de projetar uma excelente estrutura do banco de dados, proporcionará um conjunto de técnicas que guiará o projetista passo a passo no processo do projeto. E as chances de erro serão reduzidas ao mínimo. É possível que ainda sim sejam detectados erros antes que investimentos maiores sejam feitos.

A metodologia irá melhorar o produto final – o próprio banco de dados, um banco bem projetado permite o trato com os dados de inúmeras formas diferentes, problemas como presença de dados redundantes, duplicados, inválidos ou até mesmo a ausência de dados requeridos, produzem informações errôneas e fazem com que algumas consultas e relatórios demorem para serem executados.

A maioria destes problemas, senão todos, podem ser facilmente evitados se o projetista realizar um projeto correto de um banco de dados desde o início [Hernandez, 2000].

Grandes vantagens são alcançadas através da execução de um bom projeto, já que o tempo gasto com o projeto será um *investimento*, visto que o projetista não precisará revê-lo constantemente. Em um banco de dados bem projetado, as vantagens são facilmente detectadas:

- Facilidade em modificar e manter a estrutura, pois as modificações feitas em um atributo não afetam os outros; e facilidade em modificar os dados, mudanças realizadas nos valores de um campo de uma tabela não afetará outros campos;

- Facilidade em trabalhar os dados, as tabelas estão bem projetadas e os relacionamentos facilmente acessados;

- Facilidade em construir e desenvolver aplicações para o usuário, o tempo de programação pode ser gasto projetando aplicações úteis ao invés de ter que lidar com problemas advindos de um projeto deficiente.

No caso de um banco de dados distribuídos a metodologia de projeto deve ainda:

- Prever o tratamento do esquema de distribuição dos dados, técnicas de fragmentação e alocação dos dados para a estratégia *top-down*; e prever o comportamento na integração dos dados para o caso da estratégia *botton-up*.

Construir um ambiente de banco de dados distribuído é um processo de emparelhar as necessidades dos usuários às realidades da infra-estrutura física e tecnologias disponíveis. É necessário que sejam feitos acordos entre as exigências do negócio e os custos associados com a construção da infra-estrutura física necessária para apoiar essas demandas.

Neste ponto a metodologia é apresentada não apenas como um plano de trabalho rígido, mas sim para assegurar que o conjunto de decisões principal foi tomado. Além de promover uma aproximação que ajudará a fomentar a aplicação e o pessoal de apoio técnico a conhecer metas estratégicas e táticas para distribuir dados e processos.

8.1.1 Proposta metodológica para fragmentação mista

Esta metodologia é baseada no algoritmo teórico de grafo, que faz o agrupamento de um conjunto de atributos e predicados dentro de um conjunto de fragmentos verticais e horizontais. A aplicação em conjunto desses dois algoritmos forma uma tabela, a união de células dessa tabela em grupos híbridos tentam minimizar o número de acesso a disco necessários para processar uma transação.

Na ilustração abaixo, apresentada por [Navathe et alii, 1994], pode-se observar as alternativas relacionadas em executar a fase de distribuição dos dados.

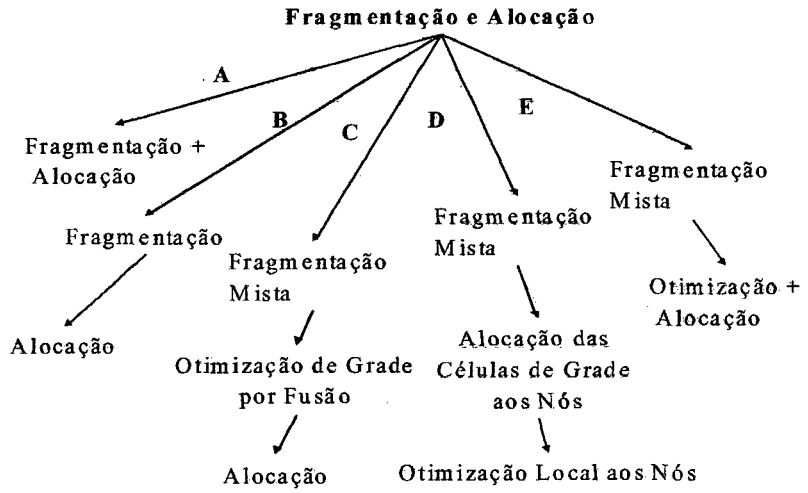


Figura 8.1 - Opções para o projeto de distribuição

A alternativa A da citada figura trata os dois problemas de forma integrada, enquanto as demais alternativas tratam a fragmentação e a alocação como dois problemas de otimização independentes.

Nas alternativas A e B assume-se que a fragmentação vertical e a fragmentação horizontal são realizadas separadamente, enquanto nas alternativas C, D e E é empregada a fragmentação mista, que consiste na aplicação simultânea de fragmentação horizontal e vertical.

A fragmentação mista divide as relações em unidades denominadas células de grade que são agrupadas durante o processo de otimização de grade para formar os fragmentos finais essa modalidade foi apresentada em [Navathe, Karlapalem e Ra, 1994].

Na alternativa C, a otimização de grade é realizada antes da alocação. Em contrapartida, na alternativa D, alocação é realizada com base nas células de grade e a otimização é executada posteriormente. Finalmente, na alternativa E, a alocação e a otimização de grade são tratadas simultaneamente.

A alternativa A reduz o projeto de BDD a um único problema porém, tem sido pouco abordada devido à complexidade do espaço de busca a ser explorado. A proposta envolve as atividades de compor as tabelas e criar um otimizador das células das tabelas, os passos para a metodologia mista são ilustrados a seguir.

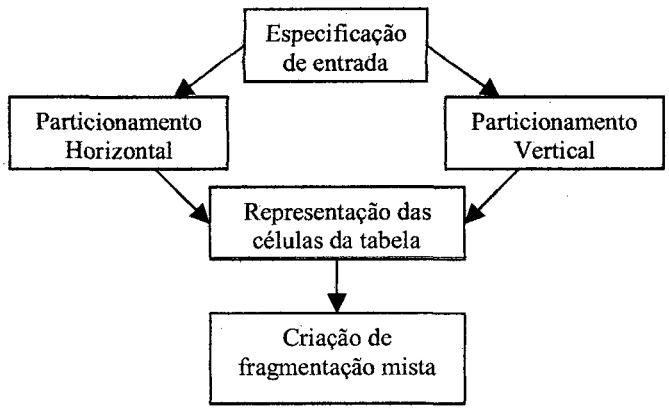


Figura 8.2 – Passos da metodologia mista

Na etapa de "*Especificação de entrada*" todos os dados sobre as relações, seus atributos, cardinalidades, predicados entre outros são considerados, bem como as informações sobre as transações, ou seja, freqüência, emprego dos atributos e dos predicados, essas informações estarão contidas na matriz da utilização dos atributos.

Além desses aspectos, as restrições na distribuição são consideradas, como qualquer particionamento pré-determinado ou alocação fixa dos dados. Informações sobre os sites, e custos de transmissão são utilizadas particularmente para solucionar o problema da alocação.

Os passos de particionamento vertical e horizontal podem ser concluídos coincidentemente. O esquema para representar as células da tabela é desenvolvido e estas serão armazenadas em um catálogo de sistema. Essa representação será utilizada no algoritmo que fará a fragmentação mista. O número de acessos requeridos para executar a transação deverá ser usado para computar o conjunto ótimo de fragmentos mistos assim como para minimizar o número total de acesso a disco requerido para o processamento das transações.

8.1.2 Metodologia de projeto de distribuição por Buretta [Buretta, 1997]

Segundo [Buretta, 1997], algumas tarefas devem ser cumpridas para garantir uma distribuição justa dos dados e das aplicações, construindo um rígido esquema de distribuição, sejam eles reconhecer o perfil da aplicação, infra-estrutura física, criar ligação entre os grupos de recuperação de dados, perfil de dados-para-processos, perfil de integração global de dados, criação de esquema de alocação de dados, validação do esquema de alocação considerando capacidade, dificuldades e tecnologias disponíveis, validação do esquema de alocação contra o nível de serviço requisitado e a implementação do esquema de distribuição.

Na etapa de preparar o perfil da distribuição da aplicação, é descrita a natureza da distribuição da aplicação e iniciado a solidificação das exigências de rede, essas informações constam da etapa de análise de requisitos descrita no capítulo 4.

Para prover informações suficientes é necessário identificar os tipos lógicos de local para aplicação, ou seja, categoria de sites, como centrais, filiais, regionais e etc. Os tipos de usuários finais para cada aplicação, como gerentes, balconistas, comerciantes e etc. Identificar as transações e os processos que são executados por cada tipo de usuário final, com previsão de execução de taxas e identificação antecipada da quantidade de usuários finais.

A configuração de rede e a estrutura física dos dados para a empresa devem ser projetado para apoiar esta estrutura organizacional.

No reconhecimento da infra-estrutura física são documentados os recursos físicos disponíveis em cada site e são identificadas quaisquer restrições para uma distribuição em potencial.

Para determinar a viabilidade da distribuição dos dados e ou processos em dado local, é importante ter uma compreensão completa de todos os recursos e obstáculos daquele possível local de desenvolvimento.

Algumas questões são consideradas como a conectividade de rede, recursos de hardware e software, sua capacidade e disponibilidade, níveis de segurança, sistemas disponíveis e exigências atualmente requeridas, suporte administrativo e outros níveis de habilidade, todos esses dados são descritos na etapa de requisitos do sistema.

Na próxima tarefa são identificados os grupos de recuperação de dados e as oportunidades em potencial para implementação de dados distribuídos através dos sites.

Um importante componente do esquema de distribuição é a integridade, ela deve ser preservada não só durante uma atividade normal mas também durante uma atividade de administração através de recuperação de falhas. A ênfase principal desta tarefa é o agrupamento de fontes de dados primárias. Sendo que estas são os recipientes de toda a atividade transacional.

Dentro da identificação do Esquema Conceitual Global estão incluídos o desenvolvimento do Diagrama de Entidade-Relacionamento (DER), o reconhecimento das ligações de integridade entre dados e entidades e o entendimento completo de qualquer apoio de integridade entre os sites.

Esse ambiente requer a divisão do diagrama de entidade-relacionamento em subconjuntos de forma que nenhum deles contenha referências fundamentais a outro subconjunto. A meta é eliminar ou pelo menos minimizar o número de partições que poderiam ter estes referências cruzadas nos subconjuntos.

O apoio de integridade entre os sites deve ser provido pelo código da aplicação ou deve ser coordenado por procedimentos externos. Esses grupos de recuperação de

dados geralmente representam dados de transações atômicas que devem ser recuperadas em conjunto.

Uma das etapas de fragmentação consiste em preparar o perfil de dados-para-processos, essa etapa tem por objetivo identificar dados usados por processos e realçar os processos com carga de processamento mais pesada.

Nessa etapa é realizado o DER, o diagrama da decomposição funcional e o conhecimento das características de cada processo de aplicação.

Exigência crítica do nível de serviço, um tempo de resposta muito rápido é requerido. A exigência para quantias grandes de dados e ou processo complexo.

Logo a seguir é identificado o perfil de integração global de dados, o objetivo dessa etapa é a integração do uso de dados global, esta tarefa inclui o seguinte: utilização de perfil previamente executado, conhecimento dos dados externos e da aplicação.

Mas este papel de tutela geralmente requer o seguinte: responsabilidade por coordenar as regras empresariais associadas aos dados, responsabilidade por coordenar as regras administrativas que apoiam cada entidade de dados e garantir a segurança dos dados dentro do domínio, identificação dos dados requeridos, da fonte primária para cada conjunto de dados, identificação da taxa de mudança típica para cada conjunto de dados, entre outras coisas.

A próxima tarefa compreende a criação de esquema de alocação dos dados, dentro do esquema conceitual local, nessa etapa é integrado todo o conhecimento adquirido sobre o esquema. Ele se toma um caminho inicial executado em tarefas futuras que serão validadas com respeito a obstáculos existentes, capacidades, tecnologias, e exigências.

Essa tarefa incluem as anteriores mais o reconhecimento da estratégia de direção com respeito a tecnologia utilizada, como: identificar locais viáveis para distribuição, solidificar a seleção de um modelo de aplicação, distribuição do dados, infra-estrutura física, reconhecimento de recuperação de dados, reconhecimento das tecnologias existentes dentro da empresa, e etc.

Após a criação desse esquema é necessária a validação do esquema de alocação considerando capacidade, dificuldades e tecnologias disponíveis.

É imprescindível o conhecimento das funcionalidades das tecnologias existentes dentro da empresa, para assegurar que sejam apoiados a: confiança de uma plataforma particular, o apoio da administração, e que qualquer outro obstáculo de infra-estrutura física pode ser solucionado, assegurar que não existe nenhuma limitação de plataforma, de banco de dados, de rede, de tráfico e apoio para unidades de trabalho e para a alternativa de replicação apropriada.

Uma comparação da validação do esquema de alocação contra o nível de serviço requisitado é necessária, assim é tido o conhecimento das capacidades de desempenho das tecnologias existentes dentro da empresa.

Após essas tarefas ocorre a implementação do esquema de distribuição, os passos dessa metodologia deveriam ser seguidos como uma ferramenta para despachar o desenvolvimento de aplicações distribuídas, observando-se o comportamento e garantindo a manutenção. Se esses passos fossem seguidos não só os dados necessários estariam disponíveis, como estariam disponíveis de maneira eficiente e com o nível apropriado de latência.

8.1.3 Metodologia DATAID-D

A metodologia DATAID-D proposta para a abordagem *top-down* desenvolvida no Instituto Politecnico di Milano e referenciada em [Ferreira e Finger, 2000], sugere ao projetista os problemas relevantes ao projeto de distribuição, bem como parâmetros para solucioná-los, e como cada problema pode ser tratado.

Essa metodologia propõe duas novas fases: a análise de requisitos de distribuição e o projeto de distribuição.

A análise de requisitos de distribuição obtém informações sobre a distribuição, tais como predicados para fragmentação horizontal e a freqüência de ativação de cada aplicação em cada local. Essa fase utiliza os requisitos do usuário em relação a distribuição, o esquema global de dados e as operações construídas na fase do projeto conceitual.

O projeto de distribuição aloca os dados aos locais definidos a partir do esquema global de dados e das tabelas lógicas de acesso construídas na fase de projeto lógico global, e dos requisitos de distribuição obtidos na fase de análise de requisitos de distribuição.

Através do modelo de entidade-relacionamento e da tabela de acesso que indica o tipo de operação de acesso à base de dados, ou seja leitura ou escrita, é gerado um esquema lógico dos dados e das tabelas de acesso para cada local. Para essa metodologia o projeto de distribuição é subdividido em quatros fases:

a) projeto de fragmentação: nessa etapa é aplicada técnicas de fragmentação nas entidades com o objetivo de determinar possíveis unidades de alocação. Esse passo é uma decisão lógica realizada através da seleção de predicados das tabelas de polarização e da composição destes dentro de fragmentos definidos logicamente;

b) Alocação não redundante: nessa etapa, é realizada o mapeamento de cada fragmento ao local onde é mais freqüentemente usado, essa freqüência é obtida pela somatória, para todas as transações emitidas daquele local, do produto entre polarizações e freqüência de uso do fragmento.

c) Alocação redundante: realizada através da seleção de locais adicionais para cada fragmento inicialmente alocado de forma não-redundante, usando heurísticas "gananciosas". Antes da cópia os benefícios devem ser avaliados em relação a recuperação da informação, se compensam o custo e a complexidade gerados pela atualização das cópias.

d) Reconstrução de esquemas locais: construção de esquema local a partir da versão final da alocação de fragmentos. Essa metodologia sugere posicionar os relacionamentos no mesmo local das entidades ou fragmentos com maior cardinalidade. A figura 8.1 indica as subdivisões citadas da metodologia DATAID-D.

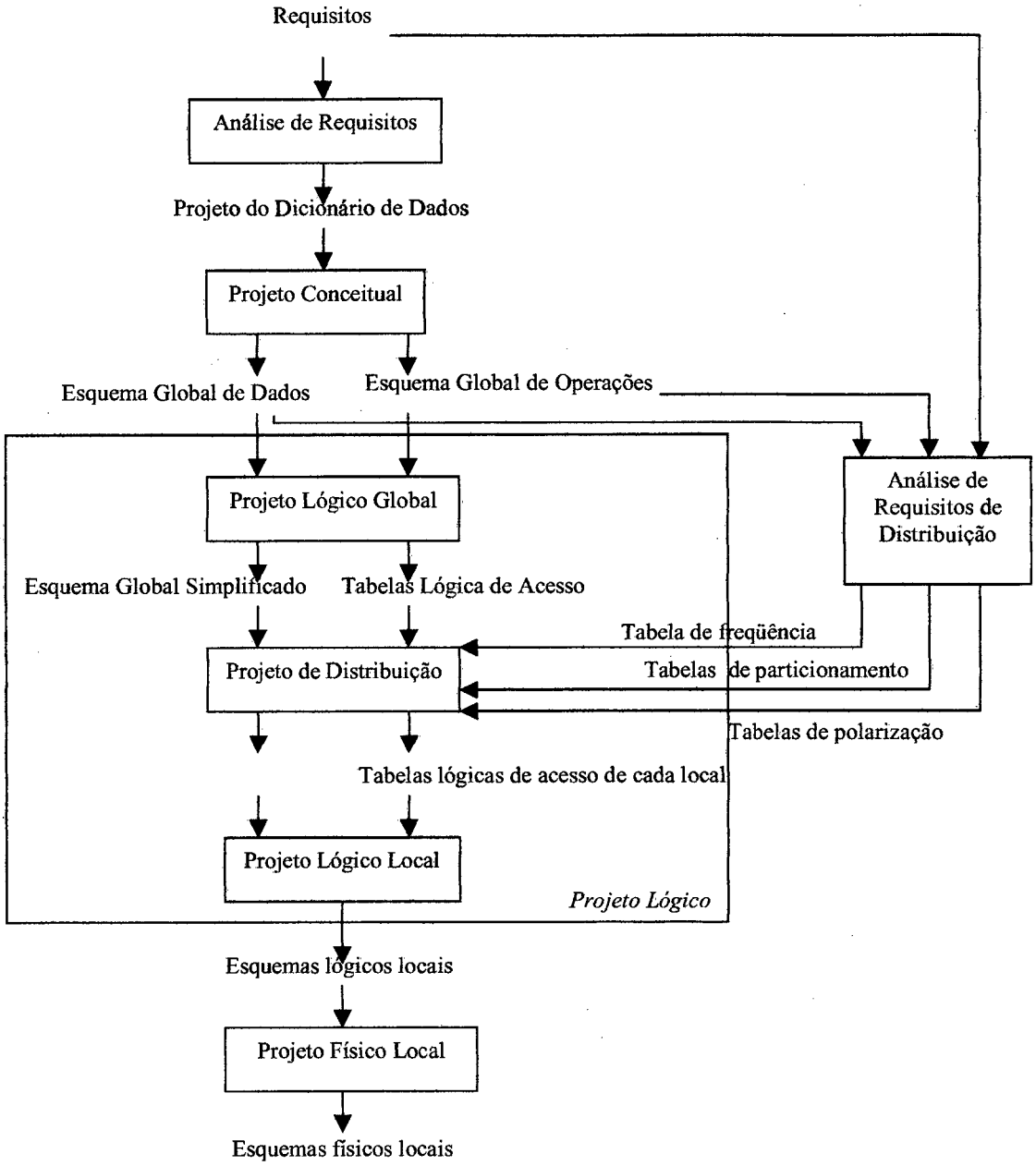


Figura 8.3 – Metodologia DATAID-D

8.1.4 Metodologia para projeto de bancos de dados distribuídos proposta por Miranda [Miranda, 1999]

Estudos realizadas no IME por [Miranda,1999] culminaram em uma proposta metodológica para projeto de bancos de dados distribuídos. O objetivo da proposta é descrever uma abordagem alternativa caracterizada pelo tratamento do projeto de BDD como um problema de otimização que unifica fragmentação e alocação e pela utilização de heurísticas genéricas de busca local.

A fim de atingir esse objetivo, dois níveis de procedimentos foram idealizados. O primeiro nível tem com premissa apenas a otimização da fragmentação e da alocação em uma única fase através de uma heurística de busca local. O ponto crítico desta abordagem é o fato de que a mesma consiste em uma busca em um espaço de soluções extremamente complexo.

As entradas para o primeiro nível de procedimentos são as características dos ambientes onde será desenvolvido e implantado o SBDD e seu produto é uma ferramenta para execução de procedimentos de segundo nível.

O segundo nível, que constitui a metodologia para o projeto do BDD propriamente dito, recebe como entrada informações de um esquema global, de uma rede e de um conjunto de aplicações e fornece como saída os esquemas de fragmentação e alocação dos dados. De forma genérica os passos seguidos no segundo nível são:

1) Aquisição de informações: é a obtenção e entrada na ferramenta das informações exigidas pelos modelos do esquema local, da rede e das aplicações.

2) Escolha da solução inicial: nesta fase, o projetista seleciona a solução inicial que será utilizada como ponto de partida para o algoritmo de busca local. A escolha de uma solução inicial sem fragmentação corresponde à opção por uma estratégia de projeto *top-down*, enquanto uma solução inicial com fragmentos mínimos denota uma estratégia *bottom-up*. O projetista tem ainda liberdade para adotar outras estratégias como, por exemplo, a realização de um projeto inicial empírico que aparente estar próximo da solução ótima ou ainda a utilização de uma solução inicial aleatória;

3) Otimização: é a fase em que o projetista emprega o otimizador da ferramenta a fim de obter uma solução de bom custo para o projeto;

4) Pós-otimização: é a fase em que o projetista pode realizar alterações manuais no projeto relativas a técnicas não incorporadas à ferramenta desenvolvida como, por exemplo, a fragmentação derivada;

5) Geração dos esquemas físicos: é a utilização da interface da ferramenta com o SGBDD para obtenção dos esquemas físicos locais.

A pesquisa de [Miranda,1999] esquematiza ainda uma ferramenta no apoio a metodologia proposta, a partir de uma interface gráfica com o usuário são recebidas as informações de entrada; essas informações são armazenadas em um metabanco de dados e utilizadas pelo otimizador para produzir esquemas locais que são também armazenados no metabanco de dados; esses esquemas locais são então exibidos ao usuário para eventuais alterações; finalmente a ferramenta gera uma representação dos esquemas locais que possa ser entendida pelo SGBD.

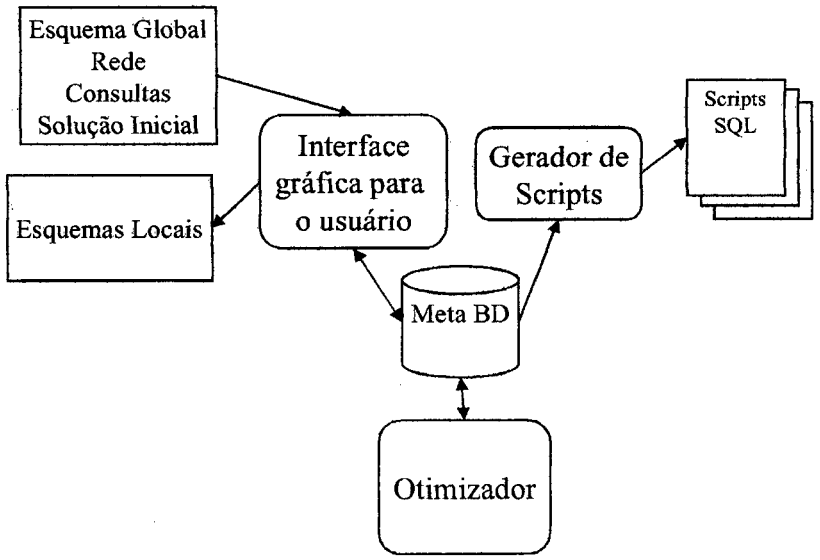


Figura 8.4 – Ferramenta para projeto de BDD

8.2 Ferramentas

Várias ferramentas são utilizadas para produzir o modelo de dados a ser implementado, algumas como o Designer/2000, a ferramenta inteligente para a etapa de projeto físico aplicada aos produtos DBDSGN e RdbExpert, o Gerenciador de Restrição, entre outras estão descritas logo a seguir.

8.2.1 Designer/2000 da ORACLE® Corporation

[Owens e Sheldon, 1998] descreveram uma ferramenta baseada em banco de dados distribuídos, a Webster University implantou um Sistema de Informação do Grupo Acadêmico, conhecido como AAIS (Academic Affairs Information System) responsável por gerenciar as informações do campus sede e seus outros 70 sites remotos. Para projetar esse banco de dados foi utilizada a ferramenta Designer/2000, que possui vários módulos diferentes responsáveis por atividades diferentes.

O módulo Designer/2000 Process Modeler, foi usado na análise e documentação dos requisitos funcionais do AAIS. O Designer/2000 Data Modeler foi responsável por criar o modelo de dados inicial, que continha as entidades, atributos e relacionamento, bem como a aplicação da terceira forma normal (3FN), além das regras de negócio e suas derivações.

Na fase de projeto o Designer/2000 Server Generator produziu o modelo de dados físico a ser implementado com a versão do ORACLE® 7.3.3 Sistema Gerenciador de Banco de Dados Relacional e superiores. Nesta etapa estão incluídos a definição das tabelas, colunas e relacionamento entre as entidades. A 3FN inicialmente aplicada foi iterativamente refinada e em alguns casos as entidades foram denormalizadas.

Para a construção do projeto de distribuição de dados do AAIS, algumas questões foram consideradas, como sobre a disponibilidade dos dados requisitados, e sobre como armazenar os dados, ou seja, a decisão de deixar todos os dados em todos os sites, ou alguns dados em alguns sites.

A definição dessas questões se apresentou após a análise da necessidade do AAIS, ou seja, não era necessário que todos os dados estivessem em todos os sites ao mesmo tempo e que nem todos os sites precisavam ter acesso as informações contidas em outros sites, apenas o campus sede necessitava ter acesso a todos os dados de todos os sites.

Outra questão que foi levada em consideração por [Owens e Sheldon, 1998]foi sobre a freqüência de acesso aos dados, e seus valores. No ambiente analisado, considerando que os sites são distribuídos por todo o continente dos Estados Unidos, Europa e Ásia a freqüência de acesso aos dados pode ser considerada basicamente como diária.

Considerações sobre a atualização de dados foram realizadas, ou seja, muitos dados voláteis como informações dos estudantes, horários, registros, são mudados constantemente pelos sites remotos. Esses dados são submetidos a aprovação do campus sede que provê semanalmente ou mensalmente informações atualizadas para os sites remotos. Já algumas informações não voláteis como dados sobre os cursos, graus e curricula que não mudam freqüentemente, também fica sob a responsabilidade do campus sede.

Após análise dessas questões sobre a disponibilidade dos dados, tipos de acesso e volatilidade foi delineado as características que o AAIS necessitava como: replicação simétrica, cópias remotas, snapshots, procedimentos de armazenamento e etc.

A replicação simétrica é utilizada na manutenção das alterações entre as cópias dos dados voláteis ao longo dos múltiplos sites. Esta característica considera de primordial relevância a disponibilidade e integridade dos dados.

Os snapshots são usados para fazer cópias das tabelas, são definidos fazendo consultas a uma tabela ou a um conjunto de tabelas.

A integridade dos dados foi mantida através da implantação de um projeto simples caso haja uma falha devido a problemas de comunicação entre os sites remotos e o campus sede. Sendo que quando da restauração da queda de comunicação os sites revêem as transações que não foram processadas e sincronizam a base de dados.

O projeto de segurança foi desenvolvido considerando quem poderia acessar o AAIS e como seria a disponibilidade dos dados depois de acessados, além de controlar a permissão e os privilégios de cada grupo de usuário.

Na construção dos protótipos dos formulários foi usado o módulo Developer/2000 Forms Designer.

O ambiente AAIS possui outras extensões de trabalhos que estão sendo pesquisadas e serão futuramente implementadas.

8.2.2 Gerenciamento de restrição em projeto de banco de dados distribuídos

[Gupta e Tiwari,1991] tratam da arquitetura de um Gerenciador de Restrições no Projeto de Banco de Dados Distribuídos (DCMS – Distributed Constraint Management System) que propõe detectar possíveis inconsistências através da checagem de restrições.

Nem todos as restrições do projeto global são disponíveis na criação do banco de dados, eles podem surgir durante o projeto de processo atual, isto é, quando as bases de dados interagem enquanto executam uma aplicação de domínio específico.

Outra dimensão do problema envolve checar as restrições quando da alteração dos fundamentos das bases de dados. Checar as restrições globais pode ser muito custoso devido à necessidade de acessar dados remotamente. Assim, a otimização da restrição em tempo de compilação é importante para a otimização da checagem do processo em tempo de execução.

Essa arquitetura é projetada para incorporar muitos estágios de otimização, seu procedimento de trabalho envolve checar as restrições mas não corrigir suas violações, apesar de notificar os participantes afetados pela violação da restrição. Essa arquitetura foi implementada no Centro de Pesquisa IBM-Almaden com o sistema de banco de dados Starburts, e portada para ORACLE®.

Os requisitos de consistência podem ser impostos em sites simples ou em múltiplos sites, e são classificados em global e local. A arquitetura representada na figura 7.1 trata as restrições como classe de objetos, permite que sejam consultadas e alteradas, pois os dados as quais são impostas também são consultados e alterados.

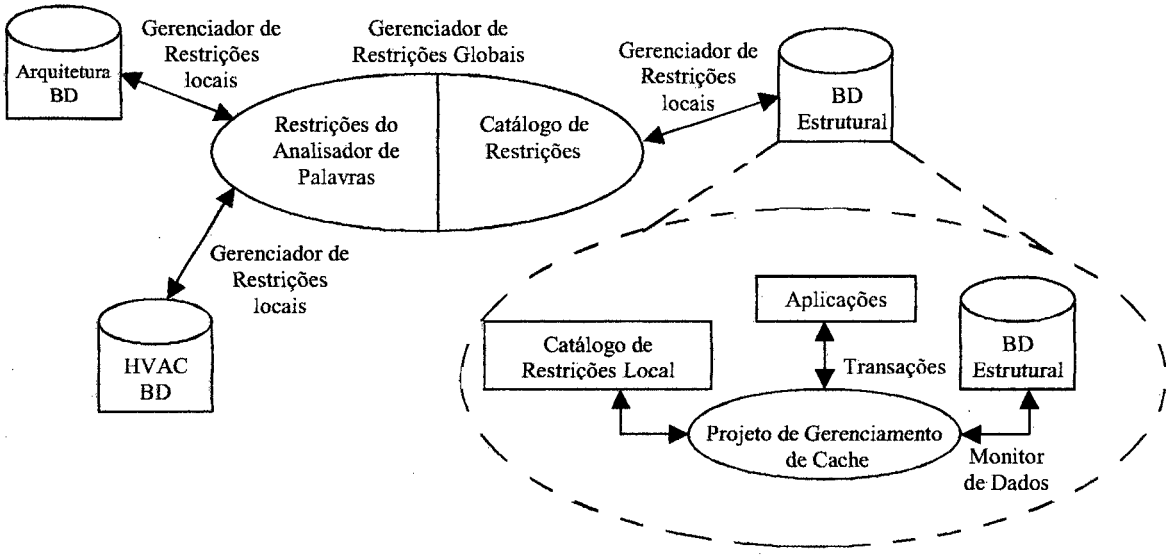


Figura 8.5 - Gerenciador de restrições no projeto de banco de dados distribuídos

A arquitetura começa a funcionar a partir da especificação das restrições, que são armazenadas no Gerenciador de Restrições Global, esse mecanismo fragmenta as restrições e produz uma informação otimizada que será distribuída aos sites participantes como um Gerenciador de Restrições Local.

Esse mecanismo é responsável por checar as restrições, e as informações compiladas são enviadas ao catálogo, que também dão suporte as consultas e alterações das restrições.

O monitor trilha as mudanças nos objetos do projeto de banco de dados e detecta a ocorrência de operações que potencialmente violam as restrições.

As notificações são a fase final do gerenciador de restrições, são ativadas apenas quando há a violação das restrições, os participantes envolvidos são selecionados como responsáveis em resolver a inconsistência despertada pela violação das restrições.

8.2.3 Ferramenta inteligente para o projeto físico de banco de dados

[Tewari, 1990] trata de ferramentas inteligentes para a fase de projeto de dados físico. Essa etapa envolve determinar a estrutura física dos arquivos do banco de dados, alocação dos dados e seleção do caminho de acesso.

O problema de alocação de arquivo de banco de dados no ambiente distribuído, bem como o particionamento dos registros (método horizontal, vertical ou misto) tem se mostrado ser um problema NP-Difícil. As soluções conhecidas tendem a ser heurísticas, e tentam reorganizar periodicamente os dados alocados inicialmente.

A figura 8.4 descreve os requisitos de entrada da fase de projeto de banco de dados físico e as saídas produzidas por esse processo.

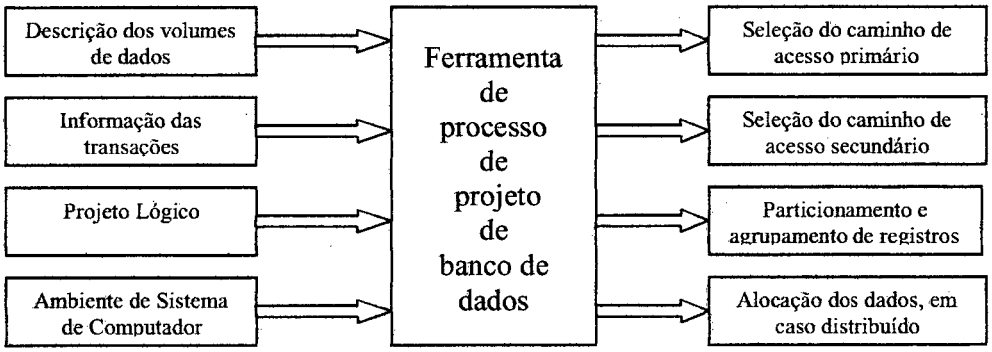


Figura 8.6 – Processo de projeto de banco de dados físico

No ambiente demonstrado são aceitos como requisitos de entrada a descrição do volume de dados, onde estão descritas as entidades, atributos, cardinalidade das relações e etc. Fazem parte da entrada também, a especificação dos tipos e freqüência de transação e a especificação da estrutura lógica dos dados em termos de diagramas ER, diagrama de estrutura de dados e diagrama de fluxo de dados.

As saídas produzidas pelo processo de projeto de banco de dados físico são a especificação do caminho de acesso primário e secundário, agrupamento de registros e atributos particionados, especificando o agrupamento horizontal ou vertical dos dados e a alocação dos recursos distribuídos, onde dados sobre o número de cópias de cada objeto são disponibilizados.

Em adição ao inicial projeto de banco de dados físico, a periodicidade da reorganização dos dados pode ter de ser executada em resposta a alteração dos parâmetros de entrada. A decisão de quando reorganizar o banco de dados é importante, e também tem se mostrado como um problema NP-Difícil, conforme afirma [Ceri e Pelagatti, 1985].

O problema de projeto de banco de dados físico é computacionalmente complexo desde que o subproblema de alocação dos atributos para registros também tem se mostrado ser NP-Difícil, devido a explosão da análise combinatorial encontrada se todas as possíveis combinações de acesso e distribuição de dados são consideradas uma solução ótima é difícil de encontrar.

Então, solução de aproximação (heurísticas) é considerada, os dados requisitados de entradas envolvem tempo-real e monitoramento online do banco de dados ou uma estimativa do perfil estatístico, fazendo de um sistema especialista a máquina ideal. O espaço de pesquisa para uma boa solução é restringido pela regras resultando numa solução eficiente.

Um sistema inteligente pode ser considerado um programa de domínio de aplicação específica, onde existe uma base de conhecimento suficiente para inferir uma ou mais soluções. Essa ambiente é caracterizado por:

- Uma base de conhecimento que contém conceitos, fatos, regras e a destreza do projeto físico de banco de dados;

- Uma máquina de inferência que consiste de um conjunto de técnicas para manipular a base de conhecimento

- Uma interface usuário externa na qual o usuário-final interage com o sistema.

O poder de um sistema especialista é caracterizado pela satisfação de uma base de conhecimento ser capaz de trabalhar eficientemente como um especialista. As características listadas acima se aplicam bem no domínio de projeto físico de banco de dados. A base de conhecimento é estatística junto com os parâmetros inicialmente providos pelos especialistas. A máquina de inferência contém regras para índices de seleção, particionamento de atributos, agrupamento de registros e alocação de arquivos.

Duas ferramentas comerciais são tratadas por Tewari, a RDT (Relational Desgin Tool – Ferramenta de projeto relacional) para SQL/DS da IBM Corporation e a RedExpert da DEC products.

Uma ferramenta de projeto de banco de dados físico deve ter as seguintes características: oferecer uma base de conhecimento evolutiva, aceitar especificações incompletas, deve ter justificativa e capacidades de explicação e permitir reverter qualquer passo. As ferramentas tratadas por Tewari possuem todas essas capacidades.

8.2.3.1 DBDSGN (RDT) para sistemas R (SQL/DS)

A DBDSGN é uma ferramenta de projeto físico experimental para o sistema R (protótipo de banco de dados relacional) que serve como base para a ferramenta comercial RDT para sistema de banco de dados SQL/DS. Essa ferramenta considera o seguinte problema: dado um conjunto de tabelas e um conjunto de acessos enunciado, junto com a expectativa de freqüência de uso, o índice de seleção do problema envolve selecionar para cada tabela:

- ordenar as regras para armazenamento de registro.
- um conjunto de índices não agrupados.
- minimizar o custo de processamento total (o custo total é definido como uma soma de custo de acesso, custo de alteração e índice de manutenção de custo)

Uma versão restrita do índice de seleção do problema tem sido apresentada como uma classe de problemas NP-Difícil, implicando que não há um procedimento ótimo de solução, a aproximação.

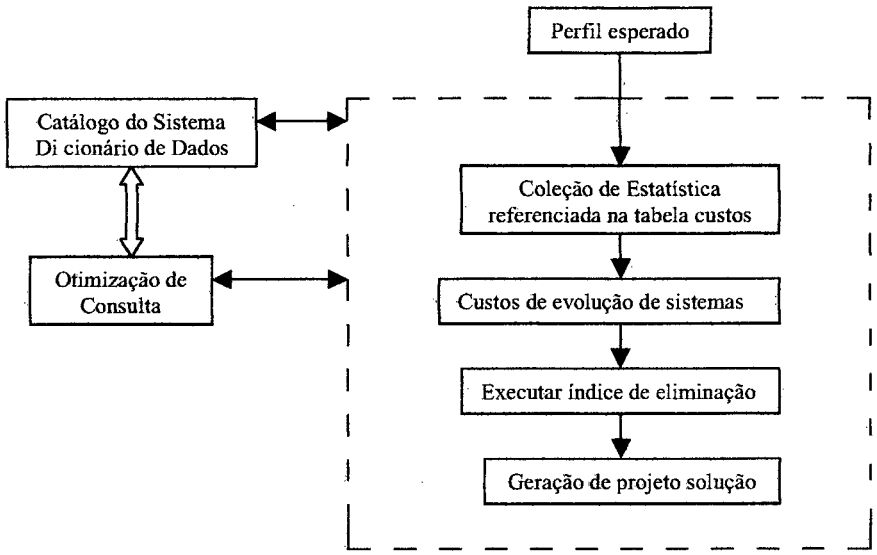


Figura 8.7 – Arquitetura DBDSGN/RDT

8.2.3.2 RdbExpert para DEC Rdb/DBMS

RdbExpert é uma ferramenta de projeto físico de banco de dados para a família de componentes DEC Rdb/DBMS. Ela trabalha em conjunto com o DECtrace para transações automáticas de banco de dados. Os dados obtidos são associados para a ferramenta DECtrace que é um evento baseado em perfil. Isto significa que os dados são coletados em locações especificadas nas aplicações, em todo o tempo que o código da aplicação é executado, sugerindo uma grande flexibilidade.

A estratégia dessa ferramenta para [Tewari, 1990] é integrar todos os subproblemas e solucionar o problema usando um procedimento de limite da ramificação para estimativa total de custo. Cada um dos subproblemas, como índice de seleção, agrupamento de registro, particionamento de atributo e locação são considerados separadamente inicialmente, subseqüentemente as possíveis soluções que melhoram a função objetivo são incrementalmente consideradas.

A próxima ilustração descreve a arquitetura da ferramenta RdbExpert.

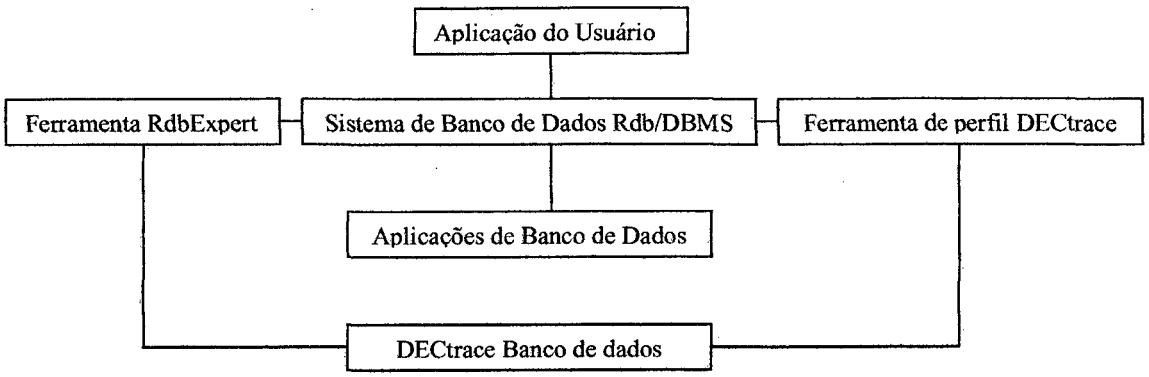


Figura 8.8 – Arquitetura RdbExpert

O quadro demonstrativo a seguir faz um resumo dos principais aspectos das metodologias apresentadas neste capítulo.

Características	**Metodologia para fragmentação híbrida**	**Metodologia proposta por Buretta**	**Metodologia DATAID-D**	**Metodologia proposta por Miranda**
Aplicação	Etapa de fragmentação para o projeto de BDD na abordagem *Top-down.*	Etapa de projeto de distribuição para o projeto de BDD na abordagem *Top-down.*	Etapa de projeto de distribuição para o projeto de BDD na abordagem *Top-down.*	Etapa de projeto de distribuição para o projeto de BDD na abordagem *Top-down.*
Técnicas utilizadas	Fragmentação horizontal e vertical.	Fragmentação horizontal e vertical, alocação dos dados.	Fragmentação horizontal e vertical, alocação dos dados.	Fragmentação horizontal e vertical, alocação dos dados.
Etapas identificadas	Agrupamento de atributos e predicados em conjuntos de fragmentos verticais e horizontais, reconhecimento de restrições e criação das células da tabela.	Reconhecimento do perfil da aplicação, da infra-estrutura física, recuperação e integração dos dados, esquema de alocação e implementação.	Projeto de distribuição, que compreende o particionamento, alocação redundante e não redundante e reconstrução de esquemas locais.	Aquisição de informações, escolha da solução inicial, entre a abordagem top-down e bottom-up, otimização e pós-otimização e geração dos esquemas físicos.
Implementada em			Instituto Politecnico di Milano	Instituto Militar de Engenharia (IME)
Ferramenta própria				SIM

Tabela 8.1 – Resumo das características das metodologias para projeto de BDD

A tabela 8.2 demonstra um resumo dos principais aspectos das ferramentas apresentadas neste capítulo.

Características	Designer/2000	Arquitetura DCMS	DBDSGN	RdbExpert
Tratamento de	Análise e requisitos funcionais, modelagem, formulários replicação simétrica, tratamento de cópias remotas, procedimentos de armazenamento.	Identificação e tratamento de restrições e inconsistências desde sua criação até a alteração e otimização	Estrutura física dos arquivos do banco de dados e seleção do caminho.	Estrutura física dos arquivos do banco de dados e seleção do caminho.
Produzida por	ORACLE®	Autores: Gupta e Tiwari [Gupta e Tiwari, 1991]	SQL/DS	DEC®
Implementada em	Webster University	Centro de Pesquisa IBM- Almaden	Laboratórios do desenvolvedor	Laboratórios do desenvolvedor

Tabela 8.2 – Resumo das características das ferramentas para projeto de BDD

8.3 Considerações Finais

Neste capítulo foram descritas algumas metodologias como para fragmentação híbrida e para o projeto de distribuição dos dados como a metodologia DATAID-D, a proposta de Buretta [Buretta, 1997] e Miranda [Miranda, 1999].

Outro item abordado foi sobre as ferramentas que auxiliam em diferentes aspectos do processo de projeto de banco de dados distribuídos, como a Designer/2000, a arquitetura DCMS, a DBDSGN e a RdbExpert.

O capítulo seguinte versa sobre as conclusões desse estudo, bem como sua relevância e perspectivas futuras.

Capítulo 9

CONCLUSÃO

9.1 Resumo do Trabalho

Entende-se por um sistema de banco de dados distribuídos um conjunto interligado de unidades computacionais, chamadas de *sites* ou *nós*, cada uma contendo um Sistema Gerenciador de Banco de Dados (SGBD), idênticos ou não, capazes de processar as transações locais e as transações globais que solicitam o acesso a dados localizados em outros nós.

As vantagens advindas dessa tecnologia devem ser garantidas na fase de projeto, pois qualquer deficiência nessa etapa pode ocasionar uma série de prejuízos indesejáveis pelas organizações.

A as abordagens de projeto de bancos de dados distribuídos se ramificam tanto na fase de criação como na fase de tratamento de dados existentes. Neste estudo, após um amplo levantamento bibliográfico, constatou-se a importância da realização de um bom projeto, bem como foram pesquisadas e analisadas algumas etapas metodológicas, metodologias implementadas, ferramentas e restrições. Todos esses itens foram abordados e distribuídos em capítulos conforme a relevância do assunto: projeto de banco de dados, o caso distribuído, particionamento dos dados, alocação dos dados e integração dos dados.

Inicialmente foi apresentado um capítulo introdutório sobre os conceitos de banco de dados distribuídos e por fim foi desenvolvido um capítulo sobre as ferramentas e metodologias para o projeto de distribuição.

A próxima seção traz as conclusões desse estudo.

9.2 Conclusões

Para usufruir os benefícios alcançados com bancos de dados distribuídos, como autonomia local, ininterrupção das operações, transparência, independência de hardware, sistema operacional e rede são necessários alguns investimentos, principalmente ao projetar a distribuição dos dados.

Não se pode ignorar as dificuldades que esse tipo de banco de dados traz, a complexidade em assegurar a coordenação entre os nós, toma a forma de aumento nos custos do desenvolvimento de aplicações, maior potencialidade de defeitos, diminuição da velocidade na obtenção das informações e aumento da sobrecarga de processamento.

Duas alternativas distintas de distribuição dos dados podem ser citadas no ambiente de SBDD, a homogênea e a heterogênea, é importante ressaltar as diferenças de procedimentos realizados quanto as abordagens utilizadas no projeto de cada uma das alternativas, a *top-down* e a *bottom-up*.

Uma abordagem *bottom-up* acontece normalmente no ambiente heterogêneo, quando bancos de dados preexistentes são integrados no intuito de formar um único e coeso multidatabase.

A palavra forte nessa etapa é integração, ela irá fornecer a interoperabilidade necessária aos bancos de dados individuais e autônomos, que passarão a responder em um nível mais alto, como um grande banco de dados heterogêneo.

Existem duas alternativas propostas para integração: com esquema global e sem esquema global. Na primeira alternativa, obtém-se um sistema fortemente acoplado, visto que o esquema global é a união dos esquemas locais.

De acordo com [Mânica, 2001] 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. A atenção redobrada deve ocorrer quando a heterogeneidade entre os esquemas a serem integrados for grande, nessa situação a opção torna-se inviável devido à dificuldade na automação e na adaptação a mudanças dos esquemas locais que quando ocorrem produzem uma nova integração.

Na outra alternativa o resultado é um sistema fracamente acoplado diante da integração sem um esquema global. Sendo que neste caso será responsabilidade do usuário criar e atualizar visões facilitando as mudanças dos esquemas componentes. O problema é que a transparência nessa situação não é total como no modelo com esquema global.

Apesar dos esforços desprendidos nos últimos anos, o cenário atual das pesquisas em sistemas de bancos de dados heterogêneos indica ainda não haver conhecimento suficiente que possibilite a definição de padrões para a questão de integração. 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.

Já a outra abordagem, a *top-down*, acontece normalmente em ambiente homogêneo, quando a criação do banco de dados distribuído parte do zero. Todo o trabalho é iniciado assim como no caso centralizado, na análise de requisitos, as diferenças ocorrem quando da passagem do projeto lógico para o físico, nesse momento ocorre o projeto de distribuição.

O projeto de distribuição nada mais é que a partir da formação do esquema conceitual global dá-se a distribuição de um grupo de informações sobre os sites formando o esquema conceitual local.

Esses grupos de informações são conhecidos como fragmentos, os problemas aparecem quando da aceitação de um fragmento como sendo uma unidade razoável de armazenamento. E considerando a distribuição de uma relação por unidades razoáveis o próximo problema é a alocação desses dados nos sites que compõe o BD.

Na fase do particionamento dos dados várias técnicas são conhecidas, a fragmentação horizontal, subdividida em primária e derivada, a fragmentação vertical e a fragmentação híbrida. Foram apresentados neste trabalho vários aspectos de cada uma das alternativas citadas, como grau, consistência e correção que compreende a reconstrutibilidade, completude e a disjuntividade.

No caso da fragmentação horizontal, foram descritos os algoritmos COM_MIN e o PHORIZONTAL, já da opção vertical constam o algoritmo de Agrupamento,

Particionamento e o Algoritmo Gráfico. A fragmentação híbrida ocorre da combinação das duas estratégias anteriores.

Na etapa de alocação dos dados, novos obstáculos foram identificados. Primeiramente a alocação dos fragmentos era tido como um problema de alocação de arquivo, logo se caracterizou como sendo um suposição incorreta por vários motivos, o principal deles é devido a forma complexa na qual as aplicações fazem uso de seus dados.

Ficou claro a divisão do tradicional problema de alocação de arquivo do problema de alocação do banco de dados, este cenário envolve ainda procurar a distribuição ótima dos fragmentos através dos sites.

Dois aspectos da questão da otimização podem ser considerados importantes, a performance e o custo mínimo. O trabalho mostra modelos de fórmulas que analisam esses aspectos, todas as informações participantes são baseadas nos requisitos de informações retirados da aplicação, da rede de comunicação, da capacidade de processamento, limitações de armazenamento e do banco de dados.

Outros itens abordados em alocação dos dados foram sobre ambiente estático e dinâmico, estratégia de armazenamento e medidas de custo/benefício.

A disponibilidade, uma característica muito importante de um BDD, é provida pela replicação dos fragmentos alocados nos sites. Em [Ceri e Pelagatti, 1985] fica evidente a necessidade de distinguir a implementação de uma alocação com redundante ou não-redundante.

Para ambos os casos a pesquisa apresenta alguns métodos conhecidos, o método *"best-fit"* é o que melhor se ajusta para a situação de não-redundância. A redundância introduz uma complexidade na etapa de projeto já que o grau de replicação se torna uma variável do problema.

Os métodos "ABS" e "alocação progressiva" fazem parte da opção com redundância.

A relação custo/benefício da decisão sobre alocação pode ser estimada considerando custos de armazenamento, custos de comunicação (nos casos de

consultas e atualizações), e a disponibilidade dos dados. Algumas questões já são conhecidas:

- O custo de comunicação das consultas diminui quando o número de cópias aumenta, pois muitos dados podem ser encontrados em vários sites, eliminando a necessidade de comunicação entre elas;

- O custo de comunicação da atualização aumenta de acordo com o número de cópias existentes, visto que é imprescindível atualizar todos os fragmentos;

- O custo de armazenamento aumenta de acordo com o aumento do número de cópias;

- A disponibilidade dos dados aumenta com o aumento do número de cópias.

Uma "solução ótima" seria determinar o custo mínimo considerando (armazenamento + comunicação + processamento local) [Teorey, 1999].

Também foram descritos modelos de fórmulas que mensuram o custo/benefício para alocação de fragmentos.

Após o tratamento dos ambientes de distribuição, algumas etapas metodológicas [Buretta, 1997], metodologias e ferramentas foram descritas. Neste estudo foram descritas a metodologia DATAID-D e a metodologia para projeto de bancos de dados distribuídos proposta pelo IME - Instituto Militar de Engenharia.

Como ferramentas foram apresentadas a Designer/2000 aplicada em um estudo de caso na Webster University, a arquitetura de um Gerenciador de Restrições no Projeto de Banco de Dados Distribuídos (DCMS), a ferramenta inteligente para o projeto físico de banco de dados como o DBDSGN (RDT) para sistemas R (SQL/DS) e a RdbExpert para DEC Rdb/DBMS.

O aumento das pesquisas indicam que não há um padrão na qual se enquadrem empresas no que diz respeito a metodologias ou ferramentas, não existe no mercado

uma abordagem ampla que inclua todos os passos desde a metodologia até a ferramenta para auxiliar nessa complexa tarefa.

As soluções existentes são aplicadas a ambientes específicos como a do Instituto Militar de Engenharia e a Webster University, e não soluções gerais que podem ser aplicadas a qualquer organização dispersa em áreas geográficas.

[Buretta, 1997] afirma que grandes empresas desenvolvedoras como IBM, Oracle e Sybase tem investido em pesquisas no intuito de produzir mais soluções, o mesmo acontece com pesquisas acadêmicas que fazem com que a busca de novas soluções nesta área cresça a cada dia.

9.3 Relevância do Trabalho

Com o advento da utilização de bancos de dados distribuídos surgiu a necessidade de procedimentos exatos na fase de projeto, visto que as empresas que investiram nessa tecnologia precisam de garantias de que os investimentos terão retorno. Nesta etapa, a maior dificuldade é como proceder a distribuição e integração dos dados.

Este estudo retrata uma reunião compilada das várias técnicas, metodologias e ferramentas para projeto de banco de dados distribuídos na visão de diferentes autores. Sendo que a partir do material apresentado poderão surgir novas propostas respeitando as diversas restrições impostas pelo ambiente distribuído.

9.4 Perspectivas Futuras

Assim como o advento da utilização de bancos de dados distribuídos exigiu novos procedimentos na etapa de projeto, novas funções são exigidas para as novas perspectivas que surgiram a partir de então.

O ambiente tratado neste trabalho assume que as informações depois de distribuídas ou integradas estão estaticamente disponíveis às aplicações. Acontece

que o cenário não é sempre esse, é necessário admitir que os dados têm se tornado dinâmico para atender as exigências dos usuários.

Após a realização deste trabalho, uma nova perspectiva de linha de pesquisa surgiu, ela está relacionada a performance de projeto de banco de dados distribuídos para o ambiente dinâmico.

Parâmetros importantes são reconhecidos nessa situação como número de nós, número de transações, localização dos dados, distribuição de transação, tipo de rede de comunicação, todas essas questões têm se tornado assunto de estudo devido a sua relevância.

GLOSSÁRIO

1FN – primeira forma normal

2FN – segunda forma normal

3FN – terceira forma normal

4FN – quarta forma normal

5FN – quinta forma normal

AA – matriz de afinidade de atributos

AAIS - sistema de informação sobre grupos acadêmicos

ABS - método conhecido como "benéfico a todos os sites"

AC - custo de acesso

AM – matriz de afinidade global

BDD – banco de dados distribuídos

BEA – algoritmo de ligação de energia

BOTTOM-UP– uma das abordagens na estratégia de projeto de banco de dados distribuídos, normalmente utilizado quando da integração de vários bancos de dados existentes.

BQ – conjunto de aplicações que acessam determinados atributos

CA – matriz de afinidade agrupada

CBQ – conjunto de aplicações que acessam determinados atributos

CC- controle de concorrência

CLIENT/SERVER– cliente/servidor

COQ – conjunto de aplicações que acessam determinados atributos

DAP - problema de alocação de banco de dados

DCMS - sistema gerenciador de restrições distribuídas

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

DER – diagrama entidade-relacionamento

ECG – esquema conceitual global

ECL– esquema conceitual local

EE – esquema externo

EIL – esquema interno local

ER– entidade-relacionamento

FAP - problema de alocação de arquivo

IBM – marca registrada, empresa desenvolvedora de software

IE - execução de integridade

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

MER– modelo entidade-relacionamento

MULTIDATABASE– sistema de múltiplos bancos de dados.

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

O(n) - algoritmo com complexidade linear.

$O(n^2)$ – algoritmo com complexidade exponencial.

OQ – conjunto de aplicações que acessam determinados atributos

ORACLE– marca registrada, empresa desenvolvedora de software

PARTITION - algoritmo de particionamento

PC- custo de processamento

PEER-TO-PEER– ponto-a-ponto

RM - matriz

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.

SGBDDH – sistema gerenciador de banco de dados distribuído heterogêneo.

SGBDH – sistema gerenciador de banco de dados heterogêneo.

Sybase – marca registrada, empresa desenvolvedora de software

TA – atributos do topo

TC - custo de transmissão

TOP-DOWN – uma das abordagens na estratégia de projeto de banco de dados distribuídos, normalmente utilizado quando do desenvolvimento inicial de bancos de dados.

TQ – conjunto de aplicações que acessam determinados atributos

UM - matriz

REFERÊNCIAS BIBLIOGRÁFICAS

[Apers, 1988] APERS, P. M. **Data Allocation in Distributed Database Systems**. ACM Transactions on Database Systems, Vol. 13, Nº. 3, Setembro de 1988, Pag. 263 - 304.

[Batini et alii, 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.

[Buretta, 1997] BURETTA, M. **Data Replication: tools and techniques for managing distributed information**. Wiley, 1997.

[Ceri e Pelagatti, 1985] CERI, S.; PELAGATTI, G. **Distributed Databases Principles & Systems**. McGraw-Hill, 1985

[Chakravarthy et al., 1992] CHAKRAVARTHY, S. et al. **An Objective Function For Vertically Partitioning Relations in Distributed Databases and its Analysis**. Department of Computer and Information Sciences - University of Florida – Gainesville, 1992.

[Davenport, 1981] DAVENPORT, R. **Design of Distributed Data Base Systems**. Comput: 1981.

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

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

[Gupta e Tiwari, 1991] GUPTA, A; TIWARI, S. **Constraint Management on Distributed Design Databases**.

[Hernandez, 2000] HERNANDEZ, M. **Aprenda a Projetar seu próprio Banco de Dados**. São Paulo: Makron Books, 2000.

[Heuser, 1998] HEUSER, C. A. **Projeto de Banco de Dados**. Porto Alegre: Sagra Luzzatto, 1998.

[Karlapalem e Pun, 1995] KARLAPALEM, K.; PUN, N. M. **Query Driven Data Allocation Algorithms for Distributed Database Systems**. Department of Computer Science – University of Science and Technology : Hong Kong, 1995.

[Kern, 1994] KERN, M. V. **Banco de dados relacionais: teoria e prática de projeto**. São Paulo: Éruca, 1994. 228p.

[Korth e Silberschatz, 1995] KORTH, H; SILBERSCHATZ, A. **Sistema de Banco de Dados**. São Paulo: Makron Books, 1995.

[Machado e Abreu, 1995] MACHADO, F. R. N.; ABREU, F. **Projeto de Banco de Dados: uma visão prática**. São Paulo: Érica, 1995. 298.

[Mânica, 2001]. MÂNICA H. **Bancos de Dados Distribuídos Heterogêneos: Arquiteturas, Tecnologias e Tendências**. Dissertação de Mestrado, Departamento de Ciência da Computação. Universidade Federal de Santa Catarina, 2001.

[Mesquita e Finger, 1998] MESQUITA, J. E.; FINGER, M.. **Projeto de dados em Banco de Dados Distribuídos**. Florianópolis : XIII Simpósio Brasileiro de Banco de Dados 1998.

[Mesquita, 1998] MESQUITA J. E. **Projeto de Dados em Banco de Dados Distribuídos**. Dissertação de Mestrado, Departamento de Ciência da Computação, Instituto de Matemática e Estatística da Universidade de São Paulo: 1998.

[Miranda, 1999] MIRANDA, G.M. **Uma Metodologia para Projeto de Bancos de Dados Distribuídos**. Dissertação de Mestrado, Instituto Militar de Engenharia: 1999.

[Navathe e Ra, 1989] NAVATHE, B. S.; RA, Minyoung. **Vertical Partitioning for Database Design: A Graphical Algorithm**. ACM Computer Survey, Dezembro, 1989.

[Navathe et al., 1994] NAVATHE, B. S.; KARLAPALEM, K.; RA, M.. **A Mixed Fragmentation Methodology for Initial Distributed Database Design.** Database Systems R&D Center – University of Florida – Gainesville, 1994.

[Navathe, Ceri e Wiederhold, 1984] NAVATHE, B. S.; CERI, S.; WIEDERHOLD, G. **Vertical Partitioning Algorithms for Database Design.** ACM Computer Survey, Vol. 9, n 4, Dezembro, 1984.

[Owens e Sheldon, 1998] OWENS. D.; SHELDON, F. **Tool-Based Approach to Distributed Database Design.** ACM Computer Survey, 227-231, December 1998.

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

[Teorey, 1999] TEOREY, T. **Database Modeling & Design.** San Francisco: Morgan Kaufmann, 1999.

[Tewari, 1990] TEWARI, R. **Expert Design Tools for Physical Database Design.** ACM Computer Survey, 538-550, 1990.

[Yan et alii, 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 alii, 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.