PROPOSTA DE UM SISTEMA DE MANIPULAÇÃO DE ARQUIVOS EM UM AMBIENTE DISTRIBUÍDO

FRANCISCO EDMAR AGUIAR PEREIRA

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.).

Aprovada por:

LUIZ ANTONIO CARNEIRO DA CUNHA COUCEIRO
Presidente

SUELI MENDES DOS SANTOS

EBER ASSIS SCHMITZ

YSMAR VIANNA E SILVA FILHO

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 1982

PEREIRA, FRANCISCO EDMAR AGUIAR

Proposta de um Sistema de Manipulação de Arquivos em um Ambiente Distribuído [Rio de Janeiro] 1982.

xi, 108 p., 29,7cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1982).

Tese - Universidade Federal do Rio de Janeiro, Programa de Engenharia de Sistemas.

1. Assunto I. COPPE/UFRJ II. Título (Série)

Ao potencial da minha família
Valéria, Edmar Neto e Samya
com muito amor.

AGRADECIMENTOS

- Ao professor Luiz Antonio C. C. Couceiro pela brilhante orientação e, além de tudo, pela amostra de que uma pesquisa sadia se faz com amizade, dedicação, tolerância e liberdade de expressão.

- Aos membros da banca examinadora, profs. Eber Assis Schmitz, Ysmar Vianna e Silva Filho e Sueli Mendes dos Santos.

- A todos os professores que compõem o quadro docente do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ.

- Aos colegas Araruna, Arlindo, Chico, Clécio e Paulo César pela amizade e contribuições expontâneas que me dedicaram durante a efetivação deste trabalho.

- À Da. Heliêta, Eliana e ao Maurício pelos inúmeros favores prestados desinteressadamente.

- Às bibliotecárias do Centro de Tecnologia e do Núcleo de Computação Eletrônica da UFRJ.

- À Norma, pelo excelente trabalho de datilografia.

- A todos aqueles que influiram direta ou indiretamente neste trabalho.

AGRADECIMENTOS ESPECIAIS

Aos meus pais, Edmar de Castro Pereira e Maria Valderiza Aguiar Pereira, pois sem eles nada disso seria possível.

Ao Departamento de Estatística e Matemática Aplicada da UFCe pelos incentivos moral e financeiro que muito contribuíram para efetivação deste trabalho.

RESUMO

A idéia deste trabalho surgiu da necessidade de se oferecer recursos aos usuários de processamento de dados para o desenvolvimento de aplicações em ambiente distribuído, ou seja, obter informações das bases de dados existentes em locais geograficamente dispersos e conectados através de uma rede de comunicação de dados. Propõe-se o desenvolvimento de ferramentas de "software" que permitam a transferência total ou parcial de arquivos entre estes locais, através de uso de um conjunto de comandos facilmente assimilável por pessoas não especializadas em computação.

A B S T R A C T

The present work has been motivated by the need of offering resources to data processing users for developing applications in a destributed environment. In other words, collect information from data bases operating in different places and connected through a network of data communication. A development of special software tools is proposed which enables transfering whole or parts of files among these places, through the application of a set of instructions that can be easily assimilated by non specialized users.

ÍNDICE

CAPÍTULO I - INTRODUÇÃO

I. INTRODUÇÃO

Uma empresa, de maneira geral, necessita manipular uma variedade de informações no seu dia-a-dia. Isto, de certa forma, gera uma preocupação de como encontrar a melhro maneira de organizar estas informações.

Tratando-se de empresas de médio e grande porte, o processamento de dados com o uso de computadores eletrônicos é, sem dúvida, a solução mais comumente utilizada em nossa época. No entanto, para as empresas de pequeno porte, o uso de processamento eletrônico de dados tornou-se viável devido a facilidade e o baixo custo introduzido nestes últimos anos com o advento mini e micro-computadores.

De uma forma ampla, a introdução de micro e mini-computadores, oferecendo um atrativo fator de custo/benefício, viabilizou o emprego de novas opções de processamento, isto é, a descentralização e a distribuição do processamento de dados.

Os sistemas de informação em sua concepção tradicional são implementados pelo binômio programas/arquivos - cada programa recebe e fornece dados para diversos arquivos. Freqüentemente, por mais acurada que tenha sido a fase de análise, após algum tempo de vida útil, tais sistemas necessitam ser reestruturados - novas aplicações foram desenvolvidas e novos dados, incluídos. Em certos casos a reprogramação é inviável economicamente e parte-se, então, para a adaptação do sistema as mudanças havidas.

A adaptação, normalmente, introduz redundância de informação que, por sua vez, implica perda de integridade do dado no instante de sua atualização - enquanto uma cópia já está atualizada outras podem estar ainda com o valor antigo.

Uma solução alternativa para o problema de estruturação das informações, é a introdução de um sistema de gerência de banco de dados o que permite um certo grau de independência dos dados em relação às aplicações e possíveis alterações envolvendo a base de dados. No Capítulo II deste trabalho, procuramos focalizar os tipos de processamento - centralizado, descentralizado e distribuído - e os envolvimentos de banco de dados para cada um desses tipos. No Capítulo III, faremos uma abordagem sobre as três principais técnicas de se organizar um banco de dados - hierárquico, rede e relacional - sem, contudo, apontarmos nenhuma estrutura como sendo a melhor opção. Isto se dá ao fato de acreditarmos que a escolha do modelo como ideal, está diretamente relacionada com o tipo de aplicação.

No entanto, dentro do contexto de processamento de dados distribuído - onde as bases de dados estão despersas geograficamente - será proposta uma alternativa, nos Capítulos IV e V, de como um usuário de um ambiente distribuído poderá comunicar-se com as bases de dados - ou conjunto de arquivos - de uma maneira simples e de fácil uso, isto é, sem necessitar de informações de como elas estão logicamente estruturadas bem como a localização física dos dados no ambiente distribuído.

O problema da localização física dos dados pode

ser resolvido com o uso de um diretório de dados para o sistema global. Para isto, apontamos diferentes maneiras de se alocar este diretório em um ambiente distribuído. Para as diferentes estruturas dos dados, é proposto o desenvolvimento de procedimentos exclusivos, para cada estrutura distinta, os quais sejam capazes de recuperar os conjuntos de dados existentes em cada uma delas. Desta forma, o usuário poderá obter dados do ambiente distribuído, através de uma linguagem geradora de arquivos auxiliares, visando ao atendimento de suas aplicações.

CAPÍTULO II - A EMPRESA E O PROCESSAMENTO DE DADOS

II. A EMPRESA E O PROCESSAMENTO DE DADOS

II.1 - Introdução

O objetivo deste capítulo é focalizar as principais situações de processamento dos dados nas empresas e, em seguida, fornecer informações gerais sobre banco de dados.

II.2 - Os Dados na Empresa

Uma empresa, de maneira geral, está dividida em órgãos comumente chamados de departamentos. Estes, para executar suas funções, necessitam coletar e manter em arquivos determinadas informações indispensáveis ao bom andamento de seu trabalho.

Com freqüência, uma informação pode estar sendo utilizada por mais de um departamento ao mesmo tempo, estando arquivada, deste modo, em locais distintos, o que denominamos de redundância ou repetição de informações.

Se uma informação, desta natureza, sofrer alguma alteração em um dos departamentos que a mantém, outro departamento, que dela se utiliza, pode deixar de atualizá-la, o que provoca inconsistência entre arquivos do mesmo tipo em locais distintos. Também é muito comum que uma informação seja alterada ou introduzida por pessoas não autorizadas ou qualificadas para tal operação.

II.2.1 - Processamento Centralizado

Em geral, os sistemas centralizados caracterizam-se pela existência das seguintes condições:

a) concentração de tarefas de processamento de dados nos centros de processamento eletrônicos;

b) equipamentos de médio ou grande porte, de propósito geral;

c) inexistência de comunicação com outros sistemas durante a execução de aplicações;

d) equipe especializada em computação eletrônica não vinculada aos usuários das informações processadas, mas ao centro de processamento de dados;

e) certo nível de autonomia do centro de processamento de dados em relação à corporação da empresa.

Devido a esta situação, os problemas abordados no item anterior podem continuar existindo mesmo que a empresa centralize o processamento de suas informações em um centro de computação eletrônica, no qual os arquivos existentes, gravados em fitas ou discos magnéticos, pertençam a sistemas projetados para atender aos departamentos separadamente. Estes arquivos estarão ligados a determinados programas que, por sua vez, estarão presos

às características físicas e lógicas destes arquivos. Qualquer mudança em uma destas características, implicará alterações nos programas, podendo causar erros de sérias complicações para o sistema a um custo muito elevado.

Este modelo de realização das tarefas de processamento de dados é implementado na maioria das empresas que se utilizam de computadores eletrônicos, fazendo com que o Centro de Processamento de Dados seja reconhecido como um órgão prestador de serviços.

No entanto, devido à pressão dos usuários para ter participação mais efetiva no processamento de suas aplicações e ao surgimento do mercado de computadores de pequeno porte, de baixo custo e performance aceitável, os sistemas de grande porte, centralizadores de funções, têm introduzido ou expandido as facilidades de acessos remotos, oferecendo recursos, como macro-linguagens por exemplo, de fácil entendimento e manuseio por parte dos usuários finais dos dados.

II.2.2 - Processamento Descentralizado

Neste caso deixa de existir a concentração de tarefas de processamento dos dados em uma única instalação de computação eletônica. A empresa dispõe de pequenos centros de processamentos, em locais geograficamente distintos, com divisão de tarefas.

Segundo Scherr[17], um sistema de processamento deve ser dito: (1) descentralizado, quando os centros de processa-

mentos - nós - não são cooperantes, ou seja, não trocam informações entre si durante a execução de tarefas; (2) distribuído, quando os nós são cooperantes - trocam informações durante a execução de tarefas.

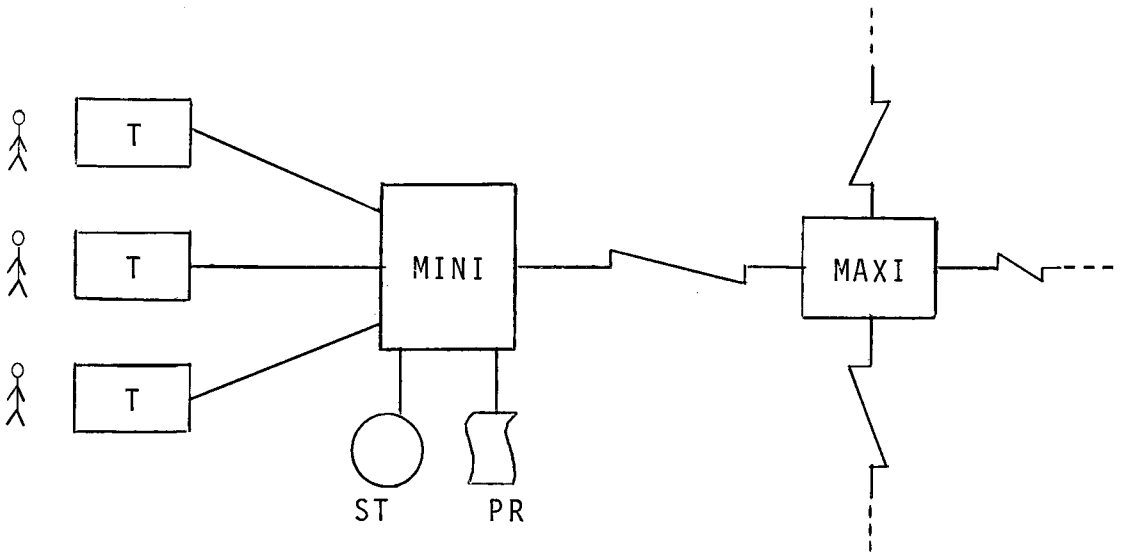
Como uma empresa, de maneira geral, está subdividida em órgãos dispersos geograficamente e que necessitam trocar informações para execução de suas tarefas, assemelha-se muito com a filosofia de sistemas distribuídos que é, também, separar subsistemas para serem administrados e processados em lugares geograficamente distintos com nós cooperantes. Vale a pena lembrar de que esta é uma condição necessária, mas não suficiente, para que uma empresa venha a optar por um sistema de processamento distribuído.

As figuras (II.1(a) e II.1(b)) mostram dois, entre outros, tipos de sistema descentralizado, nos quais:

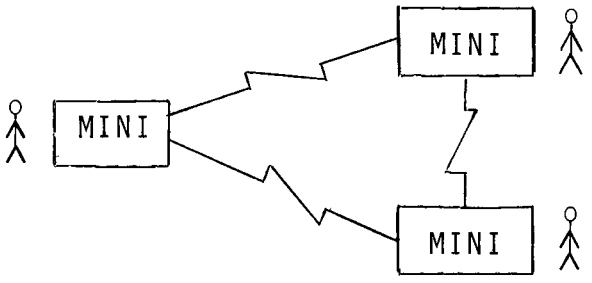
a) o primeiro tipo envolve a combinação de minicomputadores com um computador de grande porte. Neste tipo de sistema a filosofia de processamento é executar pequenas tarefas nos minicomputadores, enquanto tarefas mais complexas e outras que por razões administrativas devem ser centralizadas, são executadas no computador de grande porte;

b) o segundo tipo envolve uma rede de mini e/ou microcomputadores interligados com divisão de tarefas. Este caso pode envolver "load sharing"

entre os elementos ativos.



(a) MINI-MAXI



(b) MULTIPLO-MINI

Figura II.1 - Dois Tipos de Sistemas Distribuídos

II.3 - Banco de Dados

Um banco de dados é composto por uma base de dados e um sistema responsável pelo gerenciamento e manipulação desta base de dados.

II.3.1 - Base de Dados

É formada por um aglomerado de informações, logicamente organizadas, que se relacionam entre si.

Estas informações devem ser organizadas de maneira tal que possam atender eficientemente, a uma grande variedade de usuários, profissionais ou não na área de processamento de dados.

II.3.2 - Sistemas de Gerência de Banco de Dados (SGBD)

São instrumentos fabricados por "software" especialmente preparado com a finalidade de gerenciar e manipular eficientemente uma base de dados atentando, entre outros, para os seguintes aspectos:

a) o modelo deve integrar uma coleção de dados que seja utilizável por ampla variedade de usuários;

b) deve primar pela qualidade e integridade dos dados existentes;

c) devem ser tomados cuidados com a privacidade e segurança dos dados;

d) deve permitir um controle centralizado da base de dados, o que se faz necessário para uma efi

ciente administração;

e) deve ter capacidade para programas de aplicação e para que usuários ocasionais executem as funções de pedidos de informações, manipulação de dados, definição de dados e facilidades de controle dos dados.

II.3.2.1 - Levantamento e Independência dos Dados

O levantamento ou coleta dos dados que irão compor banco de dados deve ser efetuado em conjunto com o administrador do banco de dados e o pessoal responsável pelo fornecimento e operação dos resultados obtidos no dia-a-dia da empresa.

Esta integração administrador-usuário permite conscientização das reais necessidades da empresa em relação ao processamento de suas informações.

Caso a empresa disponha de um sistema mecanizado em computador eletrônico, os arquivos existentes devem ser trabalhados e selecionados com cuidado para não tornar deficiente o sistema de gerência de banco de dados, tendo em vista que estes tipos de arquivamentos, na sua grande maioria, não são condizentes com as estruturas físicas e lógicas dos SGBD's.

O conceito de independência dos dados é dirigido ao acesso à base de dados. Os usuários ocasionais e programas de aplicação devem ver, de forma transparente, os aspectos específicos quanto à organização lógica, organização física e considerações quanto ao armazenamento do banco de dados no computador.

Por independência física, entende-se a capacidade de isolar as aplicações das mudanças na organização física dos dados utilizados; por exemplo, mudanças na localização dos dados (entre os periféricos), ligações internas dos dados, estratégias de ordenação interna, mudanças no acesso às várias partes do banco de dados, etc. Independência lógica é a capacidade de as aplicações que se utilizam dos dados não sofrerem alterações em conseqüência de mudanças na organização lógica do banco de dados em uso.

II.3.2.2 - Qualidade e Integridade dos Dados

Dada uma variedade de procedimentos utilizando-se de um banco de dados, a integridade deve ser observada quanto aos seguintes aspectos: coordenação dos acessos aos dados por diferentes aplicações; propagação de alteração de valores em outras cópias e valores dependentes; preservação de elevado grau de consistência e correção dos dados. Com diferentes usuários processando variadas porções do banco de dados, lhe é impossível responsabilizar-se pela consistência dos dados e pela manutenção dos relacionamentos entre itens de dados, mesmo porque podem ser desconhecidos do usuário ou mesmo ser proibido de acessá-los. Uma das maiores razões de um sistema de banco de dados é manter o controle e preservar a integridade da base de dados.

II.3.2.3 - Privacidade e Segurança

A segurança e a privacidade estão ligadas à existência de um mecanismo de controle para as operações de leitura, inserção, remoção e alteração, não permitindo que usuários não autorizados possam efetuar qualquer uma destas operações em rela-

ção aos dados que compõem a base de dados, acidental ou maliciosamente.

II.3.2.4 - Administração e Controle

O ingrediente fundamental para introdução de um banco de dados em qualquer empresa é a função envolvendo o projeto, a administração e o controle do banco de dados. A responsabilidade pela descrição e controle dos dados, não deve ser difundida entre os vários usuários e analistas. Ela deve ser centralizada e ficar sob a responsabilidade de um administrador de banco de dados (ABD). O ABD deve ser um experiente e altamente qualificado indivíduo (ou grupo de indivíduos) em relação à estrutura da empresa e ao seu sistema de informações.

O projeto global do banco de dados, a definição dos dados e as atribuições de acessos à base de dados por parte dos usuários devem ser acompanhados pelo ABD colocando sempre em pauta a performance e eficiência do sistema.

II.3.2.5 - Capacidade para Programas de Aplicação

Os primeiros sistemas de gerência de banco de dados disponíveis no mercado eram orientados para profissionais, técnicos, em processamento de dados, esquecendo-se do usuário final dos dados, o qual, na maioria dos casos, pouco ou nada conhece sobre o assunto.

Como estes sistemas estão cada vez mais abrangentes e devido ao fato de que os dados manipulados são valiosos re

cursos que devem ser compartilhados por diversos setores de uma empresa, a participação do usuário final tornou-se um ponto de apoio às pesquisas, para oferecer, a estes usuários, condições de consulta à base de dados com o menor esforço possível em relação a conhecimentos computacionais.

O acesso a qualquer parte do banco de dados deve ser possível: (1) via uma linguagem de consulta de alto nível e autocontida; (2) via declarações de entrada/saída, usadas em programas escritos e em qualquer linguagem de programação convencional.

II.4 - Banco de Dados Distribuído

Um banco de dados é dito distribuído quando um banco de dados integrado logicamente é distribuído entre vários, fisicamente distintos mas interligados, centros de processamento de dados.

II.4.1 - Banco de Dados Distribuído Homogêneo

É um banco de dados distribuído em que são idênticos os vários sistemas de gerência de banco de dados existentes nos centros de processamento de dados de uma rede de computadores. O "hardware" de cada centro é também idêntico e permite mais geralmente que os sistemas de gerência de banco de dados cooperem entre si.

II.4.2 - Banco de Dados Distribuído e Heterogêneo

É um banco de dados distribuído em que são distin

tos os vários sistemas de gerência de banco de dados existentes nos centros de processamento de dados de uma rede de computadores. Os "hardwares" de cada centro podem ser distintos ou não, contanto que o "software" seja diferente. Os nós podem ser cooperantes.

II.4.3 - Distribuição dos Dados

Um banco de dados distribuído - homogêneos ou heterogêneo - pode ter sua base de dados duplicada ou particionada.

II.4.3.1 - Base de Dados Duplicada

Nesse caso, a base de dados é total ou parcialmente duplicada pelos dois ou mais nós da rede. A figura (II.2) mostra um banco de dados distribuído em que todas as ocorrências

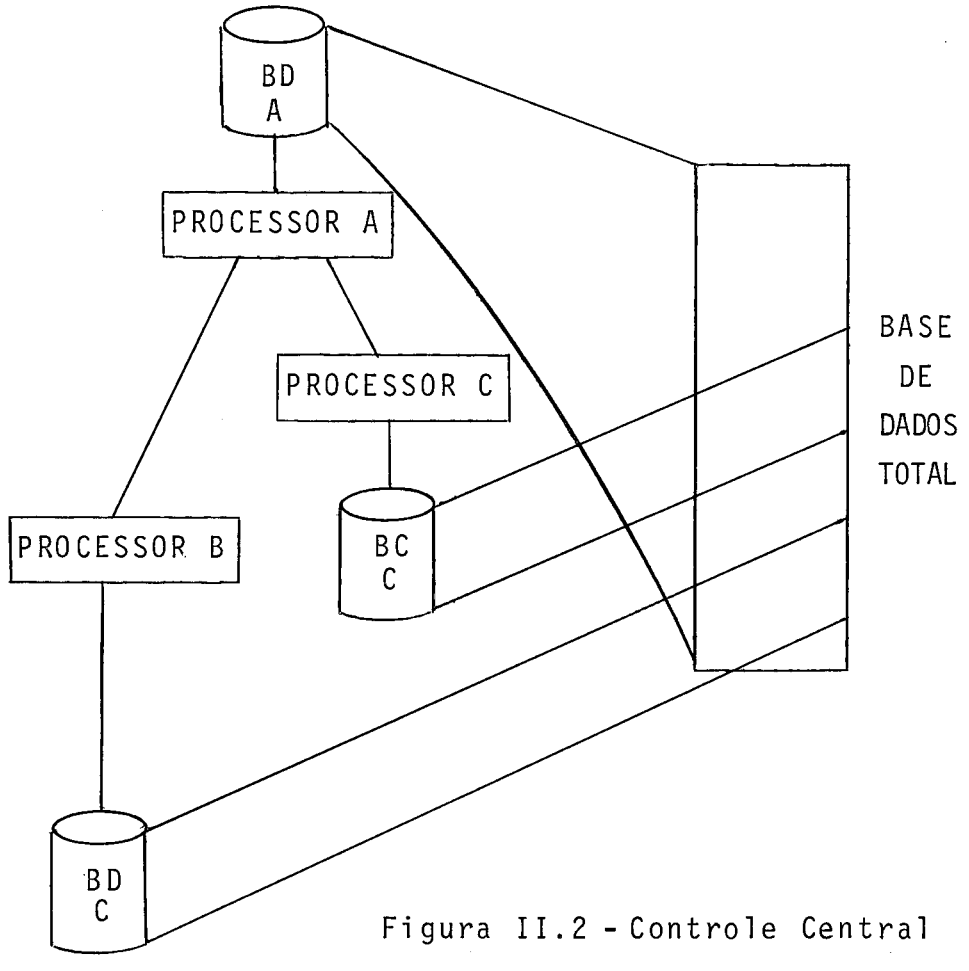


Figura II.2 - Controle Central

de cada tipo de dado estão disponíveis no periférico de armazenagem pelo centro A, enquanto só algumas das ocorrências estão disponíveis pelos centros B e C. Uma duplicação da base de dados pode ser total - cada localização pode ter a mesma estrutura de dados - ou parcial - somente um centro contém a estrutura completa, enquanto os outros centros contêm somente subconjunto desta estrutura, como é mostrado nas figuras (II.3.(a e b)) e (II.4.(a, b e c)), respectivamente.

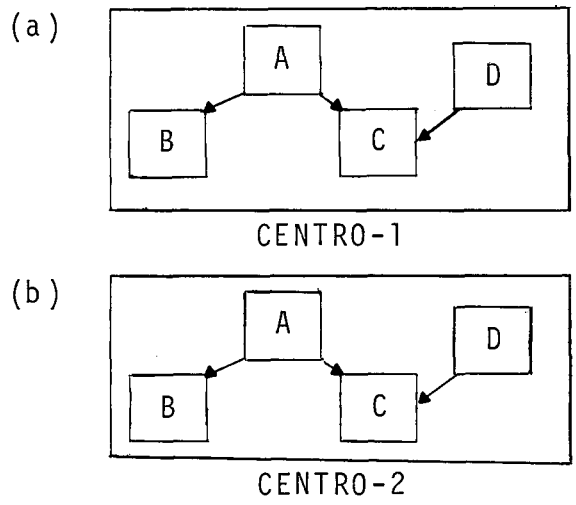


Figura II.3 - Base de Dados Duplicada Totalmente ou Duplicação por Estrutura

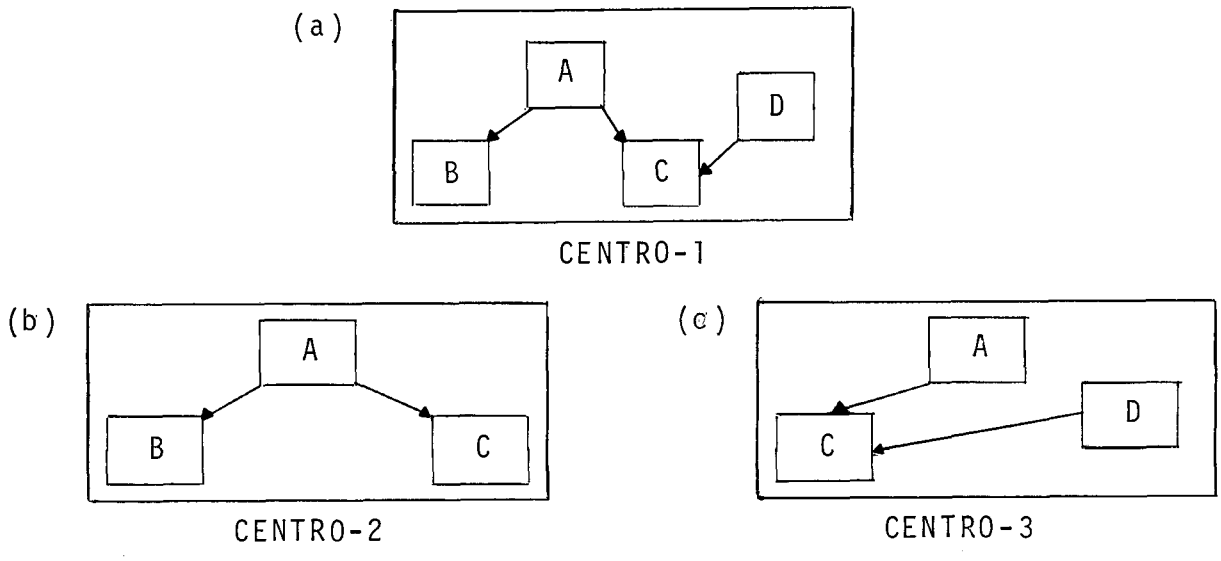
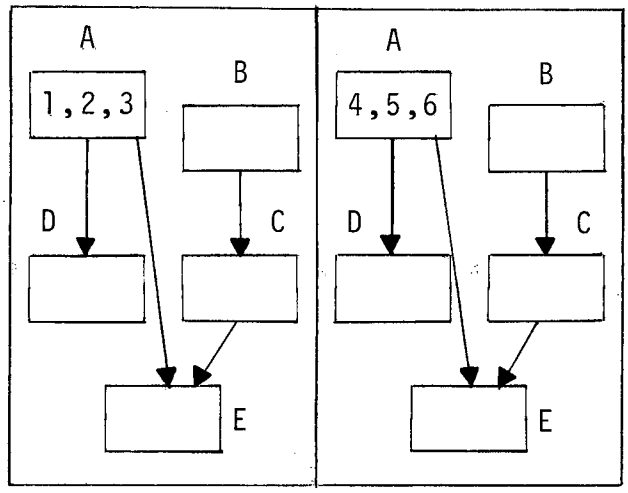


Figura II.4 - Base de Dados Duplicada Parcialmente

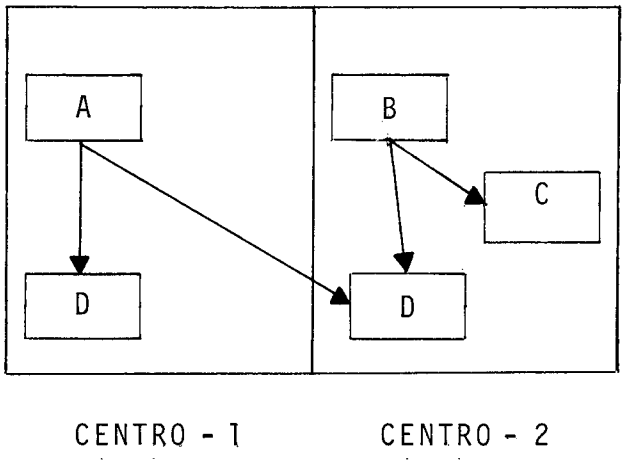
II.4.3.2 - Base de Dados Particionada

Em uma base de dados particionada não existe duplicação dos dados armazenados pelos vários nós da rede. O particionamento pode ser por ocorrência ou por estrutura. Em particionamento por ocorrência, ocorrências específicas são alocadas em localizações (centros) específicos: por exemplo, se tomarmos a relação A - ou conjunto X - com ocorrências 1, 2 e 3 (ou registros membros 1, 2 e 3), estas podem ser armazenadas pelo centro-1, enquanto as ocorrências 4, 5 e 6 podem ser armazenadas pelo centro-2. Em particionamento por estrutura, relações ou conjuntos são alocados em locais distintos: por exemplo, as relações - ou conjuntos - A e E alocados no centro-1 e B, C e D alocados no centro-2. As figuras (II.5.a) e (II.5.b) exemplificam os dois casos, respectivamente.

As bases de dados duplicada e particionada não são, necessariamente, mutuamente exclusivas. Os dois tipos de distribuição de dados podem ser representados em uma implementação particular. Por exemplo, a figura (II.2) mostra uma duplicação da base de dados entre A e B, entre A e C e entre B e C e, também, exibe um certo grau de particionamento entre as bases de dados B e C.



(a) Particionamento por valor



(b) Particionamento por estrutura

Figura II.5 - Base de Dados Particionada

CAPÍTULO III - MODELOS DE SISTEMAS DE GERÊNCIA DE BANCO DE DADOS

III. MODELOS DE SISTEMAS DE GERÊNCIA DE BANCO DE DADOS

III.1 - Introdução

Neste capítulo nos propomos a fazer uma abordagem - discreta - das principais formas de se modelar logicamente uma base de dados.

III.2 - Estrutura Geral de um SGBD

Antes das conceituações dos modelos, mostraremos na figura III.1 um fluxo generalizado, de como procede um sistema de gerência de banco de dados ao ser referenciado pelos comandos de I/O. Após o fluxo, descreveremos cada um dos seus passos.

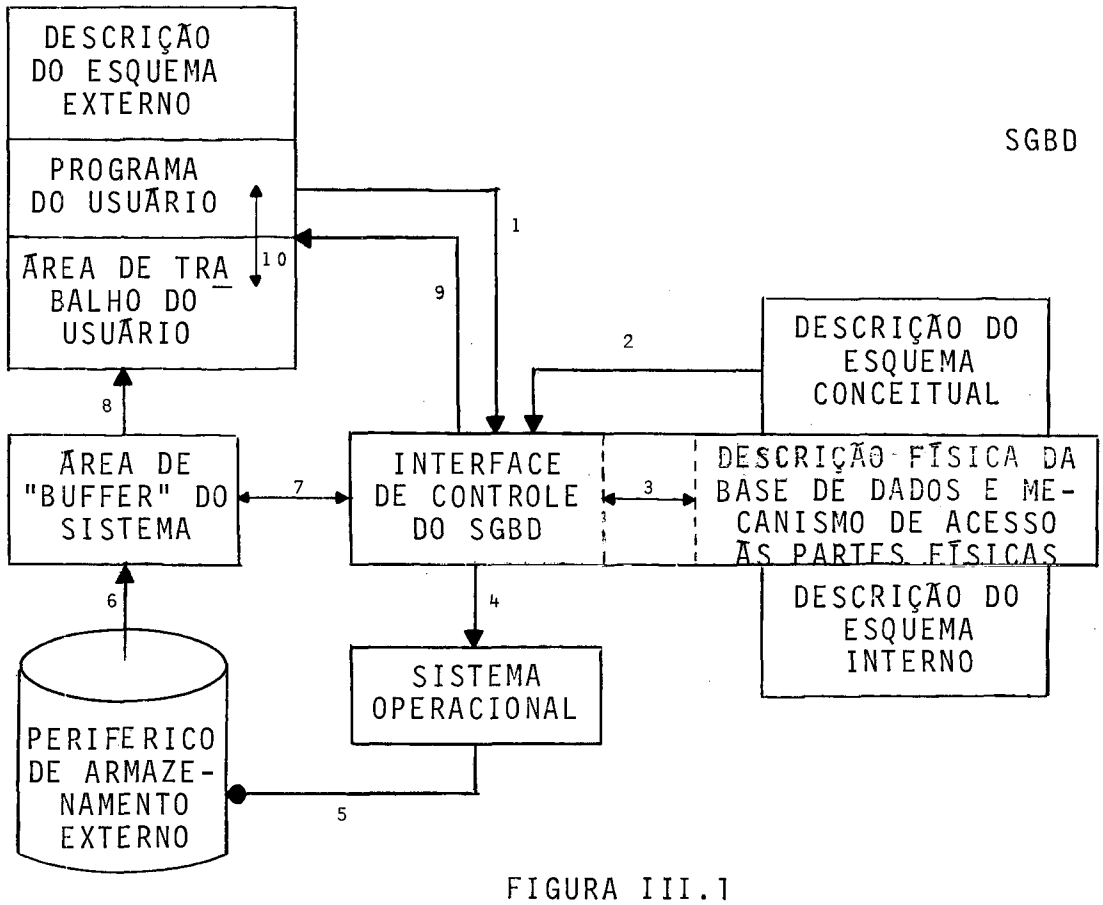


FIGURA III.1

1. A declaração de I/O ou consulta é detectada e analisada sintática e semanticamente pelo SGBD.

2. O SGBD verifica se o esquema associado à declaração de I/O está definido em seu diretório e se pode ser acessado pelo usuário.

3. O SGBD verifica, em relação aos registros a serem acessados pela declaração, sua posição física dentro do periférico de armazenamento determinando os blocos que devem ser acessados.

4. O SGBD utiliza os comandos de I/O do sistema operacional da máquina para acessar fisicamente os registros ou blocos que contenham os registros desejados.

5. O sistema operacional ativa o periférico de armazenamento externo e acessa os registros requisitados.

6. O sistema operacional transfere os dados da memória secundária para a área de "buffer" do sistema, onde somente o SGBD pode acessar.

7. Os registros solicitados pela declaração de I/O do usuário são selecionados pelo SGBD e, depois de formatados, liberados para o usuário.

8. O SGBD transfere os registros selecionados, da área de "buffer" do sistema para a área de trabalho do usuário.

9. O SGBD informa ao usuário o sucesso ou insucesso da operação.

10. O programa de aplicação passa a manipular os dados da área de trabalho.

III.3 - Modelagem

Um modelo de banco de dados é uma maneira de estruturar logicamente as informações contidas na base de dados.

Estes modelos dividem-se em três tipos - hierárquico, rede e relacional - e distinguem-se, principalmente, pelos tipos de elos utilizados.

Estes são ligações entre registros, não necessariamente de arquivos distintos, divididas em duas modalidades, a saber:

- ELOS EXPLÍCITOS: quando a ligação é feita através de uma informação adicional como, por exemplo, o endereço físico do registro.

- ELOS IMPLÍCITOS: quando a ligação é feita através de um ou mais campos do próprio registro.

Nos elos implícitos as ligações estão contidas nos próprios campos que constituem o registro, enquanto nos elos explícitos há necessidade de informações adicionais.

Os três principais modelos de banco de dados são:

- Hierárquico;
- Rede;
- Relacional.

Para facilitar a percepção das diferentes representações dos dados, definiremos uma base de dados padrão, que servirá como exemplo nas definições dos três modelos.

Seja a base de dados, contendo informações sobre os fornecedores (número, nome, cidade), as peças fornecidas contendo (número, nome, cor, peso) e os projetos a serem abastecidos pelos fornecedores com as citadas peças, contendo (número, nome, cidade).

Os fornecedores, peças e projetos serão apresentados em forma de tabela, contendo, respectivamente, os "label's" F(F#, FNOME, CID), P(P#, PNOME, COR, PESO) e J(J#, JNOME, CID), e tomarão as formas exigidas por cada modelo conforme sejam mencionados.

F

F#	FNOME	CID
F1	JOAO	RIO
F2	PEDRO	SP
F3	ALANO	FOR
F4	CARLOS	RIO
F5	JOSE	FOR

P

P#	PNOME	COR	PESO
P1	PARAFUSO	PRETA	18
P2	PORCA	PRETA	18
P3	PORCA	AZUL	20
P4	FLANDE	VERDE	40
P5	FERRO	BRANCA	100
P6	FIO	VERDE	25

J

J#	JNOME	CID
J1	PUNCH	RIO
J2	READER	FOR
J3	CONSOLE	SP
J4	CONSOLE	NAT
J5	TAPE	REC
J6	TERMINAL	POR

FIGURA III.2 - Exemplo de Base de Dados

III.3.1-Modelo Hierárquico

O usuário de um banco de dados hierárquico "vê" a base de dados como uma coleção - ou floresta - de árvores disjuntas, contendo em seus nodos as ocorrências de registros.

Cada árvore da base de dados é formada por um registro raiz e seus dependentes e parte dos seus elos são explícitos.

Exceto no nodo raiz, toda ocorrência de um registro está obrigatoriamente subordinada a um nodo ascendente. A um nodo ascendente, podem estar ligados vários nodos descendentes. Por outro lado, cada nodo descendente só pode estar ligado a apenas um ascendente direto. Logo, o relacionamento em um modelo hierárquico é de 1:n.

Para exemplificarmos o modelo hierárquico, usaremos o exemplo dos fornecedores e peças fornecidas da fig.(III.2).

Neste exemplo será tomada a decisão de que os for

necedores serão ascendentes das peças fornecidas. Esta decisão é arbitrária, visto que nada nos impede de subordinar os fornecedores às peças, uma vez que, por natureza, trata-se de um relacionamento n:m.

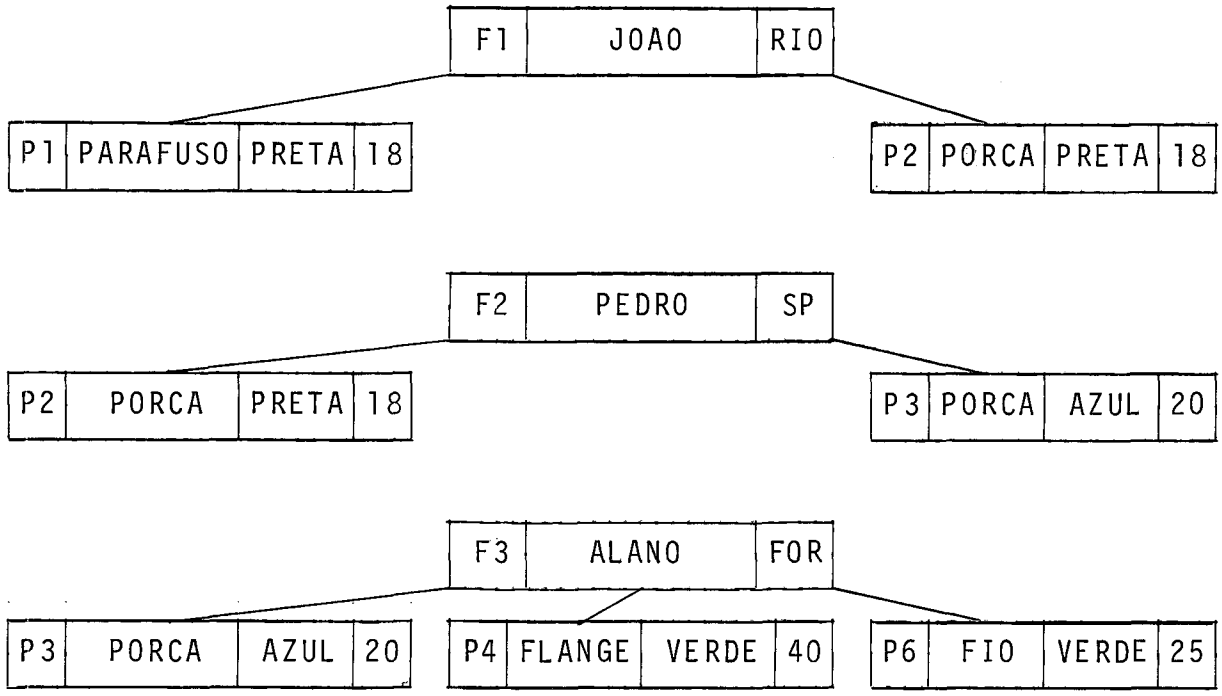


FIGURA III.3 - Estrutura Hierárquica

Note-se que as peças P2 e P3 estão repetidas em dois segmentos distintos. Este é um dos casos adequados para mostrar que o exemplo dos fornecedores não adapta-se perfeitamente ao modelo hierárquico tendo em vista que uma peça pode ser fornecida por mais de um fornecedor.

Neste relacionamento - fornecedores e peças - a pergunta "quais as peças fornecidas pelo fornecedor Fx?" é facilmente atendida; basta localizar o fornecedor Fx e obter todos os seus descendentes. Já a pergunta "quais os fornecedores da peça Py?" exige uma busca em todos os fornecedores para verificar se ele possue como descendente a peça Py.

Note também que, pode existir na base de dados, informações sobre FORNECEDORES sem que estes estejam fornecendo peças; não é possível no entanto, manter informações sobre peças sem seus respectivos fornecedores.

III.3.1.1 -Linguagem de Dados

Os comandos apresentados a seguir têm por finalidade manipular uma base de dados organizada segundo o modelo hierárquico, usando uma ordenação típica, denominada pré-ordem, conforme o algoritmo seguinte, com R sendo a raiz:

P1: Acesse o registro R, se ainda não tiver sido acessado. (Na primeira vez, será acessado o registro raiz).

P2: Se existir descendente imediato de R, faça:

R igual ao primeiro deles ainda não acessado e volte para P1.

P3: Se R for igual à raiz, PARE.

P4: Faça R igual ao seu ascendente (pai).

P5: Volte para P1.

Para o caso da figura (III.3), com o registro (F1, JOÃO, RIO) sendo raiz, a ordenação, segundo o método acima, ficaria da seguinte maneira:

a) acesse o registro R

R = | F1 | JOÃO | RIO |

b) faça R igual ao primeiro descendente imediato e acesse R

R = | P1 | PARAFUSO | PRETA | 18 |

c) faça R igual ao próximo descendente imediato e acesse R

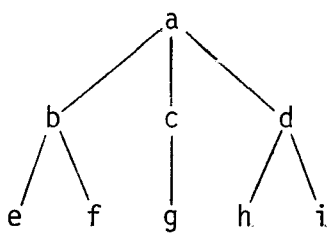
R = | P2 | PORCA | PRETA | 18 |

d) se R igual à raiz, PARE.

A ordenação final seria:

(F1,...)(P1,...)(P2,...)

A figura (III.4) mostra um segundo exemplo do procedimento em pré-ordem.



Estrutura Hierárquica

a - b - e - f - c - g - d - h - i

Ordem de Acesso

FIGURA III.4 - Acesso em pré-odem

a) GET UNIQUE

Usado para efetivação de acessos diretos a segmentos do banco de dados ou para definir a posição inicial de um processamento seqüencial.

b) GET NEXT

Usado para recuperar o próximo segmento corrente. Pode ser especificado o tipo de segmento a considerar mas, caso seja omitido, o acesso obedecerá à ordenação pré-ordem.

c) GET NEXT WITHIN PARENT

Usado para recuperar o próximo segmento corrente dentro da mesma família. A diferença deste comando para o GET NEXT é que ele encerra os acessos após obter o último segmento na mesma família, enquanto o GET NEXT continua até o último registro da base de dados.

d) HOLD

A cláusula HOLD pode ser combinada com qualquer um dos comandos acima. Sua função é manter o registro para, em seguida, efetuar uma alteração através de um dos comandos REPLACE ou DELETE.

e) INSERT

Tem como função inserir novos registros na ba-

se de dados.

Para inserir um novo registro deve-se primeiro acessar o seu ascendente e depois efetuar a inserção do novo registro, como descendente, através do comando INSERT.

f) REPLACE

Tem como função alterar o conteúdo de um registro na base de dados ("update"). O registro a ser alterado deve primeiro ter sido acessado por um comando, usando a cláusula HOLD.

g) DELETE

Usado para retirar da base de dados um registro e todos os seus descendentes. O registro a ser retirado deve ser primeiro acessado por um comando, usando a cláusula HOLD, e o comando DELETE trata de removê-lo juntamente com seus descendentes.

Vejamos, a seguir, alguns exemplos usando os comandos mencionados e a base de dados da figura (III.3).

EX1: GET UNIQUE FORNECEDORES WHERE (F# = F1)

O registro acessado é o (F1, JOÃO, RIO)

GET UNIQUE PEÇAS WHERE (F# = RESEARCH)
AND (P# = P3)

O registro acessado é o (P3, PORCA, AZUL, 20)

EX2: GET NEXT FORNECEDORES

Os registros acessados são (F1, JOÃO, RIO;
F2, PEDRO, SP;
F3, ALANO, FOR)

EX3: GET NEXT WITHIN PARENT PEÇAS

Se o segmento corrente fosse (F1, JOÃO, RIO),

Seriam acessados os registros
(P1, PARAFUSO, PRETA, 18;
P2, PORCA, PRETA, 18)

EX4: GET HOLD UNIQUE FORNECEDORES WHERE (F# = F3)
REPLACE PAULO TO FNOME

O registro (F3, ALANO, FOR) seria acessado e mantido para ser feita a alteração, passando a ser (F3, PAULO, FOR).

III.3.2 - Modelo em Rede

No modelo de rede todos os elos podem ser explícitos, porque não há restrição a um só tipo de relacionamento como no modelo hierárquico. Para cada tipo de relacionamento existem elos com nomes ou rótulos distintos. Os elos podem ser n:m.

Uma forma especial de modelo de rede foi apresentada pelo Data Base Task Group (DBTG) da Conference on Data Systems and Languages (CODASYL). O modelo COSASYL é particularmente importante por ser uma tentativa de padronização em bancos de dados segundo o modelo em rede. Por esta razão, definiremos

neste trabalho o modelo em rede conforme os padrões da CODASYL. Embora seja sabido que o padrão único não foi aceito, várias idéias nele contidas foram adotadas em sistemas de gerência de banco de dados segundo o modelo em rede, FURTADO[13].

No modelo CODASYL um relacionamento n:m não pode ser representado diretamente, devendo ser traduzido em termos de dois ou mais relacionamentos 1:n.

Os elos que representam os relacionamentos são denominados conjuntos - CODASYL. Os conjuntos - CODASYL são responsáveis pelas ligações existentes entre dois arquivos, chamados de DONO e MEMBRO, sendo que:

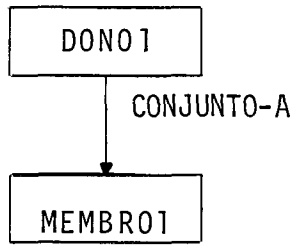
a) DONOS e MEMBROS devem ser registros de tipos diferentes;

b) dada a restrição a relacionamentos 1:n, a um DONO podem corresponder vários MEMBROS, mas cada MEMBRO só está relacionado a um DONO;

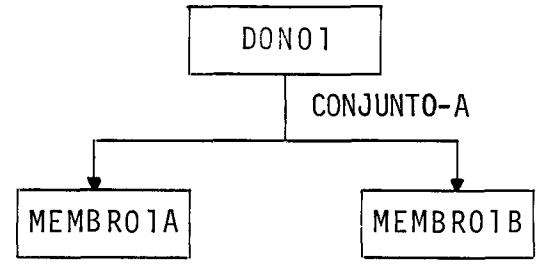
c) a propriedade anterior parece restringir a capacidade dos conjuntos - CODASYL para representar hierarquias, mas isso não ocorre porque um mesmo registro pode aparecer em mais de um conjunto - CODASYL - em mais de um DONO - sem que haja sua reprodução física.

As figuras (III.5.(a, b, c, d, e)) procuram mostrar as situações que podem ocorrer com a estrutura de um modelo em

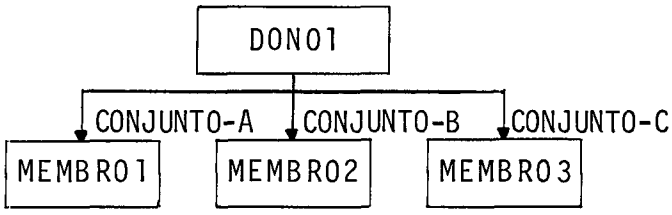
rede, segundo a CODASYL, CARDENAS[2].



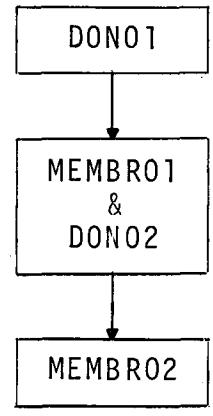
(a) Um tipo de registro DONO e um tipo de registro MEMBRO



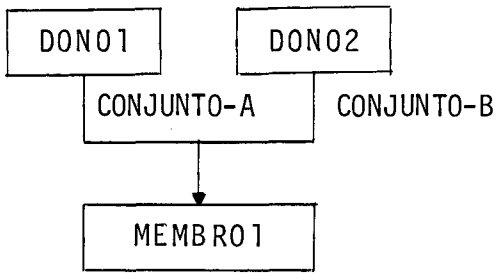
(b) Dois tipos de registros como MEMBROS de um único DONO através de um único conjunto.



(c) Um único tipo de registro DONO de três membros através de três CONJUNTOS-CODASYL diferentes.



(d) Um tipo de registro como MEMBRO de um CONJUNTO-CODASYL e DONO através de outro conjunto



(e) Um tipo de registro como MEMBRO de dois DONOS através de conjuntos distintos.

FIGURA III.5 - Situações possíveis entre registros DONOS e MEMBROS no modelo MODASYL

Uma ocorrência de um conjunto - CODASYL é formada por um dono e todos os membros a ele associados.

Para exemplificarmos o modelo em rede, usaremos o exemplo dos FORNECEDORES e PEÇAS fornecidas, conforme a figura (III.6).

Note-se que as peças P2 e P3, duplicadas na representação do modelo hierárquico, não foram duplamente representadas neste modelo. Embora exista o relacionamento envolvendo estas duas peças em mais de um conjunto - CODASYL (FORNECE), elas estão armazenadas, fisicamente, em um só lugar da base de dados.

III.3.2.1 - Linguagem de Definição de Dados

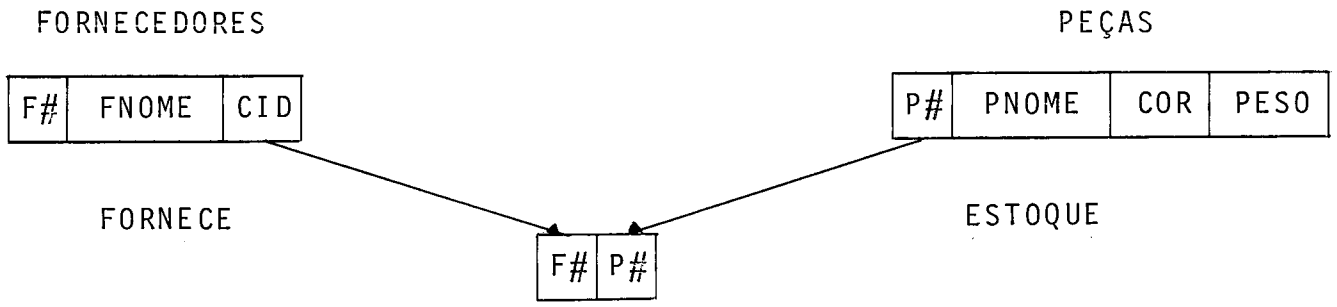
O sistema CODASYL inclue quatro tipos de declarações para descrever o esquema DDL (Data Definition Language):

1) declaração de nome do esquema - por exemplo, SCHEMA NAME IS FORNECEDORES - PECAS;

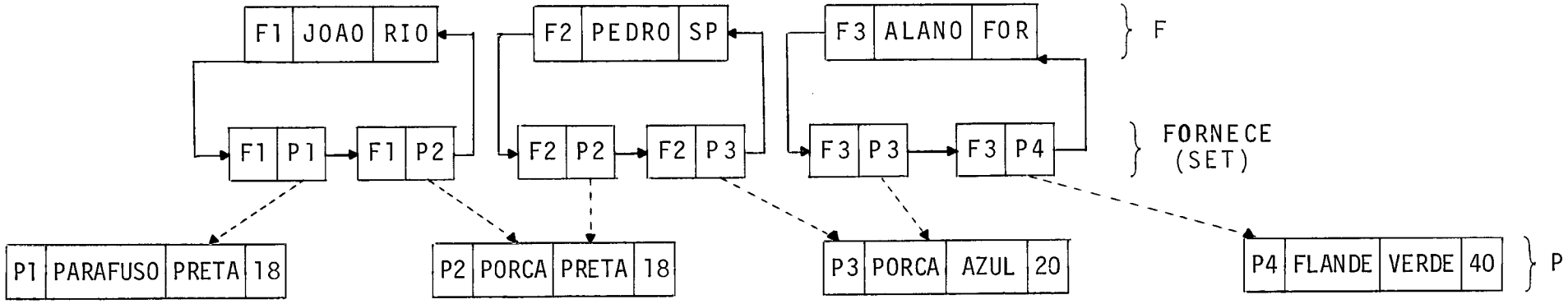
2) uma ou mais declarações de tipos de registros, definindo os itens de dados para cada tipo de registro (nome, tipo e número de caracteres);

3) uma ou mais declarações de conjuntos - CODASYL e seus relacionamentos envolvendo os tipos de registros definidos;

4) uma ou mais declarações de áreas, definindo as



(a) Esquema



(b) Parte da Base de Dados

FIGURA III.6 - Estrutura em Rede

áreas nas quais os registros serão armazenados.

O exemplo a seguir, define um esquema para a base de dados da figura (III.6) - para a linguagem COBOL - e, logo em seguida, explicaremos cada uma das declarações:

```
1.  SCHEMA NAME IS FORNECEDORES - PECAS.

2.  AREA NAME IS FORN-FC-AREA.

3.  RECOR NAME IS FORNECEDORES;

4.  PRIVACY LOCK FOR GET FIND IS GF;

5.  PRIVACY LOCK FOR MODIFY, INSERT, DELETE, REMOVE,
    STORAGE IS MIDRS;

6.  LOCATION MODE IS CALC HASH-PROC-1 USING F# IN
    FORNECEDORES;

7.  DUPLICATES ARE NOT ALLOWED;

8.  WITHIN FORN-FC-AREA.

9.  02 F ;        PICTURE X(2).
    02 FNOME;     PICTURE A(10).
    02 CID;       PICTURE A(10).

10. RECORD NAME IS PECAS;
```

```
11. LOCATION MODE IS CALC HASH-PROC-2 USING P# IN
    PECAS;

12. DUPLICATES ARE NOT ALLOWED;

13. WITHIN FORN-FC-AREA.
    02P#;        PICTURE X(2).
    02 PNOME;    PICTURE A(10).
    02 COR;      PICTURE A(8).
    02 PESO;     PICTURE X(2).

14. SET NAME IS FORNECE;

15. ORDER IS SORTED;

16. MODE IS CHAIN;

17. OWNER IS FORNECEDORES.

18. MEMBER IS PECAS MANDATORY AUTOMATIC;

19. ASCENDING KEY P#;

20. SET OCCURRENCE SELECTION IS THRU LOCATION MODE
    OF OWNER.
```

A declaração 1 define o nome do esquema.

A declaração 2 indica o nome da área na qual os registros da base de dados irão residir - a área FORN-FC-AREA con-

terá todas as ocorrências de FORNECEDORES e PECAS (FORNECEDORES e PECAS poderiam estar em áreas distintas).

Em 3 e 10 são declarados tipos de registros de nomes FORNECEDORES e PECAS, respectivamente

Em 4 e 5 são especificadas as chaves de controle de acesso para as operações citadas nas declarações.

Em 6 e 11 são definidos procedimentos para acessar FORNECEDORES e PECAS, respectivamente. (HASH-PROC-1 e HASH-PROC-2 são procedimentos definidos pelo administrador da base de dados).

Em 7, 12 e 20 declaramos que duas ocorrências não podem possuir o mesmo valor do campo chave.

Em 8 declaramos que as ocorrências dos FORNECEDORES estão fisicamente armazenadas na área FORN-FC-AREA.

Em 9 estão declarados os campos de FORNECEDORES.

Em 14 é declarado um tipo de conjunto de nome FORNECE.

De 15 a 21 são feitas declarações sobre o tipo de conjunto FORNECE.

III.3.2.2 -Linguagem de Manipulação de Dados

Os comandos da linguagem de manipulação de dados

que serão apresentados, utilizam-se, na sua maioria, dos indicadores de estado corrente.

Um indicador do estado corrente é uma variável do SGBD, a qual, em relação a cada área, tipo de registro (RECORD type) e tipo de conjunto (set type ou conjunto CODASYL), identifica o valor da chave do registro "mais" recentemente acessado na base de dados, pelo programa de aplicação. Os indicadores do estado corrente são:

1. "Current of area A": referencia a ocorrência do registro mais recentemente acessado dentro da área A. Uma área é uma porção do espaço de armazenagem na qual registros estão fisicamente armazenados independente dos seus relacionamentos nos conjuntos - "sets". O espaço onde a base de dados está armazenada é dividido entre uma ou mais áreas.

2. "Current of record type R" - referencia, entre os registros de R, o "mais" recentemente acessado.

3. "Current of set type S" - referencia o acesso "mais" recente do conjunto S. No caso da base de dados da figura (II.6), referencia o último FC acessado.

4. "Current of run unit" - referencia a ocorrência de registro mais recentemente acessada, in

dependentemente do seu envolvimento com áreas ou conjuntos ("sets").

O "current area" referencia o último registro acessado dentro de uma área - que pode ser constituída por mais de um tipo de registro. O "current record type", referencia o último registro acessado pertencente a um determinado tipo de registro. O "current set" referencia o último par (dono, membros) de um determinado tipo de conjunto ("set type") e o "current run unit" referencia o último registro acessado na base de dados.

Os comandos apresentados a seguir são orientados para programas de aplicação com o uso da linguagem COBOL. Na sua maioria, são definidos através dos indicadores de estado corrente.

(a) OPEN

Tem como função abrir todas as áreas que serão usadas pelo "RUN UNIT".

(b) CLOSE

Permite fechar ou liberar áreas.

(c) ORDER

Permite especificar, para um dado conjunto, uma nova ordem dos seus registros-membros a ser usada apenas durante a execução do programa.

(d) MOVE

Permite obter um valor explícito (uma chave da base de dados) para qualquer um dos indicadores de estado corrente (RUN UNIT, RECORD SET ou AREA).

(e) GET

Recupera ou dá saída para o programa do registro-objeto indicado pelo RUN UNIT CORRENTE.

(f) FIND

Permite localizar, sem recuperar, uma ocorrência de registro, fazendo com que os indicadores correntes da área, do tipo de registro e do tipo de conjunto - "set type" - passem a ter o mesmo valor do indicador "run unit". Com isto, qualquer operação envolvendo mudanças na base de dados será efetuada sobre o último registro acessado, indicado pelo "run unit".

(g) STORE

Permite armazenar uma nova ocorrência de registro na base de dados.

(h) DELETE

Permite remover o indicador corrente do RUN UNIT em todos os conjuntos nos quais ele é um

membro e todos os membros de um conjunto onde ele é dono.

(i) INSERT

Permite inserir o indicador corrente do RUN UNIT como membro de um tipo de conjunto especificado.

(j) MODIFY

Permite modificar todos ou alguns itens específicos do RUN UNIT corrente, com os valores de uma UWA.

III.3.3 - Modelo Relacional

O usuário de um banco de dados estruturado conforme o modelo relacional interpreta a base de dados como sendo um conjunto de tabelas chamadas de RELAÇÕES.

Os elementos de uma relação são tuplas que, por sua vez, são constituídas de valores de domínios.

Uma notação matemática, para definir uma relação, seria: se D1, D2, ..., Dn são domínios, não necessariamente distintos, uma relação é um subconjunto do produto cartesiano dos domínios, ou seja, R $\subset$ D1 X D2 X D3 X ... X Dn.

Como uma relação pode ser representada como tabela, existe a seguinte correspondência de terminologia:

- cada domínio corresponde a uma coluna da tabela;

- cada tupla, a uma linha.

Conforme definições do modelo relacional propostas por CODD[4], uma relação é formada por um conjunto de tuplas. Então, a ordem destas tuplas, não altera a relação e todas são distintas.

O grau de uma relação é dado pela sua quantidade de domínios. Por exemplo, uma relação de grau quatro (4) quer dizer que nela existem quatro domínios.

A relação "F" da figura (III.7) tem as seguintes características:

- seus domínios são F#, FNOME e CID;

- é de grau três;

- o grupo (F1, JOÃO, RIO) forma uma tupla;

- é de cardinalidade igual a cinco (5) pois contém cinco tuplas.

Os relacionamentos no modelo relacional podem ser dentro da mesma relação ou entre relações distintas através de comparações entre seus domínios. Neste modelo todos os elos são implícitos.

Para manipulação do banco de dados são oferecidos

dois formalismos: a álgebra relacional e o cálculo relacional neste trabalho será comentado a álgebra relacional, decisão puramente arbitrária, usando como exemplo as relações da figura (III.7).

FORNECEDORES

F

F#	FNOME	CID
F1	JOAO	RIO
F2	PEDRO	SP
F3	ALANO	FOR
F4	CARLOS	RIO
F5	JOSE	FOR

PEÇAS

P

P#	PNOME	COR	PESO
P1	PARAFUSO	PRETA	18
P2	PORCA	PRETA	18
P3	PORCA	AZUL	20
P4	FLANDE	VERDE	40
P5	FERRO	BRANCA	100
P6	FIO	VERDE	25

PROJETOS

J

J#	JNOME	CID
J1	PUNCH	RIO
J2	READER	FOR
J3	CONSOLE	SP
J4	CONSOLE	NAT
J5	TAPE	REC
J6	TERMINAL	FOR

FIGURA III.7 - Estrutura Relacional

Para a base de dados da figura (III.7), será criada uma relação, denominada fornecimentos, que conterá todos os elos entre as três relações existentes. Vale notar que o modelo permite a existência de fornecedores, peças ou projetos que não estejam relacionados entre si.

FPJ

F#	P#	J#	QTDE
F1	P1	J1	200
F1	P2	J4	200
F2	P2	J1	100
F2	P3	J2	100
F2	P3	J3	100
F2	P3	J4	100
F3	P3	J1	400
F3	P4	J2	400
F4	P5	J1	300
F4	P5	J2	300

FIGURA III.8 - Relação dos Fornecimentos

Uma relação é dita união-compatível se seus domínios são do mesmo número e tipo. Esta denominação será usada para definir a validade de algumas operações da álgebra relacional que veremos a seguir.

III.3.3.1 - Álgebra Relacional

a) UNIÃO

É uma operação efetuada entre duas relações união-compatíveis e tem como resultado uma outra relação contendo todas as tuplas, sem repetição, das duas relações;

Ex: S ← F ∪ J

b) INTERSEÇÃO

É uma operação efetuada entre duas relações união-compatíveis e tem como resultado uma outra relação contendo as tuplas existentes nas relações F e J;

Ex: S ← F ∩ J

c) DIFERENÇA

É uma operação efetuada entre duas relações união-compatíveis e tem como resultado uma relação que contém as tuplas existentes na primeira relação e não existentes na segunda;

Ex: S ← F - J

d) PRODUTO CARTESIANO

É uma operação efetuada entre duas relações e tem como resultado uma relação que contém a concatenação de todas as tuplas da primeira com a segunda relação;

Ex: S ← F • J

e) PROJEÇÃO

É uma operação efetuada em uma relação e tem como resultado uma relação que contém apenas alguns domínios, indicados na operação, da relação existente;

Ex: S ← F(F#, FNOME)

f) RESTRIÇÃO

É uma operação efetuada em uma relação e tem como resultado uma relação que contém apenas as tuplas da relação que satisfazem uma determinada condição;

Ex: S ← F(F# > F2)

g) JUNÇÃO

É uma operação efetuada entre duas relações - F e FPJ, por exemplo - e tem como resultado uma outra que conterá a concatenação das tuplas das duas relações, conforme uma condição entre dois

domínios destas relações (F e FPJ);

Ex: S ← F(F# = F#)FPJ

h) DIVISÃO

Seja a operação de divisão:

S ← SPJ(J# ÷ J#)J

para saber quais são as tuplas que passaram a compor a relação "S", procede-se da seguinte maneira:

1. Grupa-se as tuplas de SPJ que têm o mesmo valor para seqüência de domínios F#, P# e QTDE.

2. Toma-se a projeção de J(J#).

3. Verifica-se quais os grupos de tuplas de SPJ que contêm todos os valores da projeção J(J#) no domínio J#.

4. Copia-se para S os valores de F#, P# e QTDE que satisfazem a condição.

Veja como são efetuados os procedimentos acima para a expressão utilizada:

Procedimento 1

Procedimento 2

FPJ

	F#	P#	J#	QTDE
GRUPO 1	F1	P1	J1	200
GRUPO 2	F1	P2	J4	200
GRUPO 3	F2	P2	J1	100
GRUPO 4	F2	P3	J2	100
	F2	P3	J3	100
	F2	P3	J4	100
GRUPO 5	F3	P3	J1	400
GRUPO 6	F3	P4	J2	400
GRUPO 7	F4	P5	J1	300
	F4	P5	J2	300

J(J#)

J#
J1
J2
J3
J4
J5
J6

FIGURA III.9 - Formação dos Grupos

Procedimento 3

Não existe nenhum grupo que satisfaça a condição.

Procedimento 4

A relação S será vazia.

Vejamos agora alguns exemplos práticos.

Ex1: Obter o nome e a cidade dos fornecedores do projeto J4.

PASSO1: obteremos, da tabela de fornecimentos(FPJ), todas as tuplas onde J# = J4.

S1 ← FPJ(J# = J4)

S1

F#	P#	QTDE
F1	P2	200
F2	P3	100

FIGURA III.10 - Resultado da Restrição

PASSO2: efetuar uma junção entre as relações F e S1.

S2 ← F(F# = F#)S1

S2

F#	FNOME	CID	F#	P#	J#	QTDE
F1	JOÃO	RIO	F1	P2	J4	200
F2	PEDRO	SP	F2	P3	J4	100

FIGURA III.11 - Resultado da Junção

PASSO3: projetar em S2 os domínios FNOME e CID.

S3 ← S2(FNOME, CID)

S3

FNOME	CID
JOÃO	RIO
PEDRO	SP

FIGURA III.12 - Relação que responde a pergunta

Nos próximos exemplos, mostraremos apenas a se-

qüência de comandos.

Ex2: obter o número, o nome e a cidade dos projetos em andamento.

S4 FPJ(J#) : projeção

S5 J(J# = J#)S4 : junção

S6 S4(J#, JNOME, CID) : projeção

Ex3: obter o código das peças de cor preta

S7 ← P(COR = PRETA) : restrição

S8 ← S7(P#) : projeção

Ex4: obter os fornecedores da cidade do RIO

S9 ← F(CID = RIO) : restrição

III.4 - Conclusões Quanto ao Uso dos Modelos

Os sistemas de gerência de banco de dados disponíveis comercialmente estão, em geral, comprometidos de alguma forma com um dos três modelos descritos anteriormente: hierárquico, rede e relacional.

Na realidade, um SGBD deveria permitir que o usuário utilizasse o modelo que, em sua opinião, fosse mais adequado para cada base de dados ou que, simplesmente, fosse da sua preferência.

Mais importante ainda seria, dispor-se de interfaces que permitissem a diferentes usuários de uma mesma base de dados manipulá-la conforme o modelo que escolhesse - veja figura (III.13). DATE[7] apresenta uma linguagem denominada "Unified Database" (UDL), a qual pemite que o usuário defina os três tipos de estruturas - hierárquica, rede e relacional - para a mesma base de dados. Não é uma linguagem autocontida; é, na realidade, uma extensão para as linguagens de programação existentes (COBOL, PL/I, ...) e pode ser incorporada, com algumas modificações apropriadas, em uma variedade de linguagens hospedeiras.

Todavia, dentro do contexto de processamento de dados distribuído e heterogêneo, isto é, onde existe variedade de "hardware" e "software" entre os nós do ambiente distribuído, o problema de modelagem de base de dados torna-se de complexidade maior. É suposto que, em cada nó do ambiente distribuído, possa existir uma base de dados - denominada local ou remota - observando a modelos gerenciados por SGBD distintos. Considerando-se que determinadas aplicações necessitam obter dados envolvendo duas ou mais bases de dados do ambiente distribuído, isto é, ter acesso a informações armazenadas e gerenciadas em diferentes nós do sistema distribuído.

Para atender a estes casos, duas alternativas devem ser colocadas: 1) exigir que o usuário tenha conhecimento de como utilizar cada modelo de banco de dados existente no ambiente distribuído; 2) criar interfaces - em nível de "hardware" e/ou "software" - que permitam, através de procedimentos padronizados e de fácil uso, a obtenção de dados do ambiente distribuído.

A primeira alternativa, embora de fácil implementação, tornou-se pouco aceitável, tendo em vista que exige do usuário saber manipular cada SGBD existente no ambiente distribuído.

Neste trabalho será proposta uma solução, baseada na segunda alternativa, na qual o usuário poderá acessar dados do ambiente distribuído de forma ainda limitada, porém simples e de fácil uso. É proposto o desenvolvimento de um conjunto de facilidades de "software" que oferecem uma interface, entre os usuários e o sistema distribuído, assumindo que seja de forma transparente para o usuário a execução de todas as tarefas necessárias à compatibilização entre os diversos modelos e SGBD's em uso pelos nós do sistema distribuído.

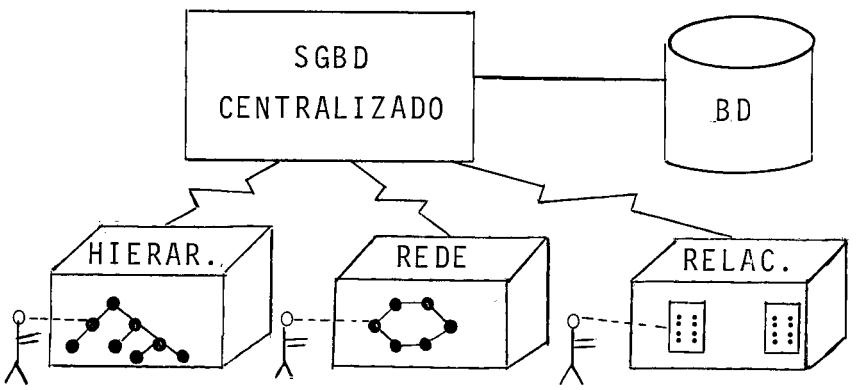


FIGURA III.13 - SGBD centralizado, ideal.

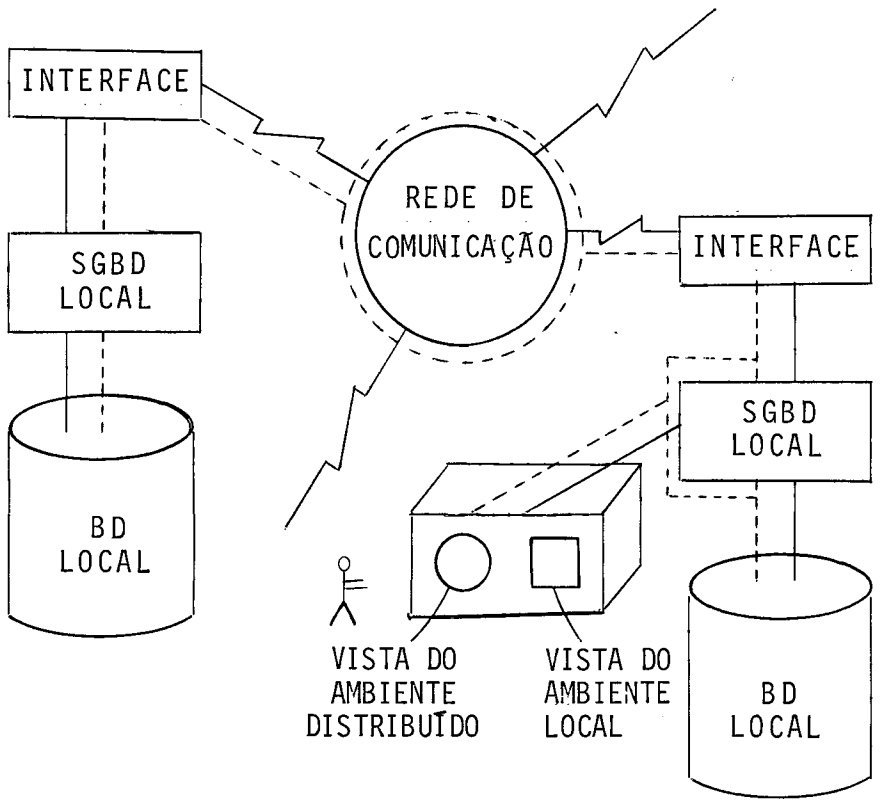


FIGURA III.14 - Visão do usuário em relação ao que se propõe neste trabalho.

CAPÍTULO IV - PROPOSTA PARA PROCESSAMENTO DE DADOS DISTRIBUÍDO

IV. PROPOSTA PARA PROCESSAMENTO DE DADOS DISTRIBUÍDOS

IV.1 - O Problema de Cooperação entre Diferentes SGBDs

A proposta que se discute neste trabalho está ligada ao problema de portabilidade entre modelos em um ambiente de processamento de dados distribuído, ou seja, uma alternativa para transportar bases de dados, ou parte delas, pelos diferentes nós de um sistema distribuído heterogêneo.

A proposta aqui discutida não vem oferecer, aos usuários da rede que suporta o banco de dados distribuído, uma linguagem tão poderosa em manipulação, edição e alteração de conjunto de dados como as linguagens de manipulação de banco de dados normalmente oferecem. O que se propõe na realidade, é oferecer comandos simples que permitam uma comunicação com as bases de dados existentes em um sistema distribuído, sem a preocupação com sua localização física, estrutura lógica e física ou, mais precisamente, com sua modelagem.

Para tratar do problema de cooperação entre vários SGBDs em um ambiente de processamento de dados distribuído, consideramos, em primeiro lugar, a existência de várias bases de dados locais distintas B_1, B_2, ... B_n, as quais são usadas independentemente, gerenciadas por SGBDs heterogêneos. Consideramos, também, que uma determinada classe de usuários necessita de uma vista global B das bases de dados B_1, B_2, ..., B_n para construir uma determinada classe de aplicações.

IV.2 - Tipos de Nós em um Sistema Distribuído

O conjunto de dados formando um banco de dados distribuído está armazenado nos diversos nós da rede de computadores que suporta o sistema distribuído. Stefano Spaccapietra, DRAFFAN[10], classifica estes nós em três tipos, a saber:

IV.2.1 - Nós de Armazenagem

São os nós que têm como função armazenar os dados que formam o banco de dados distribuído. Este tipo de nós apenas fornece informações à rede sem executar, contudo, nenhum pedido de informação envolvendo o ambiente distribuído.

IV.2.2 - Nós de Acesso

São os nós que têm como função única interagir com o ambiente do banco de dados distribuído para obter informações. Este tipo de nós não armazena dados do banco de dados distribuído.

IV.2.3 - Nós de Troca

São os nós mais completos. Eles efetuam as funções dos nós de acesso e armazenagem no ambiente do processamento de dados distribuído.

Estes três tipos de nós são ilustrados na figura (IV.1).

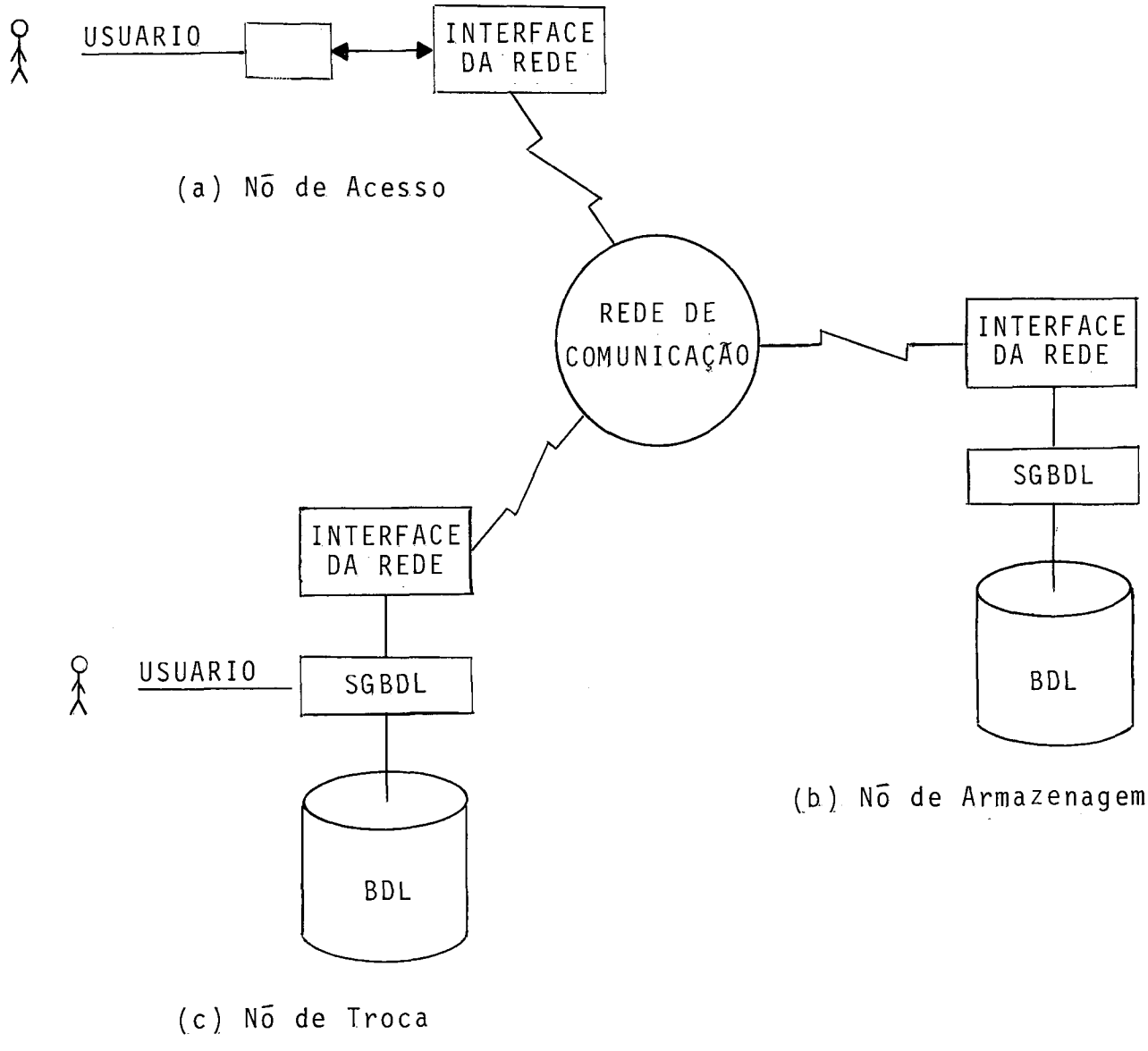


FIGURA IV.1 - Tipos de Nós em um Ambiente Distribuído

IV.3 - Sistema de Comunicação Proposto

Em um ambiente distribuído o usuário pode comunicar-se com a base de dados local, diretamente através SGBD - LOCAL ou com as base de dados distribuídas entre os vários nōs de armazenagem da rede através do sistema a ser proposto - deste ponto em diante chamado Sistema Intermediário (SI).

A interação do usuário com o ambiente do banco de dados distribuído é formulada com base em uma linguagem de manipu

lação de dados (LMD), que é interceptada pelo sistema intermediário, o qual determina quais os nós que devem ser acessados para satisfazer o pedido de informação do usuário. Isto introduz a necessidade de um diretório de dados da rede, o qual deve ser acessado pelo sistema intermediário com este propósito. O diretório de dados da rede deve ser capaz de indicar os nós de armazenagem, baseado nas várias unidades de dados que compõem o ambiente do banco de dados distribuído. Por exemplo: dado um nome-de-relação ou conjunto-de-dados o diretório determina o nó que a contém, o nome do arquivo a ser acessado e a localização física no arquivo da informação desejada. Ele não oferece informações sobre a localização física dos nós ou os caminhos entre eles. Esta informação faz parte das facilidades de comunicação da rede DRAFFAN[10].

Tendo acessado o diretório de dados, o sistema intermediário coordena o processamento, obtendo os dados solicitados, e responde ao usuário em forma de geração de arquivos auxiliares em uma área de saída.

Cada base de dados, no ambiente distribuído, será recuperada observando um procedimento exclusivo que interage com o sistema de gerência desta base de dados específica para obter os dados que venham a ser solicitados pelo sistema intermediário. Este procedimento pode ser programado utilizando-se os próprios recuros oferecidos por cada SGBD - LOCAL e tem por finalidade recuperar os registros de um determinado arquivo da base de dados em questão e transmití-lo para o S.I., observando um formato padronizado.

Qualquer restrição por parte do usuário quanto aos

registros a serem recuperados será analisada pelo sistema intermediário que, ao receber os registros, seleciona-os conforme a restrição e grava-os em um arquivo auxiliar.

IV.3.1 - O Sistema Intermediário em um Banco de Dados Distribuído

Chamamos de base de dados local (BDL) o subconjunto do banco de dados distribuído (BDD) armazenado em um nó de armazenagem; logo, o BDD é a união de todos os BDLs. Neste ponto vale alertar que a base de dados local pode ser formada por uma base de dados independente ou ser parte de uma grande base de dados, distribuída entre os nós de armazenagem da rede - uma base de dados particionada entre dois ou mais nós de armazenagem do ambiente distribuído.

Cada BDL é gerenciada por um sistema de gerência de banco de dados local (SGBDL), de padrões independentes, que é complementado por uma rotina de comunicação local (RCL), a qual serve de mediadora entre o SGBDL e o sistema intermediário.

O sistema intermediário local (SIL) é a parte do sistema intermediário residente nos nós ativos. Ele é interceptado pelo usuário quando este deseja comunicar-se com o ambiente distribuído; logo, o SI é a união de todos os SIL.

Este conjunto de definições está ilustrado na figura (IV.2), bem como os tipos de nós em um sistema distribuído.

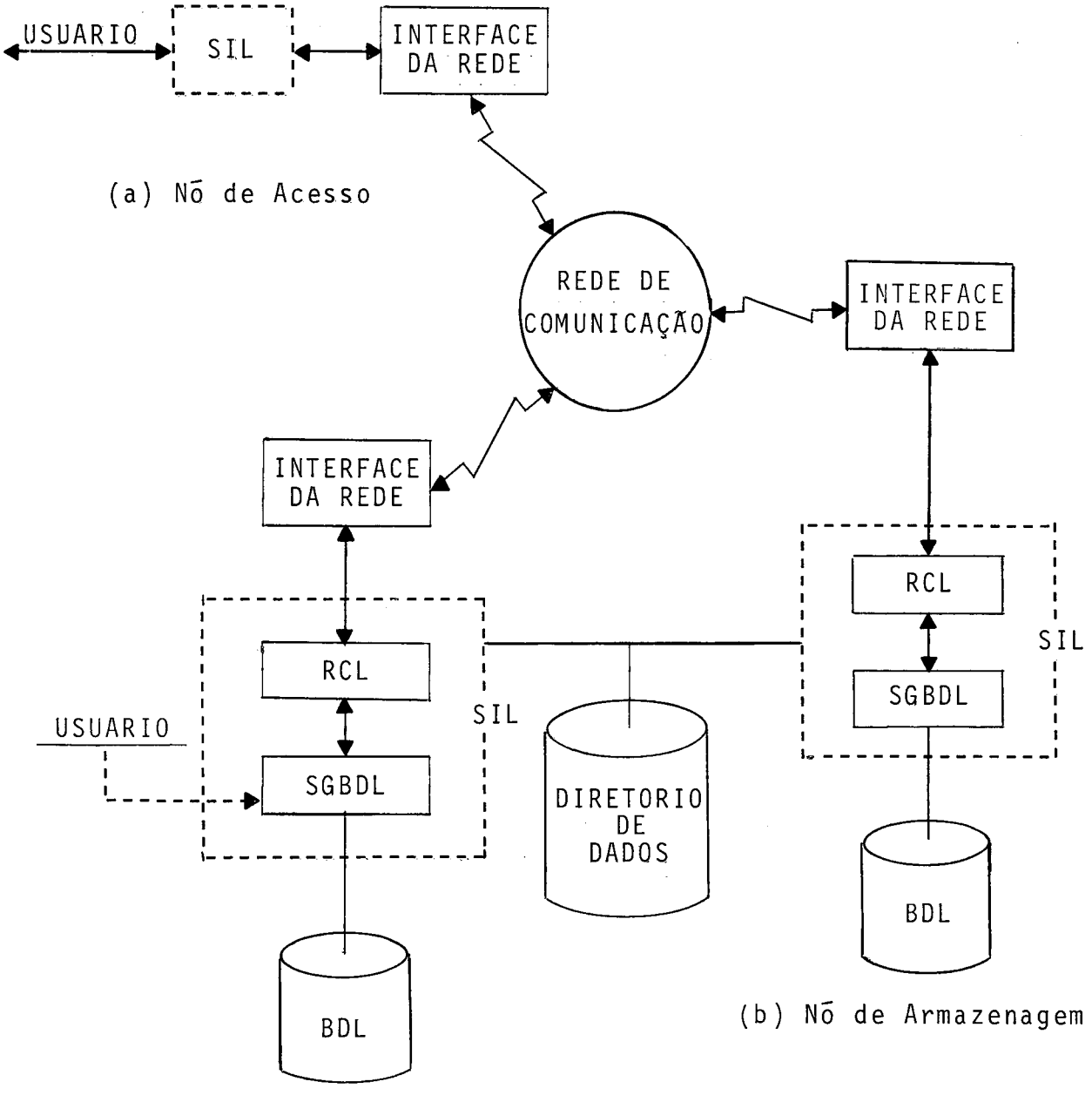


FIGURA IV.2 - Tipos de Nós e Componentes do Sistema de Processamento de Dados Distribuído

IV.3.2 - Estrutura do Sistema Proposto

O diagrama da figura (III.3) mostra a estrutura funcional do sistema intermediário como um todo. Posteriormente descreveremos a funcionalidade de cada módulo.

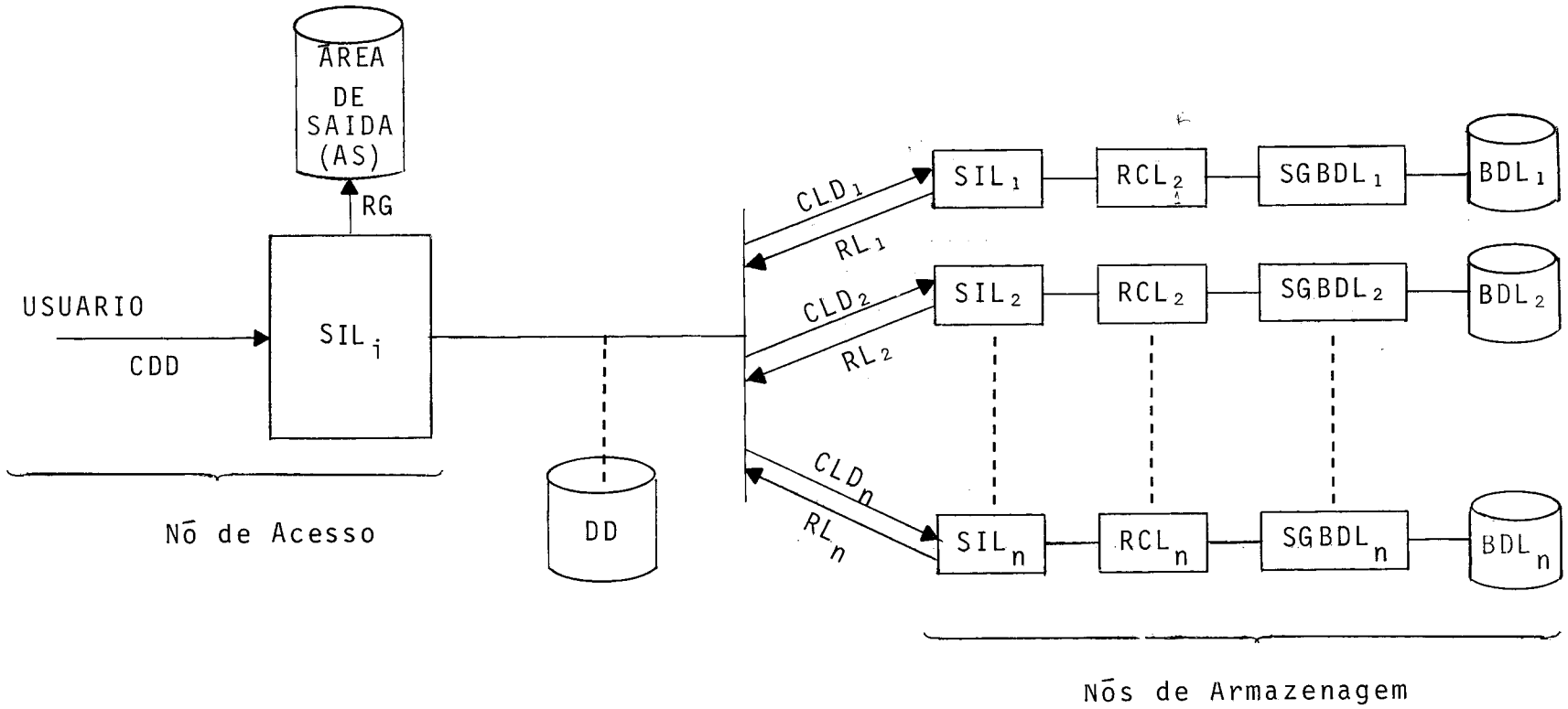


FIGURA IV.3 - Esquema de Consultas e Respostas de Dados

Legenda:
CDD: Consulta para dados distribuído; RG: Resposta global; SIL_i: Sistema intermediário local i; DD: Diretório de dados; CLD_i: Consulta local de dados para o nó i; RL_i: Resposta local do nó i; RCL_i: Rotina de Comunicação i; $SGBDL_i$: Sistema de gerência de banco de dados local i; BDL_i: Base de dados local i.

Ao ser interceptado pelo usuário, o sistema intermediário local no nó ativo efetua uma consulta ao diretório de dados (DD) para, através das unidades de dados que compõem a CDD, determinar o nó, ou os nós, ao qual deve ser enviada a CDD para uma consulta local de dados (CLD).

O sistema intermediário local do nó de armazenagem que contém os dados solicitados pelo usuário interage com a rotina de comunicação local (RCL) deste nó, para obter os registros do arquivo, especificados na CLD e transfere para o sistema intermediário local do nó que fez a consulta global apenas os registros que satisfizerem a esta CLD.

Os registros selecionados são armazenados na área de saída, em forma de arquivo auxiliar. Caso a CDD envolva mais de um arquivo, está implícito que este segundo arquivo já está armazenado na área de saída. Então, o sistema intermediário que fez a consulta global trata de completar a operação envolvendo os registros obtidos do ambiente distribuído com os registros do arquivo existente na área de saída, para logo depois comunicar ao usuário o sucesso ou insucesso da operação.

IV.3.2.1 - Descrição Funcional dos Módulos

Dentre os módulos da figura (IV.3), o sistema intermediário será assunto do próximo capítulo. Neste ponto será feita uma descrição de como deve funcionar cada um dos outros módulos, tomando como preocupação principal a filosofia das operações que serão sugeridas no capítulo seguinte, bem como a visão do usuário em relação ao ambiente distribuído.

O usuário do nó ativo comunica-se com o ambiente distribuído, onde as bases de dados locais estão logicamente organizadas de acordo com qualquer um dos modelos de banco de dados, utilizando-se de declarações, funções e comandos definidos para o sistema intermediário. Ele deve "ver" o ambiente do banco de dados distribuído como um conjunto de arquivos os quais poderá duplicá-los total ou parcialmente (conforme uma condição) sem, contudo, pensar em executar nenhuma operação que não seja a de seleção de arquivos ou conjunto de dados.

Devem ser permitidas ao usuário, as seguintes alternativas:

a) inicializar a área de saída com um grupo de registros obtidos de qualquer base de dados e em qualquer momento em que esteja se utilizando do sistema intermediário, oferecendo, desta maneira, condições de reiniciar a seleção de dados caso tenha cometido algum erro - registros obtidos e não desejados, por exemplo - durante a consulta ou se deseja uma nova aplicação após o término de uma outra;

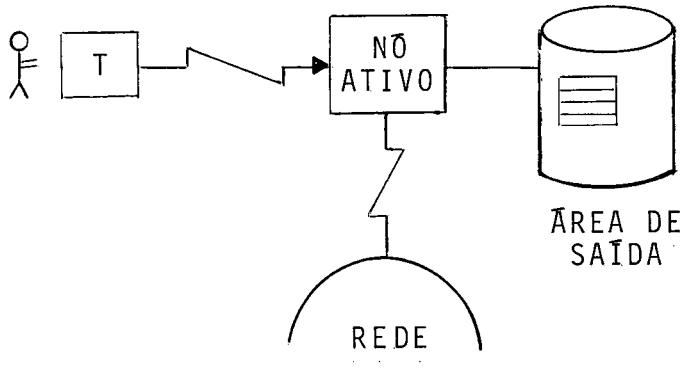


FIGURA IV.4 - Inicializar área de saída

b) acessar qualquer arquivo do ambiente distribuído sem que seja necessário referenciar em qual base de dados ou nó ele está armazenado e nem se preocupar com sua estrutura lógica ou física;

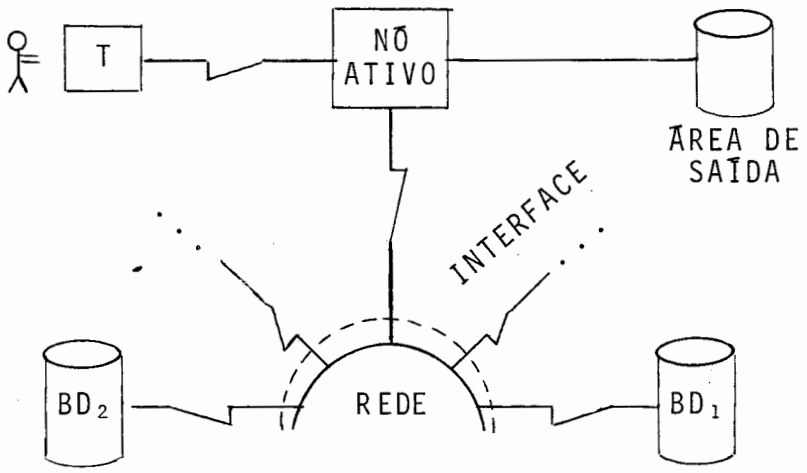


FIGURA IV.5 - Interface localizando o pedido de informação do usuário

c) inserir novos registros logo depois do último existente na área de saída, permitindo ao usuário unir arquivos, ou relações, compatíveis entre si;

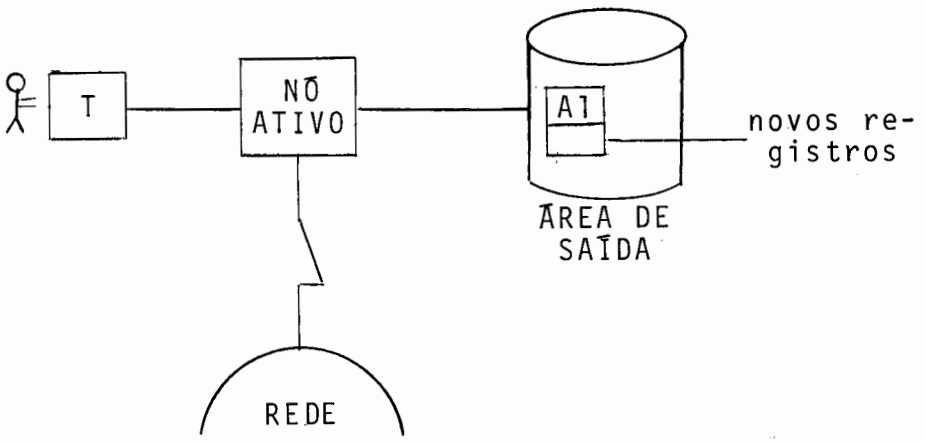


FIGURA IV.6 - Inserção de registros na área de saída

d) inserir novos registros na área de saída para que sejam tratados como um outro arquivo (arquivo-lógico). Neste caso deverá ser colocada uma marca de fim de arquivo para os registros existentes, fazendo com que qualquer operação envolvendo os registros contidos na área de saída seja efetuada somente sobre os novos registros inseridos (arquivo lógico em uso). Isto faz com que o usuário possa ver a área de saída como um conjunto de arquivos lógicos, apesar de que apenas o último destes arquivos esteja disponível para futuras operações envolvendo um arquivo do ambiente distribuído e um arquivo da área de saída;

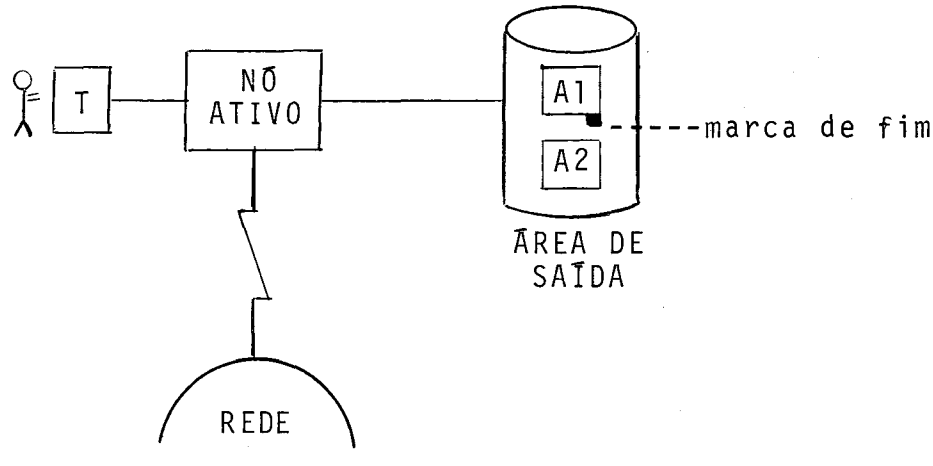


FIGURA IV.7 - Geração de Arquivos Lógicos (A1, A2) na Área de Saída

e) em uma operação envolvendo dois arquivos, o primeiro poder estar armazenado no ambiente distribuído ou poder ser qualquer arquivo lógico da área de saída. O segundo arquivo, será sempre o arquivo lógico em uso (aberto);

f) ampliar a dimensão dos registros do arquivo lógico em uso na área de saída. Seria o caso de o usuário desejar efetuar uma operação de junção - similar a proposta por CODD no modelo relacional, envolvendo o arquivo lógico em uso e um arquivo do ambiente distribuído ou um arquivo lógico da área de saída - em ambos os casos, o arquivo ampliado será o arquivo lógico em uso. Somente o segundo arquivo precisa ser referenciado na consulta feita pelo usuário ao sistema intermediário - o arquivo do ambiente distribuído ou o arquivo lógico da área de saída;

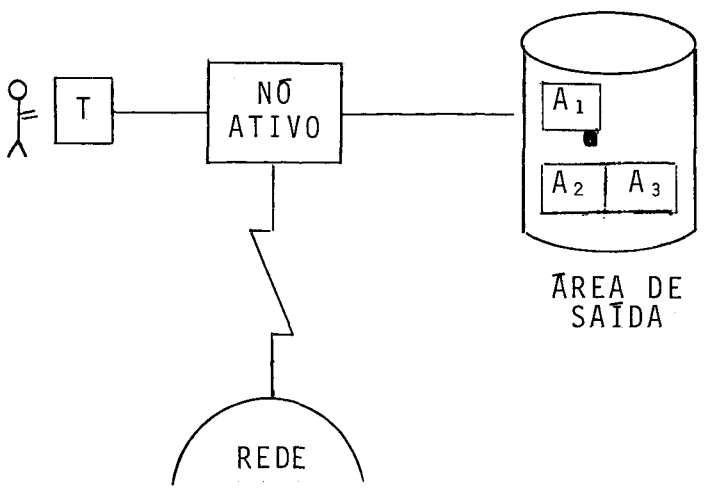


FIGURA IV.8 - Junção de Arquivos na A.S.

g) solicitar informações sobre os arquivos declarados pelo usuário ao sistema intermediário como também os arquivos que compõem o ambiente do banco de dados distribuído. A necessidade de o usuário declarar anteriormente o arquivo que deseja acessar do ambiente do banco de dados distribuído é devida à sua independência

com relação às definições desses arquivos nas suas bases de dados locais. O usuário do sistema intermediário, não necessariamente, precisa ter conhecimento dos campos (tamanho, nome e tipo) do arquivo que deseja acessar tendo em vista que os registros desse arquivo serão tratados como cadeia de caracteres e seus campos podem ser redefinidos pelo usuário sem nenhum prejuízo para as definições contidas nos esquemas das bases de dados locais;

h) solicitar todos os registros contidos na área de saída;

i) obter informações gerais sobre a área de saída, tais como: tamanho dos arquivos lógicos; vista dos registros de um arquivo lógico; número de arquivos lógicos gerados; etc.

IV.3.2.1.1 - Área de Saída

Deve ser gerenciada, visando ao atendimento das alternativas oferecidas ao usuário, de maneira tal que ofereça condições de se efetuar as operações seguintes:

a) ser inicializada - aberta - ou reinicializada em qualquer tempo. No caso da reinicialização, todo o conteúdo da área de saída será perdido, ou seja, o próximo registro será gravado no início da área de saída - veja fig. (IV.4);

Exemplo:

ABRIR AREA DE SAIDA

---------- } aplicação-1

ABRIR AREA DE SAIDA

---------- } aplicação-2

b) inserir um novo registro - ou grupo de registros - no final do arquivo lógico em uso. Chamamos de arquivo lógico em uso o grupo de registros que ainda não foi fechado com a marca de fim de arquivo - veja figura (IV.6) - exemplo: suponha que o arquivo lógico em uso contêm as informações sobre os alunos matriculados no primeiro período de 1981 - número de matrícula iniciado em 811 - e se deseja unir este grupo de alunos aos matriculados no segundo período do mesmo ano - matrícula iniciada em 812;

INSERIR ALUNOS (ALUNO# GT 8120000)

c) oferecer condições de expansão dos registros contidos no arquivo lógico em uso para o caso de uma junção - veja figura (IV.8) - exemplo: suponha que se deseje acrescentar aos registros dos alunos contidos no arquivo lógico em uso do exemplo anterior, as disciplinas já cursadas por cada um - histórico escolar.

JUNTAR HISTORICO (ALUNO# EQ A#)

O campo A# deve ter sido declarado, anteriormente, para o arquivo lógico em uso;

d) inserir uma marca de fim de arquivo lógico no final de um grupo de registros da área de saída e passar a tratar os novos registros inseridos como um outro arquivo lógico. Quer dizer que, qualquer comando, referenciando o arquivo lógico em uso da área de saída, manipulará os novos registros inseridos - veja figura (IV.7) - exemplo: suponha que se deseje, dando continuidade aos exemplos anteriores, criar mais um arquivo na área de saída desta vez com as informações sobre todos os funcionários da escola;

OBTER NOVO FUNCIONARIOS

e) gerar uma seqüência de registros sobre o arquivo lógico em uso. Neste caso o arquivo lógico em uso será eliminado, dando lugar a uma nova seqüência de registros sem prejuízo dos anteriormente gerados - exemplo: suponha que, no exemplo do item anterior, era desejado, apenas, os funcionários professores.

OBTER SOBRE FUNCIONARIOS (FUNCAO EQ 'PROFESSOR')

Os comandos usados para exemplificar cada item, serão apresentados no próximo capítulo deste trabalho.

Visto que será permitido ao usuário aglomerar um certo grupo de registros formando um arquivo lógico, o sistema intermediário deverá manter uma tabela, ou diretório dos arquivos lógicos, contendo as seguintes informações:

NÚMERO DE ORDEM	NOME	ENDEREÇO INICIAL	NÚMERO DE REGISTROS	TAMANHO MÁXIMO REGISTRO
1	FUNCIONARIO	1	150	65
2	-	9751	30	50
3	DEPARTAM.	11252	60	80

FIGURA IV.9 - Tabela dos Arquivos Lógicos da Área de Saída

a) NÚMERO DE ORDEM - indica a ordem de geração do quivo lógico, ou seja, o valor 1 (um) para o primeiro, 2 (dois) para o segundo e sucessivamente. Estes valores poderão ser usados pelo usuário para referenciar algum arquivo lógico na área de saída;

b) NOME - quando o arquivo lógico é inicializado não tem um nome específico visto que sua geração é automática por parte do SI. O mesmo é referenciado utilizando-se do número de ordem até que o usuário lhe atribua um nome. Futuramente será sugerida uma função pela qual pode-se atribuir um nome a estes arquivos lógicos

por parte do usuário. É por esta razão que para o arquivo de ordem 2 (dois) omitimos o seu nome.

A atribuição de nomes aos arquivos lógicos da área de saída não é obrigatória, dado que o usuário pode referenciá-los pelo seu número de ordem. A flexibilidade de atribuir-lhes nomes é oferecida para que o usuário possa relacionar o conteúdo de cada arquivo lógico com um código por ele atribuído;

c) ENDEREÇO INICIAL - contém o endereço físico de onde começa o arquivo lógico;

d) NÚMERO DE REGISTROS - à medida que um registro é inserido este valor é acrescido de 1 (um);

e) TAMANHO DOS REGISTROS - tamanho máximo dos registros contidos neste grupo de registros (arquivo lógico).

O usuário deve ter conhecimento de que, à medida que é colocada uma marca de fim de arquivo em um arquivo lógico, não será mais permitido inserir novos registros no mesmo, tendo em vista que a área de saída é gerada como sendo um arquivo seqüencial e os registros são inseridos no final do mesmo.

IV.3.2.1.2 - Rotina de Comunicação Local

É um procedimento construído com exclusividade pa

ra cada tipo de sistema de gerência de banco de dados distribuído.

Este procedimento pode ser programado utilizando-se os próprios recursos oferecidos pelo sistema de gerência de banco de dados local, atentando para o fato de que suas funções principais são: (1) verificar se o usuário está autorizado ao acesso aos dados; (2) recuperar - do primeiro ao último - todos os registros de um determinado arquivo, ou relação.

O teste para verificar se cada registro atende ou não a consulta elaborada pelo usuário será encargo do sistema intermediário que chamou a rotina de comunicação local.

O sistema intermediário solicitará desta rotina um a um dos registros que compõem o arquivo a ser acessado e terminará assim que a consulta do usuário for atendida plenamente ou quando forem esgotados todos os registros de determinado arquivo.

A comunicação com o sistema intermediário será feita através dos seguintes parâmetros:

(<USUARIO>, <NOME-DE-ARQUIVO>, <REGISTRO>,
<VALOR>, <TAMANHO-DO-REGISTRO>, <ESTADO>)

onde:

a) <USUARIO> - contém o código que identifica o usuário na base de dados local;

b) <NOME-DE-ARQUIVO> - contém o nome do arquivo, ou relação, que se deseja acessar na base de dados local;

c) <REGISTRO> - um vetor suficientemente capaz de receber a cadeia de caracteres que forma o registro solicitado;

d) <VALOR> - contém o número de ordem do registro a ser acessado (ordem de sequência de acesso);

e) <TAMANHO-DO-REGISTRO> - retornará com um número inteiro positivo informando o número de caracteres do registro acessado. Esta informação é fornecida pela rotina de comunicação local;

f) <ESTADO> - retornará com um código de estado da operação - final de arquivo, por exemplo. Esta informação é fornecida pela rotina de comunicação local.

IV.3.2.1.3 - Diretório de Dados

Sistemas de gerência de banco de dados empregam diretórios, ou esquemas, para definir a natureza do banco de dados gerenciado. O propósito do diretório é dar um grau de independência ao usuário, ou programa de aplicação, evitando o fornecimento de informações quanto à natureza das informações, e simplificar vários tipos de mudanças na estrutura do banco de da-

dos. Um diretório de dados, tipicamente, contém quatro tipos de informações, DAVENPORT[9]:

a) definição da estrutura lógica (nomes de registros, nomes de relações, domínios, etc.);

b) definição da estrutura física (formato dos campos, campos invertidos, etc.);

c) controles de acessos aos dados (para leitura, inserção, deleção ou alteração);

d) estatísticas de uso.

Para um banco de dados distribuído, uma categoria de informação adicional deve ser inserida no diretório: a localização de cada peça do banco de dados na rede, suas duplicações e seus particionamentos.

O "software" do sistema (o SGBD e os componentes da rede) deve ter acesso a estas informações para analisar requisições do usuário, escolher e executar um método de acesso e decidir quais os recursos a serem usados. Portanto, uma decisão tem que ser tomada com referência à estrutura é onde o diretório está localizado.

CHU[3], tem investigado a performance de quatro estruturas de diretório. Em cada uma delas, ele considera que cada nó tem seu próprio diretório local que contém informações sobre os dados por ele armazenado. As estruturas são:

IV.3.2.1.3.1 - Diretório Centralizado

Um diretório mestre está localizado por um dos nós da rede. Quando um usuário requisita dados que não estão armazenados pelo seu diretório local, o diretório mestre é consultado. O diretório centralizado deve ser regravado quando há uma mudança na localização de armazenagem ou no caso de uma adição ou deleção de uma seção do banco de dados. Existe um custo de comunicação para cada transação que requer dados remotos.

IV.3.2.1.3.2 - Diretório Centralizado e Estendido

É uma modificação do diretório centralizado. Uma vez obtida a localização ou descrição dos dados requeridos pelo usuário, estas informações são copiadas para o diretório local onde está o usuário ativo. Caso o usuário requeira os dados novamente, as informações necessárias para sua obtenção estão localizadas no seu diretório local reduzindo, desta forma, o custo de comunicação bem como o tempo de consulta do diretório mestre. Entretanto, quando as informações sobre tais dados são alterados no diretório mestre, estas informações devem, também, ser alteradas no diretório local, envolvendo um custo de comunicação.

IV.3.2.1.3.3 - Diretório Local

No caso do diretório local, não existe um diretório mestre no sistema. Quando uma informação sobre dados requisitados não está armazenada no diretório local ao usuário que fez a consulta, todos os outros diretórios locais existentes no sis-

tema são consultados até que os dados requisitados sejam localizados. Esta organização requer um alto custo de comunicação.

IV.3.2.1.3.4 - Diretório Distribuído

No caso do diretório distribuído, cada nó do sistema tem um diretório mestre - uma cópia. A vantagem para o sistema é o seu baixo tempo de resposta. As desvantagens são os custos de armazenagem destes diretórios por parte de cada nó e o de comunicação para atualização de todos estes diretorios.

As conclusões gerais das investigações de Chu, considerando que o custo de transmissão é bem maior do que o custo de armazenagem, mostram que, em diretórios com baixa taxa de modificação de arquivos, o diretório distribuído fornece o menor custo operacional. Quando a taxa de modificação cresce, o diretório centralizado e estendido fornece a solução mais econômica, enquanto para altas taxas de modificação a solução centralizada é mais barata.

Outra solução seria particionar o diretório mestre entre os vários nós de armazenagem da rede. O diretório poderia, por exemplo, ser particionado por índice, da seguinte maneira:

a) seja um sistema descentralizado composto pelos nós de armazenagem C_1, C_2, C_3 e C_4;

b) as informações sobre os conjuntos de dados, cujos nomes comecem com as letras de A-F estariam

armazenadas no diretório do nó C_1, de G - L no diretório do nó C_2, de M - Z no diretório do nó C_3 e de S - Z no diretório do nó C_4.

Desta forma o custo de armazenagem estaria dividido entre os vários nós de armazenagem e seria muito reduzido. Quanto ao custo de comunicação e tempo de consulta, seria reduzido se os usuários de cada centro tomassem consciência de que a atribuição de nomes às relações, ou conjuntos de dados, por ele criada, deve ser feita dentro do intervalo de índices que foi atribuído para o diretório do seu centro.

A tabela da figura (IV.10) pode auxiliar na escolha de um dos tipos de diretório, através da atribuição de pesos aos parâmetros influenciadores na decisão de por qual tipo de diretório optar.

Note que alguns parâmetros devem ser analisados como sendo mais influenciadores do que outros - o custo de comunicação em relação ao custo de armazenagem, por exemplo. Mostraremos alguns pontos que devem ser analisados na atribuição de peso para cada parâmetro, isoladamente:

a) CUSTO DE ARMAZENAGEM: deve-se verificar a capacidade de armazenamento de cada nó de armazenagem em relação ao volume de dados do diretorio, bem como o número de duplicações das informações necessárias para formar este diretório;

b) CUSTO DE COMUNICAÇÃO: deve ser avaliado levan-

do-se em conta consultas em outros nós da rede em relação ao nó onde se encontra o usuário que fez a consulta;

c) TEMPO DE CONSULTA: está relacionado com a distância entre o nó ativo e o nó que armazena a informação desejada no diretório de dados. O algoritmo de consultas do diretório deve ser levado em conta quanto ao tempo médio para recuperar uma informação do diretório - suponha que, para diretórios volumosos o algoritmo seja menos eficiente por se utilizar de uma pesquisa seqüencial, por exemplo;

d) VOLATILIDADE: refere-se ao volume de modificações aplicado normalmente no diretório de dados. Deve ser analisada em relação ao CUSTO DE COMUNICAÇÃO, com exceção do diretório local, e ao volume de duplicações das informações contidas no diretório.

Os pesos existentes na tabela foram atribuídos da seguinte maneira: 1) o custo de armazenagem, foi analisado somente quanto ao volume de informações duplicadas; 2) o custo de comunicação, foi analisado em relação às consultas em outros nós - note que, para os diretórios centralizado/estendido e o particionado, o peso varia de acordo com as informações que podem ser obtidas no próprio nó onde se encontra o usuário; 3) o tempo de consulta, em relação, também, a buscas de informações em outros nós; 4) a volatilidade, para os quatro primeiros tipos de diretório -

centralizado, centralizado e estendido, local e distribuído - está baseada nas investigações de CHU[3], enquanto, para o diretório particionado, está competindo com os dois primeiros - centralizado e centralizado/estendido.

TIPOS DE DIRETÓRIO	CUSTO DE ARMAZENAGEM	CUSTO DE COMUNICAÇÃO	TEMPO DE CONSULTA	VOLATILIDADE		
				BAIXA	MÉDIA	ALTA
CENTRALIZADO	4	2	2	-	-	4
CENTRALIZADO E ESTENDIDO	4 - 2	2 - 4	2 - 4	-	4	-
LOCAL	4	1	1	-	-	-
DISTRIBUÍDO	1	4	4	4	-	-
PARTICIONADO	4	2 - 4	3 - 4	-	4	4

PESOS 1 - Deficiente
2 - Regular
3 - Bom
4 - Ideal

FIGURA IV.10 - Comparação entre os Tipos de Diretório

CAPÍTULO V - O SISTEMA INTERMEDIÁRIO

V. O SISTEMA INTERMEDIÁRIO

V.1 - Introdução

O objetivo deste capítulo é apresentar o sistema intermediário através de uma linguagem que fornece os recursos necessários para comunicação entre o usuário e o ambiente de processamento de dados distribuído. Lembramos, mais uma vez, que a filosofia do sistema intermediário é a seleção de registros a partir dos arquivos existentes nas bases de dados do ambiente distribuído.

A linguagem que se sugere é composta de declarações, funções e comandos sem a necessidade de se estabelecer uma regra de prioridade nas suas utilizações por parte do usuário.

V.2 - Declarações

Tem como finalidade declarar um arquivo, que será referenciado posteriormente por um comando, de uma das bases de dados do ambiente distribuído, ou declarar um campo para um arquivo declarado anteriormente pelo usuário ou para um arquivo lógico da área de saída.

Um arquivo do banco de dados distribuído, não necessariamente, precisa ser declarado para que o usuário possa acessá-lo. A declaração de arquivo vem a oferecer ao usuário condições de redefinir os campos de um determinado arquivo, ou atribuir outro nome a um campo, caso não lembre exatamente o que lhe foi atribuído no ato da sua geração (o nome que está contido no

esquema ou diretório de dados), para efetuar alguma operação que envolva os campos do citado arquivo.

Isto é possível porque os registros obtidos do ambiente distribuído são tratados como cadeia de caracteres e suas redefinições não alteram suas características originais, tendo em vista que a declaração é feita para o sistema intermediário e não para o ambiente distribuído.

V.2.1 - Declaração de Arquivos

Tem como função informar ao sistema intermediário o nome de um arquivo existente no ambiente do banco de dados distribuído e o tamanho máximo dos seus registros, para ser usado, posteriormente, nos comandos da linguagem.

FORMATO

DECLARE ARQUIVO (<nome-de-arquivo>,
<tamanho-do-registro>)

onde:

a) <nome-de-arquivo> - deve conter o mesmo nome de um arquivo existente no ambiente distribuído;

b) <tamanho-do-registro> - um número inteiro positivo e deve ser igual ao tamanho máximo dos registros que compõem o determinado arquivo.

V.2.2 - Declaração de Campos

Tem como finalidade definir um campo para os registros de um determinado arquivo, com fins de utilização nos comados da linguagem.

O arquivo para o qual o campo será definido, pode ser: 1) um arquivo declarado anteriormente pelo usuário, não necessariamente logo após a declaração do arquivo; 2) um arquivo lógico da área de saída.

Os parâmetros da declaração de campo não precisam ser compatíveis em tipo, "label" ou número de caracteres com os do ambiente distribuído - mesmo quando usada para arquivos do ambiente distribuído - tendo em vista que serão manipulados somente pelo sistema intermediário.

FORMATO

DECLARE CAMPO (<nome-de-arquivo>[(BD)],
<nome-de-campo>,
<tamanho-do-campo>,
<posição-inicial>)

onde:

a) <nome-de-arquivo> - deve conter um nome de arquivo declarado anteriormente pelo usuário acompanhado da cláusula (BD), o nome de um arquivo lógico ou o parâmetro SAIDA(n), com n inteiro positivo, referenciando o arquivo lógico pela ordem em que foi gerado na área de saída;

b) <nome-de-campo> - deve conter um nome para o campo atribuído pelo usuário;

c) <tamanho-do-campo> - deve conter um número inteiro positivo, o qual determina o tamanho total do campo;

d) <posição-inicial> - deve conter um número inteiro positivo que determina onde começa o campo dentro dos registros.

Para exemplificar o uso das declarações de arquivo e de campo, usaremos um arquivo de funcionários contendo as seguintes informações:

```
01 FUNCIONARIO
   02 MATRICULA  PIC(5)
   02 NOME       PIC(30)
   02 ENDEREÇO   PIC(24)
   02 DATA-DE-ADM.
      03 DIA  PIC(2)
      03 MÊS  PIC(2)
      03 ANO  PIC(2)
```

1) DECLARE ARQUIVO (FUNCIONARIO, 65)

Com esta declaração, o sistema intermediário verifica a existência de um arquivo de nome FUNCIONARIO no ambiente distribuído, com registro de, no máximo, 65 caracteres.

2) Suponha que o usuário deseje acessar os funcionários admitidos em um determinado dia, mês e ano e, posteriormente, os admitidos em um certo mês do ano:

```
DECLARE CAMPO (FUNCIONARIO, DMA, 6, 60)
-------------
-------------
-------------  comandos
-------------

DECLARE CAMPO (FUNCIONARIO, MA, 4, 62)
-------------
-------------
-------------  comandos
-------------
```

é importante saber que o campo DMA, declarado primeiro, continuará existindo; para isto, o sistema intermediário deve manter uma tabela, ou esquema, com as seguintes informações:

ARQUIVO	CAMPO	CARACTERES	INICIO
FUNCIONARIOS	DMA	6	60
FUNCIONARIOS	MA	4	62

FIGURA V.1 - Tabela de Campos

V.3 - Funções

As funções serão definidas com o propósito de oferecer condições ao usuário de obter, através do sistema intermediário, informações gerais sobre:

a) os arquivos definidos pelo usuário;

b) os arquivos não definidos pelo usuário mas existentes no ambiente distribuído;

c) os arquivos lógicos contidos na área de saída.

Na apresentação das funções, usaremos os seguintes termos:

a) BD - esta cláusula indica que o usuário está referindo-se a um arquivo existente no ambiente distribuído;

b) AS - esta cláusula indica que o usuário está referindo-se à área de saída;

c) <nome-de-arquivos> - deve conter o nome do arquivo do qual se deseja a informação. O parâmetro SAIDA(n), com n inteiro positivo, pode ser usado referenciando um arquivo lógico da área de saída pela ordem em que foi gerado;

d) as cláusulas BD e AS podem ser omitidas: 1) se o espaço que lhes foi reservado estiver em branco, fica implícito que o conteúdo de <nome-de-arquivo> refere-se ao ambiente distribuído; 2) quando o parâmetro SAIDA(n) for usado no espaço <nome-de-arquivo>;

e) <nome-de-arquivo-lógico> - deve conter um nome de arquivo da área de saída.

V.3.1 - Função NREG

Usada para obter de um determinado arquivo, o número de registros que ele contém.

FORMATO

NREG { BD / AS } <nome-de-arquivo>

Exemplos:

NREG BD FUNCIONARIOS

NREG AS ALUNOS

NREG FUNCIONARIOS

NREG SAIDA(1)

V.3.2 - Função TREG

Usada para solicitar o tamanho máximo dos registros de um determinado arquivo.

FORMATO

TREG { BD / AS } <nome-de-arquivo>

Exemplos:

TREG BD FUNCIONARIOS

TREG AS ALUNOS

TREG FUNCIONARIOS

TREG SAIDA(1)

V.3.3 - Função CREG

Usada para solicitar que sejam mostrados os campos que compõem os registros de um determinado arquivo. Para os arquivos lógicos da área de saída, serão mostrados todos os campos do arquivo declarados pelo usuário.

FORMATO

CREG { BD / AS } <nome-de-arquivo>

Exemplos:

CREG BD FUNCIONARIOS

CREG AS ALUNOS

CREG FUNCIONARIOS

CREG SAIDA(1)

V.3.4 - Função ESQM

Serão mostrados os campos, tamanho máximo dos registros e o número de registros de um determinado arquivo.

O uso da cláusula AS, sem um nome-de-arquivo, implica mostrar, ao usuário, todo o conteúdo do esquema, ou diretório, mantido pelo sistema intermediário para os arquivos lógicos que foram gerados na área de saída, conforme a figura (V.2).

NÚMERO DE ORDEM	NOME	ENDEREÇO INICIAL	NÚMERO DE REGISTROS	TAMANHO MÁXIMO DOS REGISTROS
1	OFERTA	1	150	65
2	-	9751	30	50
3	DEPARTAMENTO	11252	60	80

FIGURA V.2 - Diretório dos Arquivos Lógicos da Área de Saída

FORMATO

ESQM { BD / AS } [<nome-de-arquivo>]

Exemplos:

ESQM BD FUNCIONARIOS

ESQM FUNCIONARIOS

ESQM AS ALUNOS

ESQM AS

ESQM SAIDA(1)

V.3.5 - Função NOME

Usada para atribuir um nome, à escolha do usuário, a um arquivo lógico da área de saída identificado pela variável n contida no parâmetro SAIDA(n).

FORMATO

NOME[SAIDA](n)<nome-do-arquivo-lógico>

Exemplos:

NOME SAIDA(3) OFERTA

NOME(3) OFERTA

V.3.6 - Função IMPR

Usada para solicitar que sejam mostrados no vídeo de um computador, ou terminal, os registros pertencentes a um determinado arquivo.

FORMATO

IMPR { BD / AS } <nome-de-arquivo>

Exemplos:

IMPR BD FUNCIONARIOS

IMPR FUNCIONARIOS

IMPR AS OFERTA

IMPR SAIDA(3)

V.4 - Comandos

Os comandos sugeridos têm as seguintes finalidades:

a) acessar os registros de um determinado arquivo do ambiente distribuído e copiá-los para área de saída;

b) efetuar operações envolvendo dois arquivos: 1) o arquivo referenciado no comando é o do ambiente distribuído; 2) o arquivo referenciado no comando é o da área de saída; 3) o segundo arquivo é sempre o arquivo lógico em uso na área de saída.

Os termos usados nas definições dos comandos têm as mesmas características de quando usados nas funções - veja parágrafo (V.2).

Para exemplificarmos os comandos, utilizaremos as bases de dados das figuras (V.3, V.4 e V.5). Na representação

das bases de dados não nos preocupamos com sua estrutura lógica, visto que o usuário deverá "ver" as bases de dados existentes no ambiente distribuído como um conjunto de arquivos.

ARQUIVOS	CAMPOS	Nº DE CARACTERES
FUNCIONARIOS ------	FUNC#	05
	NOME	30
	FUN	15
FUNCAO ----------	FUN	15
	SALARIO	10
DEPENDENTES -------	FUN#	05
	NOME	30
	DNASC	06

FIGURA V.3 - Base de Dados BD1

ARQUIVOS	CAMPOS	Nº DE CARACTERES
CURSOS ----------	CURSO#	05
	NOME	20
ALUNOS ----------	ALUNO#	07
	NOME	30
OFERTA ----------	CURSO#	05
	FUNC#	05
	SALA	03
	VAGAS	03
MATRICULA ---------	CURSO#	05
	ALUNO#	07
	PERIODO	03

FIGURA V.4 - Base de Dados BD2

ARQUIVOS	CAMPOS	Nº DE CARACTERES
PROJETOS ----------	PROJ#	06
	NOME	30
FINANCIADOR -------	ORGAO	30
	VALOR	12
EMPREGADOS --------	FUNC#	05
	PROJ#	06
	FUNC	15

FIGURA V.5 - Base de Dados BD3

V.4.1 - Abrir Área de Saída

Deve ser usado em três situações:

a) no início da aplicação, obrigatoriamente, alocando a área de saída para o usuário;

b) durante o decorrer da aplicação com a finalidade de reiniciar a área de saída, ou seja, todo o seu conteúdo será perdido e o próximo registro será inserido no início da área de saída;

c) ao final de uma aplicação e início de uma outra.

FORMATO

ABRIR AS

Exemplo:

ABRIR AS

========== aplicação 1

ABRIR AS

========== aplicação 2

V.4.2 - Comando OBTER

Tem o propósito de recuperar os registros de um arquivo do ambiente distribuído e copiá-los para a área de saída.

Para melhor entendimento deste comando, lembramos duas particularidades da área de saída: 1) pode ser dividida em vários arquivos lógicos; 2) o arquivo lógico em uso é o arquivo da área de saída que ainda não foi fechado por marca de fim-de-arquivo.

FORMATO

$$\text{OBTER} \left\{ \begin{matrix} \text{NOVO} \\ \text{SOBRE} \end{matrix} \right\} \langle\text{nome-de-arquivo}\rangle$$

$$\left[\left(\langle\text{nome-de-campo1}\rangle \langle\theta\rangle \left\{ \begin{matrix} \langle\text{nome-de-campo2}\rangle \\ \langle\text{constante}\rangle \end{matrix} \right\} \right) \right]$$

onde:

- OBTER NOVO: será colocada a marca de fim de arquivo no arquivo lógico em uso - caso não seja

o primeiro arquivo lógico a ser gerado na área de saída - e passa-se a inserir os novos registros, obtidos do ambiente distribuído, gerando uma nova seqüência de registros para o arquivo lógico em uso;

- OBTER SOBRE: os registros do arquivo lógico em uso serão substituídos pelos registros do arquivo obtidos do ambiente distribuído;

- <nome-de-campo1> - deve conter um nome de campo, do arquivo referenciado no comando pelo usuário, ou seja, este campo refere-se ao arquivo referenciado no espaço <nome-de-arquivo>;

- <θ>: qualquer um dos operandos abaixo:

  LT - menor que;

  LE - menor que ou igual;

  GT - maior que;

  GE - maior que ou igual;

  EQ - igual;

  NE - diferente.

- <nome-de-campo2>: deve conter um nome de campo, declarado anteriormente pelo usuário do arquivo lógico em uso. Caso a opção SOBRE seja usada, fica implícito que o campo pertence ao ar-

quivo lógico anterior ao arquivo lógico em uso;

- <constante>: deve conter um valor numérico ou uma cadeia de caracteres entre apóstrofes ('cadeia-de-caracteres').

Exemplos:

1) Obter os dados dos funcionários que estão trabalhando no projeto 821152.

```
OBTER NOVO EMPREGADOS (PROJ# EQ 821152)
NOME  SAIDA(1)  EMP
DECLARE CAMPO (EMP, F#, 5, 1)
OBTER NOVO FUNCIONARIOS (FUNC# EQ F#)
```

2) Obter os alunos matriculados nos períodos superiores a 801 (primeiro período de 1980)

```
OBTER NOVO MATRICULA (PERIODO GT 801)
```

3) Obter os alunos matriculados a partir 821 (primeiro período de 82).

```
OBTER SOBRE MATRICULA (PERIODO GE 821)
```

V.4.3 - Comando INSERIR

Passa a inserir os registros do arquivo referenciado no comando, logo após o último registro do arquivo lógico em uso da área de saída.

Ao usar este comando para arquivos distintos, o usuário deve tomar o cuidado de verificar se os campos dos registros destes arquivos são compatíveis em tipo e número de caracteres.

FORMATO

INSERIR { BD / AS } <nome-de-arquivo>

[{<nome-de-campo1><θ>{<nome-de-campo2> / <constante>}}]

Exemplos:

1) Obter os funcionários professores e os diretores administrativos, bem como seus respectivos salários.

```
OBTER NOVO FUNCIONARIOS (FUN EQ 'PROFESSOR')
INSERIR BD FUNCIONARIOS (FUN EQ 'DIRETOR ADM')
DECLARE CAMPO (SAIDA(1), FN, 15, 36)
JUNTAR FUNCAO (FUN EQ FN)
```

- veja parágrafo (V.3.4) -

2) Obter os cursos dos períodos 801 e 811.

```
OBTER NOVO MATRICULA (PERIODO EQ 801)
INSERIR MATRICULA (PERIODO EQ 811)
```

V.4.4 - Comando JUNTAR

O resultado desta operação é a concatenação dos registros contidos no arquivo em uso com os registros do arquivo referenciado no comando, que satisfazem a comparação entre os campos referenciados no comando.

FORMATO

JUNTAR { BD / AS } <nome-de-arquivo>

(<nome-de-campo1><θ><nome-de-campo2>)

Exemplos:

1) Obter os funcionários que são professores e estão trabalhando no projeto 821152.

```
OBTER NOVO FUNCIONARIOS (FUN EQ 'PROFESSOR')
OBTER NOVO EMPREGADOS (PROJ# EQ 821152)
DECLARE CAMPO (SAIDA(1), F1, 5, 1)
DECLARE CAMPO (SAIDA(2), F2, 6, 6)
JUNTAR SAIDA(1) (F1 EQ F2)
```

V.4.5 - Comando COPIAR

Tem como finalidade copiar os registros contidos na área de saída para a área de trabalho do usuário ou para um determinado periférico. Caso a área de saída esteja dividida em

arquivos lógicos, será permitido ao usuário: 1) copiar cada arquivo isoladamente; 2) copiar toda a área de saída através da cláusula AS.

FORMATO

$$\text{COPIAR} \left\{ \begin{array}{l} \text{<nome-de-arquivo-lógico>} \\ \text{AS} \end{array} \right\} \left(\left\{ \begin{array}{l} \text{<área-de-trabalho-do-usuário>} \\ \text{<periférico>} \end{array} \right\} \right)$$

onde:

a) <área-de-trabalho-do-usuário> - deve conter informações sobre uma área em disco onde o usuário esteja autorizado a gerar arquivos permanentes;

b) <periférico> - unidade de saída para onde deve ser copiado os registros contidos na área de saída - fita magnética, impressora, etc.

Exemplos:

```
COPIAR AS (PRINTER)

COPIAR ARQ-TAREFAS ("especificações sobre área em
                    disco")

COPIAR SAIDA(2) (PUNCH)

COPIAR (2) (PUNCH)
```

V.4.6 - Comando ORDENAR

Tem como finalidade ordenar os registros do arquivo lógico em uso da área de saída. O usuário não necessita especificar o nome do arquivo a ser ordenado. No próprio comando fica implícito que trata-se do arquivo em aberto - arquivo lógico em uso.

FORMATO

ORDENAR { ASC / DESC } (<nome-de-campo1>)

[, { ASC / DESC } (<nome-de-campo2>)] ...

onde:

a) ASC - indica uma ordenação em ordem crescente dos valores do campo;

b) DESC - indica uma ordenação em ordem decrescente dos valores do campo;

c) <nome-de-campo1>, <nome-de-campo2>, ... - contêm nomes de campos, declarados anteriormente pelo usuário, do arquivo lógico em uso.

V.4.7 - Comando FINAL

Indica o final do processamento. Deve ser o últi

mo comando a ser usado. Caso o usuário deseje fazer uma nova aplicação, não será necessário usar este comando para mudar de aplicação, basta usar o comando ABRIR AS e a área de saída ficará livre para uma nova aplicação.

FORMATO

FINAL

Exemplo:

ABRIR AS

========== aplicação-1

ABRIR AS

========== aplicação-2

FINAL

CAPÍTULO VI - CONCLUSÕES

VI. CONCLUSÕES

Este trabalho foi dirigido no desejo de oferecer aos usuários de um sistema de processamento de dados distribuido, condições de obter informações do ambiente distribuído - através de comandos simples porém de fácil uso - sem, contudo, ter que preocupar-se com a estrutura original dos dados distribuídos pelos centros - nós - de uma rede de computadores eletrônicos. Desta forma, cada centro de processamento de dados - que compõe o ambiente distribuído - terá autonomia para estruturar suas informações da maneira que lhe for conveniente.

As bases de dados existentes no ambiente distribuído, não necessariamente, devem estar sendo gerenciadas por sistemas de gerência de banco de dados. Estas, podem ser arquivos convencionais contanto que: 1) suas unidades de dados estejam catalogadas no diretório de dados; 2) possam ser acessadas pela rotina de comunicação local, isto é, que existam interfaces desenvolvidas para tal.

O sistema proposto nesta tese não é completo e apresenta algumas limitações que podem ser posteriormente melhoradas. As ferramentas que propomos para oferecer ao usuário um sistema de comunicação no qual lhe fosse permitido acessar as informações contidas em um ambiente distribuído, poderiam oferecer uma maior flexibilidade se, ao invés do usuário se utilizar de uma linguagem como a que propomos neste trabalho, lhe fosse permitido utilizar-se dos recursos oferecidos pela própria linguagem de manipulação de dados existentes no nó ativo onde se encontra tal usuário.

O sistema intermediário seria projetado através de algumas primitivas, transparentes para o usuário, as quais seriam utilizadas para converter a operação de consulta feita pelo usuário em operações de acesso ao banco de dados distribuído.

Para cada sistema de gerência de banco de dados local existente no ambiente distribuído, - sistema heterogêneos -, existiriam duas interfaces, a saber:

1) a primeira, para cada pedido de informação que SGBD - local determinasse a não existência das unidades de dados na base de dados local, converteria - utilizando-se das primitivas do sistema intermediário - o pedido de informação para uma consulta ao ambiente distribuído;

2) a segunda, receberia o pedido de informação do sistema intermediário, e forneceria os dados para atender a consulta. Esta interface teria as mesmas características da rotina de comunicação apresentada neste trabalho.

Após receber os dados do ambiente distribuído, estes seriam adaptados ao modelo do sistema de gerência de banco de dados local, - relacional, rede ou hierárquico, por exemplo - e este sistema completaria a resposta para o pedido de informação do usuário.

Os dados obtidos do ambiente distribuído, não ne-

necessariamente, passariam a fazer parte da base de dados de forma permanente. Poderiam ser armazenados temporariamente em uma área de trabalho e, logo após a liberação desta área por parte do usuário, os dados nela contidos deixariam de fazer parte da base de dados em questão.

Futuros projetos nesta área:

- implementação do sistema próposto;
- desenvolvimento de facilidades de "updates";
- melhorar a interface de comunicação com o usuário.

BIBLIOGRAFIA

BIBLIOGRAFIA

[1] BROWN, P.J. - Levels of Language for Portable Software - Communications of the ACM, Vol. 15, Nº 12, 1972, pp. 1059-1062.

[2] CARDENAS, ALFONSO F. - DATABASE MANAGEMENT SYSTEMS - 1979.

[3] CHU, W.W. (1976) - Performance of File Directory Systems for Databases in Star and Distributed Networks - AFIPS NCC PROCEEDINGS, pp. 577-587.

[4] CODD, E.F. - Relational Database: A Practical Foundation for Productivity - Communications of the ACM, Vol. 25, Nº 2, 1982, pp. 109-117.

[5] COLEMAN, S.S. e outros - The Mobile Programming Systems, JANUS - Software - Practice and Experience, Vol. 4, 1974, pp.5-23.

[6] DATE, C.J. - An Introduction to Database Systems - Addison - Wesley Publishing Company - 2ª edição.

[7] DATE, C.J. - An Introduction to Database Systems - 3ª edição.

[8] DAVENPORT, R.A. - Distributed or Centralized Database - The Computer Journal, Vol. 21, Nº 1, 1976, pp. 7-14.

[9] DAVENPORT, R.A. - Design of Distributed Database Systems - The Computer Journal, Vol. 24, Nº 1, 1981, pp. 31-42.

[10] DRAFFAN, I.W. e F. POOLE - Distributed Database - Cambridge University Press, 1980.

[11] EDP ANALYZER - The Data Dictionary / Directory Founction - November 1974.

[12] FRY, JAMES P. e EDGAR H. SIBLEY - Evoluction of Database Management Systems - Computing Surveys, Vol. 8, Nº 1, 1978, pp. 7-14.

[13] FURTADO, A.L. e C.S. DOS SANTOS - Organização de Banco de Dados - Editora Campus.

[14] EFIP (International Federation for Information Processing) - Working on Database Architecture - Venice, Italy, 1979, pp.26-29.

[15] MEDMAN, MONTE JAY e outros - RISS, A Relational Database Management Systems for Minicomputers.

[16] NIJSSEN, G.M. (editado por) - Architecture and Models in Database Manegement Systems - North Holland Publishing Company, 1977.

[17] SCHERR, A.L. - Distributed Data Processing - IBM Systems Journal, 17(4), 1978, pp. 324-43.

[18] WAITE, W.M. - The Mobile Programming System: STAGE2 - Communications of the ACM, Vol. 13, Nº 7, 1970, pp. 415-421.

[19] WON KIM - Relational Database Systems - Computing Surveys, Vol. 11, Nº 13, 1979, pp. 185-211.